IntechOpen

Mobile Robots

Perception & Navigation

Edited by Sascha Kolski

Mobile Robots
Perception & Navigation

Edited by
Sascha Kolski

Mobile Robots: Perception & Navigation
http://dx.doi.org/10.5772/36
Edited by Sascha Kolski

Notice

Statements and opinions expressed in the chapters are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published chapters. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

First published in Croatia, 2007 by INTECH d.o.o.
eBook (PDF) Published by IN TECH d.o.o.
Place and year of publication of eBook (PDF): Rijeka, 2019.
IntechOpen is the global imprint of IN TECH d.o.o.
Printed in Croatia

Legal deposit, Croatia: National and University Library in Zagreb

Additional hard and PDF copies can be obtained from orders@intechopen.com

Mobile Robots: Perception & Navigation
Edited by Sascha Kolski

p. cm.

Print ISBN 3-86611-283-1

eBook (PDF) ISBN 978-953-51-5804-2

Preface

Mobile Robotics is an active research area where researchers from all over the world find new technologies to improve mobile robots intelligence and areas of application. Today robots navigate autonomously in office environments as well as outdoors. They show their ability to beside mechanical and electronic barriers in building mobile platforms, perceiving the environment and deciding on how to act in a given situation are crucial problems. In this book we focused on these two areas of mobile robotics, Perception and Navigation.

Perception includes all means of collecting information about the robot itself and it's environment. To make robots move in their surrounding and interact with their environment in a reasonable way, it is crucial to understand the actual situation the robot faces.

Robots use sensors to measure properties of the environment and interpret these measurements to gather knowledge needed for save interaction. Sensors used in the work described in the articles in this book include computer vision, range finders, sonars and tactile sensors and the way those sensors can be used to allow the robot the perception of it's environment and enabling it to safely accomplishing it's task. There is also a number of contributions that show how measurements from different sensors can be combined to gather more reliable and accurate information as a single sensor could provide, this is especially efficient when sensors are complementary on their strengths and weaknesses.

As for many robot tasks mobility is an important issue, robots have to navigate their environments in a safe and reasonable way. Navigation describes, in the field of mobile robotics, techniques that allow a robot to use information it has gathered about the environment to reach goals that are given a priory or derived from a higher level task description in an effective and efficient way.

The main question of navigation is how to get from where we are to where we want to be. Researchers work on that question since the early days of mobile robotics and have developed many solutions to the problem considering different robot environments. Those include indoor environments, as well is in much larger scale outdoor environments and under water navigation.

Beside the question of global navigation, how to get from A to B navigation in mobile robotics has local aspects. Depending on the architecture of a mobile robot (differential drive, car like, submarine, plain, etc.) the robot's possible actions are constrained not only by the ro-

bots' environment but by its dynamics. Robot motion planning takes these dynamics into account to choose feasible actions and thus ensure a safe motion.

This book gives a wide overview over different navigation techniques describing both navigation techniques dealing with local and control aspects of navigation as well es those handling global navigation aspects of a single robot and even for a group of robots.

As not only this book shows, mobile robotics is a living and exciting field of research combining many different ideas and approaches to build mechatronical systems able to interact with their environment.

Editor
Sascha Kolski

Contents

Perception

Navigation

1

Robot Egomotion from the Deformation of Active Contours

Guillem ALENYA and Carme TORRAS
Institut de Robòtica i Informàtica Industrial (CSIC-UPC) Barcelona, Catalonia, Spain

1.Introduction

Traditional sources of information for image-based computer vision algorithms have been points, lines, corners, and recently SIFT features (Lowe, 2004), which seem to represent at present the state of the art in feature definition. Alternatively, the present work explores the possibility of using tracked contours as informative features, especially in applications not requiring high precision as it is the case of robot navigation.

In the past two decades, several approaches have been proposed to solve the robot positioning problem. These can be classified into two general groups (Borenstein et al., 1997): absolute and relative positioning. Absolute positioning methods estimate the robot position and orientation in the workspace by detecting some landmarks in the robot environment. Two subgroups can be further distinguished depending on whether they use natural landmarks (Betke and Gurvits, 1997; Sim and Dudek, 2001) or artificial ones (Jang et al., 2002; Scharstein and Briggs, 2001). Approaches based on natural landmarks exploit distinctive features already present in the environment. Conversely, artificial landmarks are placed at known locations in the workspace with the sole purpose of enabling robot navigation. This is expensive in terms of both presetting of the environment and sensor resolution.

Relative positioning methods, on the other hand, compute the robot position and orientation from an initial configuration, and, consequently, are often referred to as motion estimation methods. A further distinction can also be established here between incremental and non-incremental approaches. Among the former are those based on odometry and inertial sensing, whose main shortcoming is that errors are cumulative.

Here we present a motion estimation method that relies on natural landmarks. It is not incremental and, therefore, doesn't suffer from the cumulative error drawback. It uses the images provided by a single camera. It is well known that in the absence of any supplementary information, translations of a monocular vision system can be recovered up to a scale factor. The camera model is assumed to be weak-perspective. The assumed viewing conditions in this model are, first, that the object points are near the projection ray (can be accomplished with a camera having a small field of view), and second, that the depth variation of the viewed object is small compared to its distance to the camera This camera model has been widely used before (Koenderink and van Doorn, 1991; Shapiro et al., 1995; Brandt, 2005).

Active contours are a usual tool for image segmentation in medical image analysis. The ability of fastly tracking active contours was developed by Blake (Blake and Isard, 1998) in the framework of dynamics learning and deformable contours. Originally, the tracker was implemented with a Kalman filter and the active contour was parameterized as a b-spline in the image plane. Considering non-deformable objects, Martinez (Martínez, 2000) demonstrated that contours could be suitable to recover robot ego-motion qualitatively, as required in the case of a walking robot (Martínez and Torras, 2001). In these works, initialization of the b-spline is manually performed by an operator. When corners are present, the use of a corner detector (Harris and Stephens, 1988) improves the initial adjustment. Automatic initialization techniques have been proposed (Cham and Cipolla, 1999) and tested with good results. Since we are assuming weak perspective, only affine deformations of the initial contour will be allowed by the tracker and, therefore, the initialization process is importantas it determines the family of affine shapes that the contour will be allowed to adjust to.

We are interested in assessing the accuracy of the motion recovery algorithm by analyzing the estimation errors and associated uncertainties computed while the camera moves. We aim to determine which motions are better sensed and which situations are more favorable to minimize estimation errors. Using Monte Carlo simulations, we will be able to assign an uncertainty value to each estimated motion, obtaining also a quality factor. Moreover, a real experiment with a robotized fork-lift will be presented, where we compare our results with the motion measured by a positioning laser. Later, we will show how the information from an inertial sensor can complement the visual information within the tracking algorithm. An experiment with a four-person transport robot illustrates the obtained results.

2. Mapping contour deformations to camera motions

2.1. Parameterisation of contour deformation

Under weak-perspective conditions (i.e., when the depth variation of the viewed object is small compared to its distance to the camera), every 3D motion of the object projects as an affine deformation in the image plane.

The affinity relating two views is usually computed from a set of point matches (Koenderink and van Doorn, 1991; Shapiro et al., 1995). Unfortunately, point matching can be computationally very costly, it being still one of the key bottlenecks in computer vision. In this work an active contour (Blake and Isard, 1998) fitted to a target object is used instead. The contour, coded as a b-spline (Foley et al., 1996), deforms between views leading to changes in the location of the control points.

It has been formerly demonstrated (Blake and Isard, 1998; Martínez and Torras, 2001, 2003) that the difference in terms of control points $\mathbf{Q'-Q}$ that quantifies the deformation of the contour can be written as a linear combination of six vectors. Using matrix notation

$$\mathbf{Q'-Q=WS} \tag{1}$$

where

$$\mathbf{W} = \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} Q^x \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ Q^y \end{bmatrix}, \begin{bmatrix} 0 \\ Q^x \end{bmatrix}, \begin{bmatrix} Q^y \\ 0 \end{bmatrix} \right)$$

and $\mathbf{S}$ is a vector with the six coefficients of the linear combination. This so-called shape

vector

$$\mathbf{S} = [t_x, t_y, M_{1,1} - 1, M_{2,2} - 1, M_{2,1}, M_{1,2}] \tag{3}$$

encodes the affinity between two views $\mathbf{d'}(s)$ and $\mathbf{d}(s)$ of the planar contour:

$$\mathbf{d}'(s) = \mathbf{M}\mathbf{d}(s) + \mathbf{t}, \tag{4}$$

where $\mathbf{M} = [M_{i,j}]$ and $\mathbf{t} = (t_x, t_y)$ are, respectively, the matrix and vector defining the affinity in the plane.

Different deformation subspaces correspond to constrained robot motions. In the case of a planar robot, with 3 degrees of freedom, the motion space is parametrized with two translations (T_x, T_z) and one rotation (θ_y). Obviously, the remaining component motions are not possible with this kind of robot. Forcing these constraints in the equations of the affine deformation of the contour, a new shape space can be deduced. This corresponds to a shape matrix having also three dimensions.

However, for this to be so, the target object should be centered in the image. Clearly, the projection of a vertically non-centered object when the camera moves towards will translate also vertically in the image plane. Consequently, the family of affine shapes that the contour is allowed to adjust to should include vertical displacements. The resulting shape matrix can then be expressed as

$$\mathbf{W} = \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} Q_0^x \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ Q_0^y \end{bmatrix} \right), \tag{5}$$

and the shape vector as

$$\mathbf{S} = (t_x, t_y, M_{11} - 1, M_{22} - 1)^T. \tag{6}$$

2.2. Recovery of 3Dmotion

The contour is tracked along the image sequence with a Kalman filter (Blake and Isard, 1998) and, for each frame, the shape vector and its associated covariance matrix are updated. The affinity coded by the shape vector relates to the 3D camera motion in the following way (Blake and Isard, 1998; Martínez and Torras, 2001, 2003):

$$\mathbf{M} = \frac{Z_0}{Z_0 R_{33} + T_z} \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, \tag{7}$$

$$\mathbf{t} = \frac{1}{Z_0 R_{33} + T_z} \begin{bmatrix} Z_0 R_{13} + T_x \\ Z_0 R_{23} + T_y \end{bmatrix}, \tag{8}$$

where R_{ij} are the elements of the 3D rotation matrix $\mathbf{R}$, T_i are the elements of the 3D translation vector $\mathbf{T}$, and Z_0 is the distance from the viewed object to the camera in the initial position.

We will see next how the 3D rotation and translation are obtained from the $\mathbf{M} = [M_{i,j}]$ and $\mathbf{t} = (t_x, t_y)$ defining the affinity. Representing the rotation matrix in Euler angles form,

$$\mathbf{R} = \mathbf{R}_z(\phi)\mathbf{R}_x(\theta)\mathbf{R}_z(\psi), \tag{9}$$

equation (7) can be rewritten as

$$\begin{aligned} \mathbf{M} &= \frac{Z_0}{Z_0 R_{33} + T_z} \mathbf{R}_z|_2(\phi)\mathbf{R}_x|_2(\theta)\mathbf{R}_z|_2(\psi) = \\ &= \frac{Z_0}{Z_0 R_{33} + T_z} \mathbf{R}_z|_2(\phi) \begin{bmatrix} 1 & 0 \\ 0 & cos\theta \end{bmatrix} \mathbf{R}_z|_2(\psi) \end{aligned}$$

where R|2 denotes de 2 X 2 submatrix of **R**. Then,

$$\mathbf{M}\mathbf{M}^T = \mathbf{R_z}|_2(\phi) \begin{bmatrix} L & 0 \\ 0 & L\cos^2\theta \end{bmatrix} \mathbf{R_z}|_2{}^{-1}(\phi) \tag{10}$$

where

$$L = \left(\frac{Z_0}{Z_0 R_{33} + T_z}\right)^2.$$

This last equation shows that θ can be calculated from the eigenvalues of the matrix $\mathbf{M}\mathbf{M}^T$, which we will name (λ_1, λ_2):

$$\cos\theta = \sqrt{\frac{\lambda_2}{\lambda_1}}, \tag{11}$$

where λ_1 is the largest eigenvalue. The angle ϕ can be extracted from the eigenvectors of $\mathbf{M}\mathbf{M}^T$; the eigenvector $\mathbf{v_1}$ with larger value corresponds to the first column of $\mathbf{R_z}|_2(\phi)$:

$$\mathbf{v_1} = \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}. \tag{12}$$

Isolating $\mathbf{Rz|2}(\psi)$from equation (10),

$$\mathbf{R_z}|_2(\psi) = (R_{33} + \frac{T_z}{Z_0}) \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\cos\theta} \end{bmatrix} \mathbf{R_z}|_2(-\phi)\mathbf{M}, \tag{13}$$

and observing, in equation (10), that

$$R_{33} + \frac{T_z}{Z_0} = \frac{1}{\sqrt{\lambda_1}},$$

$\sin\psi$ can be found, and then ψ.

Once the angles θ, ϕ, ψ are known, the rotation matrix **R** can be derived from equation (9).

The scaled translation in direction Z is calculated as

$$\frac{T_z}{Z_0} = \frac{1}{\sqrt{\lambda_1}} - R_{33}. \tag{14}$$

The rest of components of the 3D translation can be derived from **t**and **R**using equation (8):

$$\frac{T_x}{Z_0} = \frac{t_x}{\sqrt{\lambda_1}} - R_{13}, \tag{15}$$

$$\frac{T_y}{Z_0} = \frac{t_y}{\sqrt{\lambda_1}} - R_{23}. \tag{16}$$

Using the equations above, the deformation of the contour parameterized as a planar affinity permits deriving the camera motion in 3D space. Note that, to simplify the derivation, the reference system has been assumed to be centered on the object.

3. Precision of motion recovery

3.1. Rotation representation and systematic error

As shown in equation (9), rotation is codified as a sequence of Euler angles $\mathbf{R} = \mathbf{R_z}(\phi)\,\mathbf{R_x}(\theta)\,\mathbf{R_z}(\psi)$. Typically, this representation has the problem of the Gimbal lock: when two axes are aligned there is a problem of indetermination.

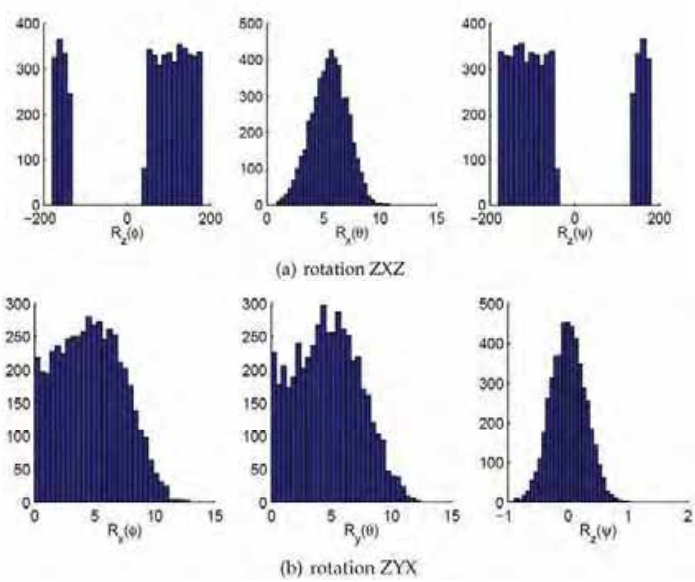


Fig. 1. Histogram of the computed rotation values for 5000 trials adding Gaussian noise with σ = 0.5pixels to the contour control points.(a) In the ZXZ representation, small variations of the pose correspond to discontinuous values in the rotation components $R_z(\phi)$ and $R_z(\psi)$. (b) In contrast, the same rotations in the ZYX representation yield continuous values.

This happens when the second rotation $\mathbf{Rx}(\theta)$ is near the null rotation. As a result, small variations in the camera pose do not lead to continuous values in the rotation representation (see $\mathbf{R_z}(\phi)$ and $\mathbf{R_z}(\psi)$ in Fig. 1(a)). Using this representation, means and covariances cannot be coherently computed. In our system this could happen frequently, for example at the beginning of any motion, or when the robot is moving towards the target object with small rotations.

We propose to change the representation to a *roll-pitch-yaw* codification. It is frequently used in the navigation field, it being also called *heading-attitude-bank* (Sciavicco and Siciliano, 2000). We use the form

$$\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) = \begin{bmatrix} c\psi c\theta & s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi - c\psi s\theta c\phi \\ -s\psi c\theta & c\psi c\phi - s\psi s\theta s\phi & c\psi s\phi + s\psi s\theta c\phi \\ s\theta & -c\theta s\phi & c\theta c\phi \end{bmatrix}, \tag{17}$$

where $s\psi$ and $c\psi$ denote the sinus and cosinus of ψ, respectively. The inverse solution is.

$$\phi = atan2(R_{32}, R_{33}) \tag{18}$$

$$\theta = asin(-R_{31}) \tag{19}$$

$$\psi = atan2(R_{21}, R_{11}). \tag{20}$$

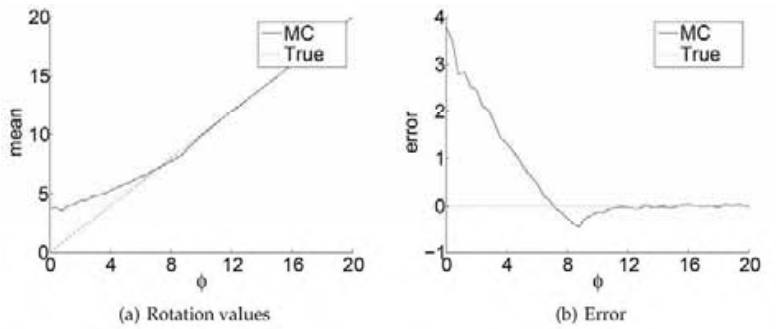


Fig. 2. Systematic error in the R_x component. Continuous line for values obtained with Monte Carlo simulation and dotted line for true values. The same is applicable to the R_y component.

Typically, in order to represent all the rotation space the elemental rotations should be restricted to lie in the $[0..2\pi]rad$ range for ψ and ϕ, and in $[0..\pi]rad$ for θ.
Indeed, tracking a planar object by rotating the camera about X or Y further than $\pi/2rad$ has no sense, as in such position all control points lie on a single line and the shape information is lost. Also, due to the *Necker reversal* ambiguity, it is not possible to determine the sign of the rotations about these axes. Consequently, without loss of generality, we can restrict the range of the rotations $\mathbf{R_x}(\phi)$and $\mathbf{R_y}(\theta)$ to lie in the range $[0..\frac{\pi}{2})$ *rad* and let $\mathbf{Rz}(\psi)$ in $[0..2\pi]$ *rad*. With this representation, the Gimbal lock has been displaced to $\cos(\theta) = 0$, but $\theta = \pi/2$ is out of the range in our application.
With the above-mentioned sign elimination, a bias is introduced for small $\mathbf{R_x}(\phi)$and $\mathbf{R_y}(\theta)$ rotations. In the presence of noise and when the performed camera rotation is small, negative rotations will be estimated positive. Thus, the computation of a mean pose, as presented in the next section, will be biased. Figure 2(a) plots the results of an experiment where the camera performs a rotation from 0 to 20°about the X axis of a coordinate system located at the target. Clearly, the values $\mathbf{R_x}(\phi)$ computed by the Monte Carlo simulation are closer to the true ones as the amount of rotation increases. Figure 2(b) summarizes the resulting errors. This permits evaluating the amount of systematic error introduced by the rotation representation.
In sum, the proposed rotation space is significantly reduced, but we have shown that it is enough to represent all possible real situations. Also, with this representation the Gimbal lock is avoided in the range of all possible data. As can be seen in Figure 1(b), small variations in the pose lead to small variations in the rotation components. Consequently, means and covariances can be coherently computed with Monte Carlo estimation. A bias is introduced when small rotations about X and Y are performed, which disappears when the rotations become more significant. This is not a shortcoming in real applications.

3.2. Assessing precision through Monte Carlo simulation

The synthetic experiments are designed as follows. A set of control points on the 3D planar object is chosen defining the b-spline parameterisation of its contour.

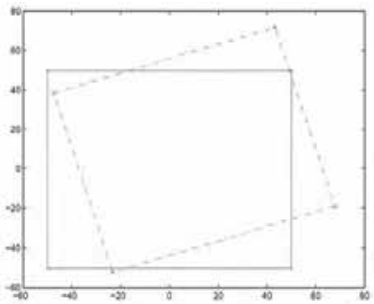


Fig. 3. Original contour projection (continuous line) and contour projection after motion (dotted line)for the experiments detailed in the text.

The control points of the b-spline are projected using a perspective camera model yielding the control points in the image plane (Fig. 3). Although the projection is performed with a complete perspective camera model, the recovery algorithm assumes a weak-perspective camera. Therefore, the perspective effects show up in the projected points (like in a real situation) but the affinity is not able to model them (only approximates the set of points as well as possible), so perspective effects are modelled as affine deformations introducing some error in the recovered motion. For these experiments the camera is placed at 5000mm and the focal distance is set to 50mm.

Several different motions are applied to the camera depending on the experiment. Once the camera is moved, Gaussian noise with zero mean and $\sigma = 0.5$ is added to the new projected control points to simulate camera acquisition noise. We use the algorithm presented in Section 2.2 to obtain an estimate of the 3D pose for each perturbed contour in the Monte Carlo simulation. 5000 perturbed samples are taken. Next, the statistics are calculated from the obtained set of pose estimations.

3.2.1. Precision in the recovery of a single translation or rotation

Here we would like to determine experimentally the performance (mean error and uncertainty) of the pose recovery algorithm for each camera component motion, that is, translations T_x, T_y and T_z, and rotations R_x, R_y and R_z. The first two experiments involve lateral camera translations parallel to the X or Y axes. With the chosen camera configuration, the lateral translation of the camera up to 250mm takes the projection of the target from the image center to the image bound. The errors in the estimations are presented in Figure 4(a) and 4(c), and as expected are the same for both translations. Observe that while the camera is moving away from the initial position, the error in the recovered translation increases, as well as the corresponding uncertainty. The explanation is that the weak-perspective assumptions are less satisfied when the target is not centered. However, the maximum error in the mean is about 0.2%, and the worst standard deviation is 0.6%, therefore lateral translations are quite correctly recovered. As shown in (Alberich-Carramiñana et al., 2006), the sign of the error depends on the target shape and the orientation of the axis of rotation.

The third experiment involves a translation along the optical axis Z. From the initial distance $Z_0 =$ 5000 the camera is translated to $Z = 1500$, that is a translation of $-3500mm$. The errors and the confidence values are shown in Figure 4(e). As the camera approaches the target, the mean error and its standard deviation decrease. This is in accordance with how the projection works[1] As expected, the precision of the translation estimates is worse for this axis than for X and Y.
The next two experiments involve rotations of the camera about the target. In the first, the camera is rotated about the X and Y axes of a coordinate system located at the target. Figure 4(b) and 4(d) show the results. As expected, the obtained results are similar for these two experiments. We use the alternative rotation representation presented in Section 3.1, so the values R_x and R_y are restricted. As detailed there, all recovered rotations are estimated in the same side of the null rotation, thus introducing a bias. This is not a limitation in practice since, as will be shown in experiments with real images, the noise present in the tracking step masks these small rotations, and the algorithm is unable to distinguish rotations of less than about 10° anyway.
The last experiment in this section involves rotations of the camera about Z. As expected, the computed errors (Fig. 4(f)) show that this component is accurately recovered, as the errors in the mean are negligible and the corresponding standard deviation keeps also close to zero.

4. Performance in real experiments

The mobile robot used in this experiment is a Still EGV-10 modified forklift (see Fig. 5). This is a manually-guided vehicle with aids in the traction. To robotize it, a motor was added in the steering axis with all needed electronics. The practical experience was carried out in a warehouse of the brewer company DAMM in El Prat del Llobregat, Barcelona. During the experience, the robot was guided manually. A logger software recorded the following simultaneous signals: the position obtained by dynamic triangulation using a laser-based goniometer, the captured reflexes, and the odometry signals provided by the encoders. At the same frequency, a synchronism signal was sent to the camera and a frame was captured. A log file was created with the obtained information. This file permitted multiple processing to extract the results for the performance assessment and comparison of different estimation techniques (Alenyà et al., 2005). Although this experiment was designed in two steps: data collection and data analysis, the current implementations of both algorithms run in real time, that is, 20 *fps* for the camera subsystem and 8 *Hz* for the laser subsystem.
In the presented experiment the set of data to be analyzed by the vision subsystem consists of 200 frames. An active contour was initialized manually on an information board appearing in the first frame of the chosen sequence (Fig. 6). The tracking algorithm finds the most suitable affine deformation of the defined contour that fits the target in the next frame, yielding an estimated affine deformation (Blake and Isard, 1998). Generally, this is expressed in terms of a shape vector (6), from which the corresponding Euclidean 3D transformation is derived: a translation vector (equations 14-16)and a rotation matrix (equations 9-13). Note that, in this experiment, as the robot moves on a plane, the reduced 4-dimensionalshape vector (6) was used.

[1]The resolution in millimeters corresponding to a pixel depends on the distance of the object to the camera. When the target is near the camera, small variations in depth are easily sensed. Otherwise, when the target is far from the camera, larger motions are required to be sensed by the camera.

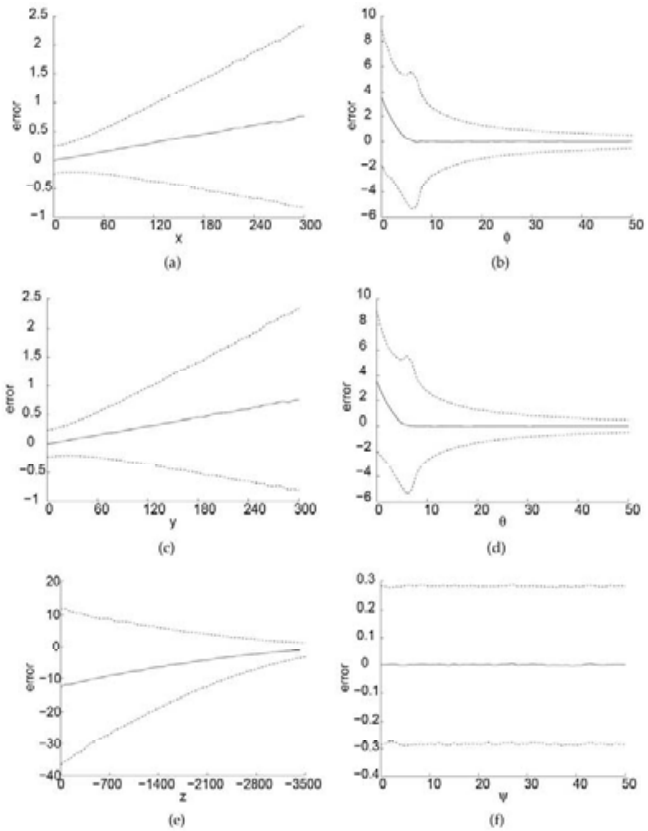
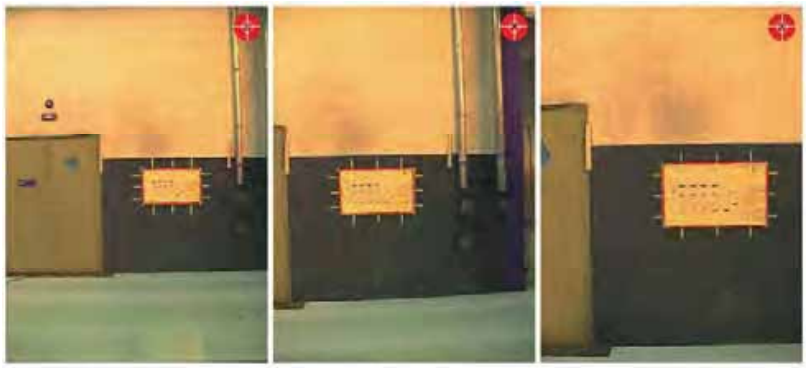


Fig. 4. Mean error (solid lines) and 2σ deviation (dashed lines) for pure motions along and about the 6 coordinate axes of a camera placed at $5000mm$ and focal length $50mm$. Errors in T_x and T_y translations are equivalent, small while centered and increasing while uncentered, and translation is worst recovered for T_z (although it gets better while approximating). Errors for small R_x and R_y rotations are large, as contour deformation in the image is small, while for large transformations errors are less significant. The error in R_z rotations is negligible.

The tracking process produces a new deformation for each new frame, from which 3D motion parameters are obtained. If the initial distance Z_0 to the target object can be estimated, a metric reconstruction of motion can be accomplished. In the present experiment, the value of the initial depth was estimated with the laser sensor, as the target (the information board) was placed in the same wall as some catadioptric marks, yielding a value of $7.7m$. The performed motion was a translation of approximately $3.5m$ along the heading direction of the robot perturbed by small turnings.

Fig. 5. Still EGV-10robotized forklift used in a warehouse for realexperimentation. Odometry, laser positioningandmonocular vision data were recollected.



(a) $frame_{first}$ (b) $frame_{middle}$ (c) $frame_{last}$

Fig. 6. Real experiment to compute a large translation while slightly oscillating. An active contour is fitted to an information board and used as target to compute egomotion.

The computed T_x, T_y and T_z values can be seen in Fig. 7(a). Observe that, although the t_y component is included in the shape vector, the recovered T_y motion stays correctly at zero. Placing the computed values for the X and Z translations in correspondence in the actual motion plane, the robot trajectory can be reconstructed (Fig. 7(b)).

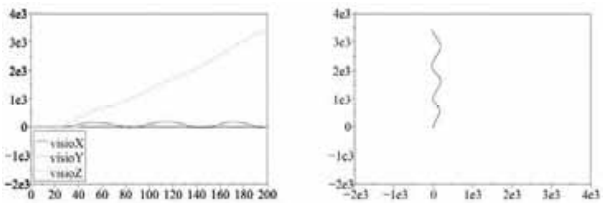


Fig. 7. (a)Evolution of the recovered T_x, T_y and T_z components (in millimeters). (b) Computed trajectory (in millimeters)in the XZ plane.

Extrinsic parameters from the laser subsystem and the vision subsystem are needed to be able to compare the obtained results. They provide the relative position between both acquisition reference frames, which is used to put in correspondence both position estimations. Two catadrioptic landmarks used by the laser were placed in the same plane as a natural landmark used by the vision tracker. A rough estimation of the needed calibration parameters (d_x and d_y) was obtained with measures taken from controlled motion of the robot towards this plane, yielding the values of 30 *mm* and 235 *mm*, respectively. To perform the reference frame transformation the following equations were used:

$$x_{cam} = x - (sin(\psi) * d_x) + (cos(\psi) * d_y),$$
$$y_{cam} = y - (sin(\psi) * d_x) + (cos(\psi) * d_y).$$

While laser measurements are *global*, the vision system ones are relative to the initial position taken as reference (Martíinez and Torras, 2001). To compare both estimations, laser measurements have been transformed to express measurement increments.

The compared position estimations are shown in Fig. 8 (a), where the vision estimation is subtracted from the laser estimation to obtain the difference for each time step.

Congruent with previous results (see Sec 3.2) the computed difference in the Z direction is more noisy, as estimations from vision for translations in such direction are more ill conditioned than for the X or Y directions. In all, it is remarkable that the computed difference is only about 3%.

The computed differences in X are less noisy, but follow the robot motion. Observe that, for larger heading motions, the difference between both estimations is also larger. This has been explained before and it is caused by the uncentered position of the object projection, which violates one of the weak-perspective assumptions.

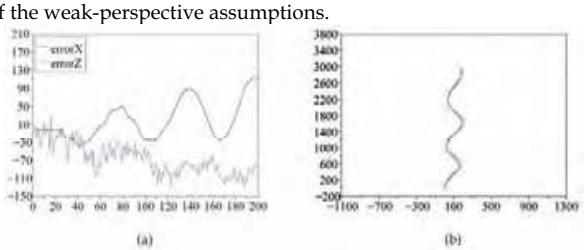


Fig. 8. Comparison between the results obtained with the visual egomotion recovery algorithm and laser positioning estimation. (a) Difference in millimeters between translation estimates provided by the laser and the vision subsystems for each frame. (b) Trajectories in millimeters in the XZ plane. The black line corresponds to the laser trajectory, the blue dashed line to the laser-estimated camera trajectory, and the green dotted one to the vision-computed camera trajectory.

Finally, to compare graphically both methods, the obtained translations are represented in the *XZ* plane (Fig. 8(b)).
This experiment shows that motion estimation provided by the proposed algorithm has a reasonably precision, enough for robot navigation. To be able to compare both estimations it has been necessary to provide to the vision algorithm the initial distance to the target object (Z_0) and the calibration parameters of the camera (*f*). Obviously, in absence of this information the recovered poses are scaled. With scaled poses it is still possible to obtain some useful information for robot navigation, for example the time to contact (Martínez and Torras, 2001). The camera internal parameters can be estimated through a previous calibration process, or online with autocalibration methods. We are currently investigating the possibility of estimating initial distance to the object with depth-from-defocus and depth-from-zoom algorithms.

5. Using inertial information to improve tracking

We give now a more detailed description of some internals of the tracking algorithm. The objective of tracking is to follow an object contour along a sequence of images. Due to its representation as a b-spline, the contour is divided naturally into sections, each one between two consecutive nodes. For the tracking, some interest points are defined equidistantly along each contour section. Passing through each point and normal to the contour, a line segment is defined. The search for edge elements (called "edgels") is performed only for the pixels under these normal segments, and the result is the Kalman measurement step. This allows the system to be quick, since only local image processing is carried out, avoiding the use of high-cost image segmentation algorithms.
Once edge elements along all search segments are located, the Kalman filter estimates the resulting shape vector, which is always an affine deformation of the initial contour.
The length of the search segments is determined by the covariance estimated in the preceding frame by the Kalman filter. This is done by projecting the covariance matrix into the line normal to the contour at the given point. If tracking is finding good affine transformations that explain changes in the image, the covariance decreases and the search segments shrank. On the one hand, this is a good strategy as features are searched more locally and noise in image affects less the system. But, on the other hand, this solution is not the best for tracking large changes in image projection. Thus, in this section we will show how to use inertial information to adapt the length of the different segments at each iteration (Alenyà etal., 2004).

5.1. Scaling covariance according to inertial data

Large changes in contour projection can be produced by quick camera motions. As mentioned above, a weak-perspective model is used for camera modeling. To fit the model, the camera field-of-view has to be narrow. In such a situation, distant objects may produce important changes in the image also in the case of small camera motions.
For each search segment normal to the contour, the scale factor is computed as

$$\mathbf{E} = \sqrt{\mathbf{N}^T (\mathbf{H}\mathbf{P}\mathbf{H}^T)\,\mathbf{N}} \tag{21}$$

where **N** are the normal line coordinates, **H** is the measurement vector and **P** is the 6 x 6 top corner of the covariance matrix. Detailed information can be found in (Blake and Isard, 1998).
Note that, as covariance is changing at every frame, the search scale has to be recalculated also for each frame. It is also worth noting that this technique produces different search ranges depending on the orientation of the normal, taking into account the directional

estimation of covariance of the Kalman filter.
In what follows, we explain how inertial information is used to adapt the search ranges locally on the contour by taking into account the measured dynamics. Consider a 3 d.o.f. inertial sensor providing coordinates (x, y, θ). To avoid having to perform a coordinate transformation between the sensor and the camera, the sensor is placed below the camera with their reference frames aligned. In this way, the X and Y coordinates of the inertial sensor map to the Z and X camera coordinates respectively, and rotations take place about the same axis. Sensed motion can be expressed then as a translation

$$\mathbf{T} = \begin{bmatrix} v_x \\ 0 \\ v_z \end{bmatrix}, \tag{22}$$

and a rotation

$$\mathbf{R} = \begin{bmatrix} \cos v_\theta & 0 & -\sin v_\theta \\ 0 & 1 & 0 \\ \sin v_\theta & 0 & \cos v_\theta \end{bmatrix} \tag{23}$$

Combining equations (7, 8) with equations (22, 23), sensed data can be expressed in shape space as

$$M_{11} = \frac{Z_0}{Z_0 R_{33} + T_z} R_{11} = \frac{Z_0}{Z_0 \cos v_\theta + v_z} \cos v_\theta \tag{24}$$

$$M_{21} = \frac{Z_0}{Z_0 R_{33} + T_z} R_{21} = 0$$

$$M_{12} = \frac{Z_0}{Z_0 R_{33} + T_z} R_{12} = 0$$

$$M_{22} = \frac{Z_0}{Z_0 R_{33} + T_z} R_{22} = \frac{Z_0}{Z_0 \cos v_\theta + v_z} \tag{25}$$

$$t_1 = \frac{1}{Z_0 R_{33} + T_z} (Z_0 R_{13} + T_x) = \frac{1}{Z_0 \cos v_\theta + v_z} (-Z_0 \sin v_\theta + v_x) \tag{26}$$

$$t_2 = \frac{1}{Z_0 R_{33} + T_z} (Z_0 R_{23} + T_y) = 0$$

As the objective is to scale covariance, denominators can be eliminated in equations (24 -26). These equations can be rewritten in shape vector form as

$$\mathbf{S} = \begin{matrix} (t_1 \;\; 0 \;\; M_{11} - 1 \;\; M_{22} - 1 \;\; 0 \;\; 0) = \\ (-Z_0 \sin v_\theta + v_x \;\; 0 \;\; -v_z \;\; Z_0 (1 - \cos v_\theta) - v_z \;\; 0 \;\; 0) \end{matrix}$$

For small rotational velocities, $\sin v_\theta$ can be approximated by v_θ and, thus,

$$\mathbf{S} = (-Z_0 v_\theta + v_x \;\; 0 \;\; -v_z \;\; Z_0 v_\theta^2 / 2 - v_z \;\; 0 \;\; 0) \tag{27}$$

The inertial sensor gives the X direction data in the range $[v_{xmin} .. v_{xmax}]$. To simplify the notation, let us consider a symmetric sensor, $|v_{xmin}| = |v_{xmax}|$. Sensor readings can be rescaled to provide values in the range $[v_{xmin} .. v_{xmax}]$. A value v_x provided by the inertial sensor can be rescaled using

$$\overline{v_x} = |v_x| \frac{\overline{v}_{xmax} - \overline{v}_{xmin}}{v_{xmax}} + \overline{v}_{xmin} \tag{28}$$

Following the same reasoning, shape vector parameters can be rescaled. For the first component we have

$$t_{1max} = Z_0 v_{\theta max} + v_{xmax} \tag{29}$$

and the expression

$$\overline{t_1} = |t_1| \frac{\overline{t}_{1max} - \overline{t}_{1min}}{t_{1max}} + \overline{t}_{1min} = |t_1| f_{t_1} + \overline{t}_{1min} \tag{30}$$

Inertial information can be added now by scaling the current covariance sub-matrix by a matrix representing the scaled inertial data as follows

$$\mathbf{E} = \sqrt{\mathbf{N}^T (\mathbf{HVPV}^T \mathbf{H}^T)\, \mathbf{N}} \tag{31}$$

where **V**is the scaled measurement matrix for the inertial sensing system defined as

$$\mathbf{V} = \begin{bmatrix} t_1 & .. & & & 0 \\ & t_2 & & & \\ . & & M_{11}-1 & & . \\ . & & & M_{22}-1 & . \\ & & & & M_{12} \\ 0 & .. & & & M_{21} \end{bmatrix} \begin{bmatrix} f_{t_1} \\ f_{t_2} \\ f_{M_{11}} \\ f_{M_{22}} \\ f_{M_{12}} \\ f_{M_{21}} \end{bmatrix} + \begin{bmatrix} \overline{t}_{1min} \\ \overline{t}_{2min} \\ \overline{M}_{11min} \\ \overline{M}_{22min} \\ \overline{M}_{12min} \\ \overline{M}_{21min} \end{bmatrix} \tag{32}$$

For testing purposes, all minimum and maximum values have been set to 1 and 2, respectively.

Fig. 9. Robucab mobile robot platform transporting four people.

5.2.Experiments enhancing vision with inertial sensing

For this experimentation, we use a Robu Cab Mobile Robot from Robosoft. As can be seen in Fig. 9, it is a relatively big mobile vehicle with capacity for up to four people. It can be used in two modes: car-like navigation and bi-directional driving.

For simplicity of the control system, the car-like driving option is used, but better results should be obtained under bi-directional driving mode as the maximum turning angle would increase. In this vehicle we mount a monocular vision system with the described 6 d.o.f. tracking system. A Gyrostar inertial sensor, from Murata, is used to measure rotations about the Y axis. To measure X and Z linear accelerations, an ADXL dual accelerometer from Analog Devices is used. All these sensors are connected to a dedicated board with an AVR processor used to make A/D conversions, PWM decoding and time integration. It has also a thermometer for thermal data correction. This 'intelligent' sensor provides not only changes in velocity, but also mean velocity and position. Drift, typical in this kind of computations, is reset periodically with the information obtained by fusion of the other sensors. This board shares memory with a MPC555 board, which is connected through a CAN bus to the control and vision processing PC. All the system runs under a real-time Linux kernel in a Pentium 233 MHz industrial box. A novel approach to distributed programming (Pomiers, 2002) has been used to program robot control as well as for the intercommunication of controll and vision processes, taking advantage of the real time operating system.

Although it might look as if the robot moves on a plane, its motions are in 6 parameter space, mainly due to floor rugosity and vehicle dampers, and therefore the whole 6D shape vector is used.

In this experiment the robot is in autonomous driving mode, following a filoguided path. In this way, the trajectory can be easily repeated, thus allowing us to perform several experiments with very similar conditions. The path followed consists of a straight line segment, a curve and another straight line.

First, the algorithm without inertial information is used. On the first straight segment, the contour is well followed, but as can be seen in Figure 10(a), when turning takes place and the contour moves quicker in the image plane, it loses the real object and the covariance trace increases.

Second, the algorithm including inertial information in the tracker is used. In this experiment, tracking does not lose the target and finishes the sequence giving good recovered pose values. As can be seen in the covariance representation in Figure 10(b), covariance increases at the beginning of the turning, but decreases quickly, showing that tracking has fixed the target despite its quick translation across the image.

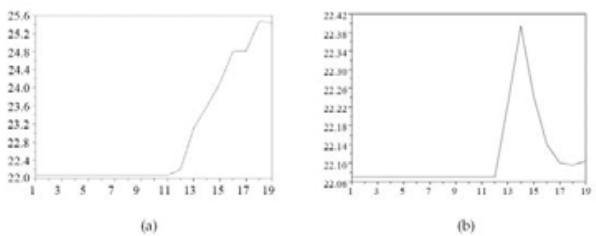


Figure 10: Covariance trace resulting from tracking without using inertial information (a) and using it (b).

6.Conclusions and future work

A method for estimating mobile robot egomotion has been presented, which relies on tracking contours in real-time images acquired with a monocular vision system. The deformation of the contour due to camera motion is codified as a 6-dimensional affine shape vector, and the algorithm to recover 3D motion information is presented.

The precision of the algorithm is analyzed through Monte Carlo simulations. The results obtained are congruent with intuition. Lateral camera translations T_x and T_y produce greater changes in pixels, so they are better recovered than the translation T_z along the optical axis. Rotations R_z about the projection axis cause large changes in the image, and are better recovered than the other two pure rotations, R_x and R_y. Estimated variances differ largely for the various motions. The largest errors and variances occur when the contour projection is un centered in the image, as weak-perspective assumptions are violated. If the distance to the target is small, more precision is attained, but perspective effects appear. Small rotations out of the plane are badly estimated, but as the rotation increases the error and the variance diminish. Rotations in the plane are correctly recovered with small variance.

A real experiment performed in a brewer warehouse has been used to validate the motion estimation algorithm and to compare it with laser positioning. Contrarily to the laser estimation procedure, a natural landmark was used and no previous intervention was needed. A relatively small deviation (about 3%) between vision and laser motion estimations was obtained. This supports vision-based egomotion estimation as a promising alternative in situations with relatively low-precision demands.

Synthetic experiments suggest that the target should be centered in the image to keep the weak-perspective assumptions and attain more precision. Real experiments show that the range of applicability of the proposed algorithm is limited as the contour should be kept within the image along all the sequence. One solution is to switch from one target contour to another when the former disappears from the image. Another solution we will explore in future work is to keep the target into the image with the use of a pan-and-tilt camera. This will allow larger robot motions.

We have also noticed that the size of the target projection in the image should be kept into a rasonable margins to be able to track and deduce valid information. The range of approaching translations in the experiments in the warehouse was 4 -5me-ters. This is also a limitation. We are exploring the use of a zooming camera to maintain the size of the projection onto the image constant. This presents some challenges, as changing the zoom complicates the pan and tilt control. Depending on the initial distance, that we assume unknow, different control gains should be applied.

We have described how inertial information can be expressed in shape space terms. We have used this to improve tracking and to provide more robustness to the Kalman filter used to estimate shape deformation. These two sources of information naturally complement one another, as inertial is suitable for quick motions whereas vision is better suited for slow and large motions. The real experiment presented, using also natural landmarks, illustrates that with the reactivity provided by the inertial information, the tracking algorithm is able to extract motion information in sequences where before itwas not.

In the future we will explore how to take advantadge of the inertial information also in the

measurement step of the Kalman filter, as inertial data can be seen also as another estimation of the performed motion. This is possible because in this paper we have derived the link between 3D motion and shape deformation. We can generalize this to more sensors, fusing their supplied data in shape space.

7. Acknowledgment

This work is partially funded by the EU PACO-PLUS project FP6-2004-IST-4-27657. The authors thank the brewer company DAMM in El Prat de Llobregat, Barcelona, and the robotics company Robosoft in Bidart, France, for their help in performing the experiments.

8. References

D.G. Lowe. Distinctive image features from scale-invariant key points. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

J. Borenstein, H.R. Everett, L. Feng, and D. Wehe. Mobile robot positioning -sensors and techniques. *Journal of Robotic Systems*, 14(4):231–249, 1997.

M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Trans. Robot. Automat.*, 13(2):251–263, 1997.

R. Sim and G. Dudek. Learning environmental features for pose estimation. *Image Vision Comput.*, 17:445–460, 2001.

G. Jang, S. Kim, W. Lee, and I. Kweon. Color landmark-based self-localization for indoor mobile robots. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1037–1042, Washington, May 2002.

D. Scharstein and A. Briggs. Real-time recognition of self-similar landmarks. *Image Vision Comput.*, 19(11):763–772, 2001.

J. Koenderinkand A. J. van Doorn.Affine structure from motion. *J. Opt. Soc. Am. A*, 8 (2):377–385, 1991.

L. S. Shapiro, A. Zisserman, and M. Brady. 3D motion recovery via affine epipolar geometry. *Int. J. Comput. Vision*, 16(2):147–182, 1995.

S.S. Brandt. Conditional solutions for the affine reconstruction ofn-views. *Image Vision Comput.*, 23(7):619–630, July 2005.

A. Blake and M. Isard. *Active contours*. Springer, 1998.

E. Martınez. *Recovery of 3Dstructure andmotion from the deformation of an active contour in a sequence of monocular images*. PhD thesis, Universitat Politecnica de Catalunya, 2000.

E. Martínez and C. Torras. Qualitative vision for the guidance of legged robots in unstructured environments. *Pattern Recognition*, 34:1585–1599, 2001.

C. G. Harris and M. Stephens. Acombined corner edge detector. In *Proc. Alvey Vision Conf.*, pages 189–192, Manchester, Aug. 1988.

T. Cham and R. Cipolla. Automated b-spline curve representation incorporating mdl and error-minimizing control point insertion strategies. *IEEE Trans. Pattern Anal. Machine Intell.*, 21(1), 1999.

J. Foley, A. van Dam, S. Feiner, and F. Hughes. *Computer Graphics. Principles and Practice*.

Addison-Wesley Publishing Company, 1996.

E. Martínez and C. Torras. Contour-based 3dmotion recovery while zooming. *Robotics and Autonomous Systems*, 44:219–227, 2003.

L.Sciavicco and B.Siciliano. *Modeling and Control of Robot Manipulators*. Springer-Verlag, London, 2000.

M. Alberich-Carraminana, G. Alenya, J. Andrade-Cetto, E. Martínez, and C. Torras. Affine epipolar direction from two views ofa planar contour. In *Proc. Adv. Concepts Intell. Vision Syst. Conf., LNCS 4179*, pages 944–955, Antwerp, Sep. 2006.

G. Alenyà, J. Escoda, A.B.Martínez, and C. Torras. Using laser and vision to locate a robot in an industrial environment: A practical experience. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 3539–3544, Barcelona, Apr. 2005.

G. Alenyà, E. Martínez, and C. Torras. Fusing visual and inertial sensing to recover robot ego motion. *Journal of Robotic Systems*, 21:23–32, 2004.

P. Pomiers. Integration policy for critical multilayered distributed robotics applications. In *IEEE Intelligent Vehicle Symposium*, 2002.

2

Visually Guided Robotics Using Conformal Geometric Computing

Eduardo Bayro-Corrochano, Luis Eduardo Falcon-Morales and Julio Zamora-Esquivel,
CINVESTAV, Department of Electrical Engineering and Computer Science. Unidad Guadalajara, Jalisco, Mexico

1. Abstract

Classical Geometry, as conceived by Euclid, was a plataform from which Mathematics started to build its actual form. However, since the XIX century, it was a language that was not evolving as the same pase as the others branches of Physics and Mathematics. In this way, analytic, non-Euclidean and projective geometries, matrix theory, vector calculus, complex numbers, rigid and conformal transformations, ordinary and partial differential equations, to name some, are different mathematical tools which are used nowadays to model and solve almost any problem in robotic vision, but the presence of the classical geometric theory in such solutions is only implicit. However, over the last four decades a new mathematical framework has been developed as a new lenguage where not only the classical geometry is included, but where many of these mathematical systems will be embedded too. Instead of using different notation and theory for each of those systems, we will simplify the whole study introducing the CGA, a unique mathematical framework where all those systems are embedded, gaining in principle clarity and simplicity. Moreover, incidence algebra operations as union and intersection of subspaces, are also included in this system through the meet and join operations. In this regard, CGA appears promising for dealing with kinematics, dynamics and projective geometric problems in one and only one mathematical framework.
In this chapter we propose simulated and real tasks for perception-action systems, treated in a unify way and using only operations and geometrical entities of this algebra. We propose applications to follow geometric primitives or ruled sufaces with an arm's robot for shape understanding and object manipulation, as well as applications in visual grasping. But we believe that the use of CGA can be of great advantage in visually guided robotics using stereo vision, range data, laser, omnidirectional or odometry based systems.

Keywords: Computer vision; Clifford (geometric) algebra; projective and affine geometry; spheres projective geometry; incidence algebra; 3D rigid motion; ruled surfaces; directed distance; visually guided robotics.

2. Introduction

The Occam's razor, a mediaeval logical principle, said that 'when you have two competing theories which make exactly the same predictions, the one that is simpler is the better'. From this perspective the CGA is a single mathematical framework that unify and include different systems as matrix algebra, projective geometry, conformal transformations and differential forms. This chapter is an introduction to the communities of computer vision and robotics of this novel computational framework, called Conformal Geometric Algebra (CGA). This subject has been also treated in a wide scope in [4].
Our mathematical approach appears promising for the development of perception action cycle systems, see Figure 1. The subjects of this chapter are an improvement to previous works [3, 5, 6, 7, 13], because using the CGA we are now including the group of transformations in our computations and expanding our study to more complex surfaces, the ruled surfaces. Other authors have used Grassmann–Cayley algebra in computer vision [14] and robotics [19], but while they can express in this standard mathematical system the key ideas of projective geometry, such as the meet, join, duality and projective split, it lacks of an inner (contractive) product and of the group of transformations, which cannot be included in a very simple and natural way to the system.

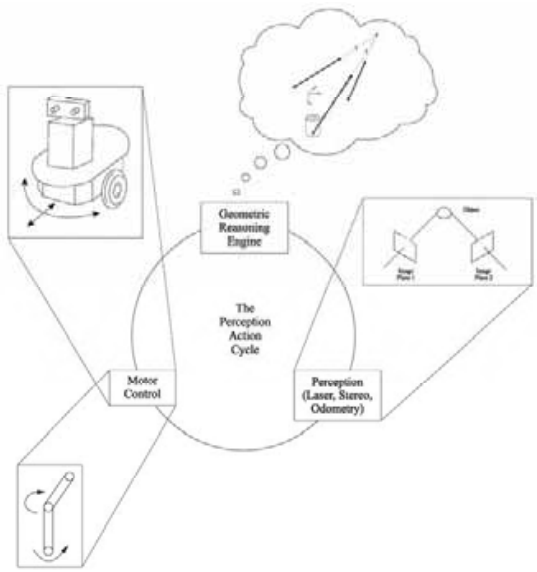


Fig. 1. Abstraction of the perception action cycle.

In fact, in the 1960's CGA take up again a proposal 'seeded' in the XIX century about build a global mathematical framework, which would include the main mathematical systems of that era: matrices and determinants; vector calculus; complex numbers; conformal transformations; Euclidean and projective spaces; differential forms; differential geometry; ordinary and partial differential equations.

In this chapter we put a lot of effort to explain clearly the CGA, illustrating the computations in great detail. Using the same ideas showed in this chapter, another practical tasks of visual guided robotics could be implemented for 3D motion estimation, *body − eye* calibration, 3D reconstruction, navigation, reaching and grasping 3D objects, etc. Thus, the idea is to introduce a suitable computational framework to the computer vision and robotic communities, which can be of great advantage for future applications in stereo vision, range data, laser, omnidirectional and odometry based systems.

CGA is the fusion of the Clifford Geometric Algebra (GA) and the non-Euclidean Hyperbolic Geometry. Historically, GA and CGA has not been taken into consideration seriously by the scientific community, but now and after the work of David Hestenes [10] and Pertti Lounesto [15] it has been taking a new scope of perspectives, not only theoretically, but for new and innovative applications to physics, computer vision, robotics and neural computing. One of the critics against CGA is the wrong idea that this system can manipulate only basic entities (points, lines, planes and spheres) and therefore it won't be useful to model general two and three dimensional objects, curves, surfaces or any other nonlinear entity required to solve a problem of a perception action system in robotics and computer vision. However, in this chapter we present the CGA, with its *algebra of incidence* [12] and rigid-motio*n* transformations, to obtain several practical techniques in the resolution of problems of perception action systems including ruled surfaces: 3D motion guidance of very non-linear curves; reaching and 3D object manipulation on very non-linear surfaces.

There are several interest points to study ruled surfaces: as robots and mechanisms are moving, any line attached to them will be tracing out a ruled surface or some other high nonlinear 3D-curve; the industry needs to guide the arm of robots with a laser welding to joint two ruled surfaces; reaching and manipulating 3D-objects is one of the main task in robotics, and it is usual that these objects have ruled surfaces or revolution surfaces; to guide a robot's arm over a critical 2D or 3D-curve or any other configuration constraint, and so forth.

The organization of this chapter paper is as follows: section two presents a brief introduction to conformal geometric algebra. Section three explains how the affine plane is embedded in the CGA. Section four shows how to generate the rigid transformations. In section five we present the way that several ruled surfaces or complex three dimensional curves can be generated in a very simple way using CGA. Section six shows how motors are usuful to obtain the Barret Hand™ forward kinematics. Section seven presents the real and simulated applications to follow geometric primitives and ruled surfaces for shape understanding and object manipulation, and section eight the applications to visual grasping identification. Conclusion are given in section nine.

2. Geometric Algebra

In general, a geometric algebra G_n is a 2^n-dimensional non-commutative algebra generated from a n-dimensional vector space V^n. Let us denote as $G_{p,q,r}$ this algebra where p, q, r denote the signature p, q, r of the algebra. If $p \neq 0$and $q = r = 0$, we have the standard Euclidean space and metric, if only $r \neq 0$ the metric is pseudoeuclidean and if $r \neq 0$ the metric is degenerate. See [17, 11] for a more detailed introduction to conformal geometric algebra.

We will use the letter e to denote the vector basis ei. In a geometric algebra Gp,q,r, the geometric product of two basis vectors is defined as

$$e_i e_j = \begin{cases} 1 & for\ i = j \in 1, ..., p \\ -1 & for\ i = j \in p+1, ..., p+q \\ 0 & for\ i = j \in p+q+1, ..., p+q+r \\ e_i \wedge e_j & for\ i \neq j \end{cases} \quad (1)$$

2.1 Conformal Geometric Algebra

The geometric algebra of a 3D Euclidean space $G_{3,0,0}$ has a point basis and the motor algebra $G_{3,0,1}$ a line basis. In the latter geometric algebra the lines expressed in terms of Plücker coordinates can be used to represent points and planes as well. The reader can find a comparison of representations of points, lines and planes using $G_{3,0,0}$ and $G_{3,0,1}$ in [8].

Interesting enough in the case of the conformal geometric algebra we find that the unit element is the sphere which allows us to represent the other geometric primitives in its terms. To see how this is possible we begin giving an introduction in conformal geometric algebra following the same formulation presented in [11] and show how the Euclidean vector space R^n is represented in $R^{n+1,1}$. Let $\{e_1, .., e_n, e_+, e_-\}$ be a vector basis with the following properties

$$e_i^2 = 1, \quad i = 1.., n; \quad (2)$$

$$e_\pm^2 = \pm 1, \quad (3)$$

$$e_i \cdot e_+ = e_i \cdot e_- = e_+ \cdot e_- = 0, \quad i = 1, .., n. \quad (4)$$

Note that this basis is not written in bold. A *null basis* $\{e_0, e_\infty\}$ can be introduced by

$$e_0 = \frac{(e_- - e_+)}{2}, \quad (5)$$

$$e_\infty = e_- + e_+, \quad (6)$$

with the properties

$$e_0^2 = e_\infty^2 = 0, \quad e_\infty \cdot e_0 = -1. \quad (7)$$

A unit pseudoscalar $E \in \mathbb{R}^{1,1}$ which represents the so-called Minkowski plane is defined by

$$\boldsymbol{E} = e_\infty \wedge e_0 = e_+ \wedge e_- = e_+ e_-. \quad (8)$$

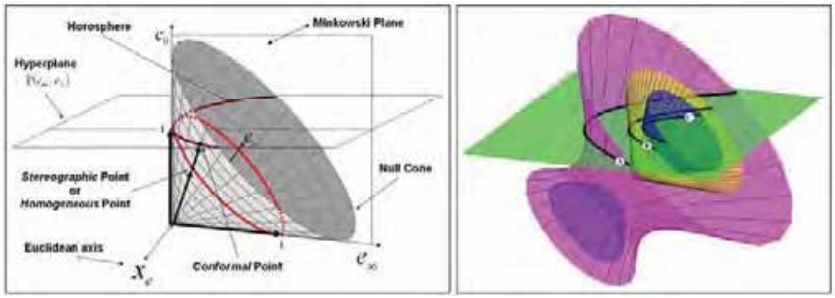


Fig. 2. (a) The Null Cone and the Horosphere for 1-D, and the conformal and stereographic representation of a 1-D vector. (b) Surface levels *A*, *B* and *C* denoting spheres of radius positive, zero and negative, respectively.

One of the results of the non-Euclidean geometry demonstrated by Nikolai Lobachevsky in the XIX century is that in spaces with hyperbolic structure we can find subsets which are

isomorphic to a Euclidean space. In order to do this, Lobachevsky introduced two constraints, to the now called *conformal point* $x_c \in \mathbb{R}^{n+1,1}$. See Figure 2(a). The first constraint is the *homogeneous* representation, *normalizing* the vector x_c such that

$$x_c \cdot e_\infty = -1, \tag{9}$$

and the second constraint is such that the vector must be a *null vector*,thatis,

$$x_3^c = 0^* \tag{10}$$

Thus, conformal points are required to lie in the intersection surface, denoted $\mathbb{N}_e^n$, between the *null cone* $\mathbb{N}^{n+1}$ and the *hyperplane* $\mathbb{P}(e_\infty, e_0)$:

$$\begin{aligned} \mathbb{N}_e^n &= \mathbb{N}^{n+1} \cap \mathbb{P}(e_\infty, e_0) \\ &= \{\mathbf{x}_c \in \mathbb{R}^{n+1,1} | \mathbf{x}_c^2 = 0, \mathbf{x}_c \cdot e_\infty = -1\}. \end{aligned} \tag{11}$$

The constraint (11) define an isomorphic mapping between the Euclidean and the Conformal space. Thus, for each conformal point $x_c \in \mathbb{R}^{n+1,1}$ there is a unique Euclidean point $\mathbf{x}_e \in \mathbb{R}^n$ and unique scalars *a*, *β* such that the mapping $\mathbf{x}_e \mapsto x_c = \mathbf{x}_e + \alpha e_0 + \beta e_\infty$. Then, the *standard form* of a conformal point x_c is

$$x_c = \mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0. \tag{12}$$

Note that a conformal point x_c and be splitted as

$$x_c = \mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0 = (x_c \wedge E)E + (x_c \cdot E)E. \tag{13}$$

We can gain further insight into the geometrical meaning of the null vectors by analyzing the isomorphism given by equation (13). For instance by setting $x_e = 0$ we find that e_0 represents the origin of R^n (hence the name). Similarly, dividing this equation by $x_c \cdot e_0 = -\frac{1}{2} x_e^{\ 2}$ gives

$$\frac{x_c}{x_c \cdot e_0} = -\frac{2}{\mathbf{x}_e^2}(\mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0) = -\frac{2\mathbf{x}_e^2}{\mathbf{x}_e^2}(\frac{1}{\mathbf{x}_e} + \frac{1}{2}e_\infty + \frac{e_0}{\mathbf{x}_e^2}) = -2(\frac{1}{\mathbf{x}_e} + \frac{1}{2}e_\infty + \frac{e_0}{\mathbf{x}_e^2}) \underset{\mathbf{x}_e \to \infty}{\longrightarrow} e_\infty. \tag{14}$$

Thus we conclude that e_∞ represents the point at infinity.

The *dual* of a multivector $A \in Gn$ is defined by

$$\mathbf{A}^* = \mathbf{A}\, \mathbf{I}_n^{-1} \tag{16}$$

where $In \equiv e_{12\cdots n}$ is the unit *pseudoscalar* of *Gn* and the *inverse* of a multivector A*n*, if it exists, is defined by the equation $A^{-1}A=1$.

Duality give us the opportunity to define the *meet* $M \vee N$ between two multivectors *M* and *N* ,using one of the following equivalent expressions

$$meet(M,N) \equiv M \vee N = M^* \cdot N = M^* \wedge N^*. \tag{17}$$

Geometrically, this operation will give us the intersection between geometric primitives through the intersection of their generated subspaces. See [12].

2.2 Spheres and planes

The equation of a sphere of radius *ρ* centered at point p*e* $\in$ R^n can be written as

$$(\mathbf{x}_e - \mathbf{p}_e)^2 = \rho^2. \tag{18}$$

Since $\boldsymbol{x}_c \cdot \boldsymbol{y}_c = -\frac{1}{2}(\mathbf{x}_e - \mathbf{y}_e)^2$, we can rewrite the formula above in terms of homogeneous coordinates as

$$\boldsymbol{x}_c \cdot \boldsymbol{p}_c = -\frac{1}{2}\rho^2. \tag{19}$$

Since $x_c \cdot e_\infty = -1$ we can factor the expression above and then

$$\boldsymbol{x}_c \cdot (\boldsymbol{p}_c - \frac{1}{2}\rho^2 e_\infty) = 0, \tag{20}$$

which finally yields the simplified equation for the sphere as

$$\boldsymbol{x}_c \cdot \boldsymbol{s} = 0, \tag{21}$$

where

$$\boldsymbol{s} = \boldsymbol{p}_c - \frac{1}{2}\rho^2 e_\infty = \mathbf{p}_e + e_0 + \frac{\mathbf{p}_e^2 - \rho^2}{2} e_\infty \tag{22}$$

is the equation of the sphere. From this equation and (13) we can see that a conformal point is just a sphere with zero radius. The vector *s* has the properties

$$\boldsymbol{s}^2 = \rho^2 > 0, \tag{23}$$

$$e_\infty \cdot \boldsymbol{s} = -1. \tag{24}$$

From these properties, we conclude that the sphere *s* is a point lying on the hyperplane $xc \cdot e\infty = -1$, but *outside* the null cone $x^2 = 0$. In particular, all points on the hyperplane outside the horosphere determine spheres with positive radius, points lying on the horosphere define spheres of zero radius (i.e. points), and points lying inside the horosphere have imaginary radius. Finally, note that spheres of the same radius form a surface which is parallel to the horosphere.

Alternatively, spheres can be dualized and represented as $(n + 1)$-vectors $s^* = sI^{-1}$ and then using the *main convolution* $\bar{I}$ of I defined as

$$\bar{I} = (-1)^{\frac{1}{2}(n+2)(n+1)} I = -I^{-1}, \tag{25}$$

we can express the constraints of equations (23) and (24) as

$$\begin{aligned} \boldsymbol{s}^2 &= -\tilde{\boldsymbol{s}}^* \boldsymbol{s}^* = \rho^2, \\ e_\infty \cdot \boldsymbol{s} &= e_\infty \cdot (\boldsymbol{s}^* I) = (e_\infty \wedge \boldsymbol{s}^*) I = -1. \end{aligned} \tag{26}$$

The equation for the sphere now becomes

$$\boldsymbol{x}_c \wedge \boldsymbol{s}^* = 0. \tag{27}$$

The advantage of the dual form is that the sphere can be directly computed from four points (in 3D) as

$$\boldsymbol{s}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3} \wedge \boldsymbol{x}_{c_4}. \tag{28}$$

If we replace one of these points for the point at infinity we get

$$\pi^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3} \wedge e_\infty, \tag{29}$$

Developing the products, we get

$$\pi^* = \boldsymbol{x}_{c_3} \wedge \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge e_\infty = \mathbf{x}_{e_3} \wedge \mathbf{x}_{e_1} \wedge \mathbf{x}_{e_2} \wedge e_\infty + ((\mathbf{x}_{e_3} - \mathbf{x}_{e_1}) \wedge (\mathbf{x}_{e_2} - \mathbf{x}_{e_1}))\boldsymbol{E}, \tag{30}$$

which is the equation of the plane passing through the points x_{e1}, x_{e2} and x_{e3}. We can easily see that $x_{e1} \wedge x_{e2} \wedge x_{e3}$ is a pseudoscalar representing the volume of the parallelepiped with sides xe_1, $xe_2$and xe_3.Also, since $(xe_1 - xe_2)$ and $(xe_3 - xe_2)$ are two vectors on the plane, the expression $((x_{e1} - x_{e2}) \wedge (x_{e3} - x_{e2}))^*$ is the normal to the plane. Therefore planes are spheres passing through the point at infinity.

2.3 Geometric identities, duals and incidence algebra operations

A circle z can be regarded as the intersection of two spheres s_1 and s_2. This means that for each point on the circle $x_c \in z$ they lie on both spheres, that is, $x_c \in s_1$and $x_c \in s_2$. Assuming that s_1 and s_2 are linearly independent, we can write for $x_c \in z$

$$(\boldsymbol{x}_c \cdot \boldsymbol{s}_1)\boldsymbol{s}_2 - (\boldsymbol{x}_c \cdot \boldsymbol{s}_2)\boldsymbol{s}_1 = \boldsymbol{x}_c \cdot (\boldsymbol{s}_1 \wedge \boldsymbol{s}_2) = \boldsymbol{x}_c \cdot \boldsymbol{z} = 0, \tag{31}$$

this result tells us that since x_c lies on both spheres, z=($s1 \wedge s1$) should be the intersection of the spheres or a circle. It is easy to see that the intersection with a third sphere leads to a point pair. We have derived algebraically that the wedge of two linearly independent spheres yields to their intersecting circle (see Figure 3), this topological relation between two spheres can be also conveniently described using the dual of the meet operation, namely

$$\boldsymbol{z} = (\boldsymbol{z}^*)^* = (\boldsymbol{s}_1^* \vee \boldsymbol{s}_2^*)^* = \boldsymbol{s}_1 \wedge \boldsymbol{s}_2, \tag{32}$$

this new equation says that the dual of a circle can be computed via the meet of two spheres in their dual form. This equation confirms geometrically our previous algebraic computation of equation (31).

The dual form of the circle (in 3D) can be expressed by three points lying on it as

$$\boldsymbol{z}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge \boldsymbol{x}_{c_3}, \tag{33}$$

seeFigure3.a.

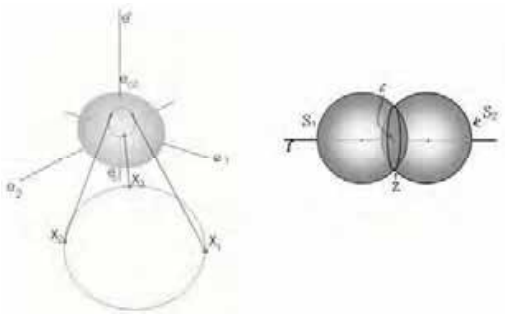


Fig. 3. a) Circle computed using three points, note its stereographic projection. b) Circle computed using the meet of two spheres.

Similar to the case of planes show in equation (29), lines can be defined by circles passing through the point at infinity as

$$\boldsymbol{l}^* = \boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge e_\infty. \tag{34}$$

This can be demonstrated by developing the wedge products as in the case of the planes to yield

$$\boldsymbol{x}_{c_1} \wedge \boldsymbol{x}_{c_2} \wedge e_\infty = \mathbf{x}_{c_1} \wedge \mathbf{x}_{c_2} \wedge e_\infty + (\mathbf{x}_{c_2} - \mathbf{x}_{c_1}) \wedge \boldsymbol{E}, \tag{35}$$

from where it is evident that the expression $x_{e1} \wedge x_{e2}$ is a bivector representing the plane where the line is contained and (x_{e2}– x_{e1}) is the direction of the line.

The dual of a point p is a sphere s. The intersection of four spheres yields a point, see Figure 4.b . The dual relationships between a point and its dual, the sphere, are:

$$\boldsymbol{s}^* = \boldsymbol{p}_1 \wedge \boldsymbol{p}_2 \wedge \boldsymbol{p}_3 \wedge \boldsymbol{p}_4 \leftrightarrow \boldsymbol{p}^* = \boldsymbol{s}_1 \wedge \boldsymbol{s}_2 \wedge \boldsymbol{s}_3 \wedge \boldsymbol{s}_4, \tag{36}$$

where the points are denoted as p_i and the spheres s_i for i =1, 2, 3, 4. A summary of the basic geometric entities and their duals is presented in Table 1.

There is another very useful relationship between a $(r-2)$-dimensional sphere A_r and the sphere s (computed as the dual of a point s). If from the sphere A_r we can compute the hyperplane $\boldsymbol{A}_{r+1} = \boldsymbol{e}_\infty \wedge \boldsymbol{A}_r \neq 0$, we can express the meet between the dual of the point s(a sphere) and the hyperplane A_{r+1} getting the sphere A_r of one dimension lower

$$(-1)^r \boldsymbol{s}^* \cap \boldsymbol{A}_{r+1} = (\boldsymbol{s}^* I) \cdot \boldsymbol{A}_{r+1} = \boldsymbol{s}\boldsymbol{A}_{r+1} = \boldsymbol{A}_r. \tag{37}$$

This result is telling us an interesting relationship: that the sphere A_r and the hyperplane A_{r+1} are related via the point s (dual of the sphere s^*), thus we then rewrite the equation (37) as follows

$$\boldsymbol{s} = \boldsymbol{A}_r \boldsymbol{A}_{r+1}^{-1}. \tag{38}$$

Using the equation (38) and given the plane π (A_{r+1}) and the circle z(A_r) we can compute the sphere

$$s = z\pi^{-1}. \tag{39}$$

Similarly we can compute another important geometric relationship called the *pair of points* using the equation (38) directly

$$s = PPL^{-1}. \tag{40}$$

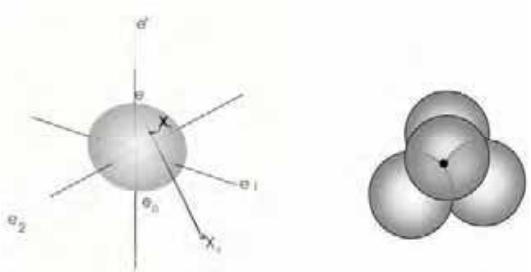


Fig. 4. a) Conformal point generated by projecting a point of the affine plane to the unit sphere. b) Point generated by the meet of four spheres.

Entity	Representation	Grade	Dual Representation	Grade
Sphere	$\boldsymbol{s} = \mathbf{p} + \frac{1}{2}(\mathbf{p}^2 - \rho^2)e_\infty + e_0$	1	$\boldsymbol{s}^* = \boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}\wedge\boldsymbol{d}$	4
Point	$\boldsymbol{x} = \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty + e_0$	1	$\boldsymbol{x}^* = \boldsymbol{s}_1\wedge\boldsymbol{s}_2\wedge\boldsymbol{s}_3\wedge\boldsymbol{s}_4$	4
Plane	$\pi = \boldsymbol{n}I_E - de_\infty$ $\boldsymbol{n} = (\boldsymbol{a}-\boldsymbol{b})\wedge(\boldsymbol{a}-\boldsymbol{c})$ $d = (\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c})I_E$	1	$\pi^* = e_\infty\wedge\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}$	4
Line	$\boldsymbol{L} = \pi_1\wedge\pi_2$ $\boldsymbol{L} = \boldsymbol{n}I_E - e_\infty\boldsymbol{m}I_E$ $\boldsymbol{n} = (\boldsymbol{a}-\boldsymbol{b})$ $\boldsymbol{m} = (\boldsymbol{a}\wedge\boldsymbol{b})$	2	$\boldsymbol{L}^* = e_\infty\wedge\boldsymbol{a}\wedge\boldsymbol{b}$	3
Circle	$\boldsymbol{z} = \boldsymbol{s}_1\wedge\boldsymbol{s}_2$	2	$\boldsymbol{z}^* = \boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}$	3
Point Pair	$\boldsymbol{PP} = \boldsymbol{s}_1\wedge\boldsymbol{s}_2\wedge\boldsymbol{s}_3$ $\boldsymbol{PP} = \boldsymbol{s}\wedge\boldsymbol{L}$	3 2	$\boldsymbol{PP}^* = \boldsymbol{a}\wedge\boldsymbol{b}$	2

Table 1. Entities in conformal geometric algebra.

Now using this result given the line $L$and the sphere s we can compute the pair of points PP (see Figure 5.b)

$$PP = sL = s \wedge L. \tag{41}$$

3. The 3D Affine Plane

In the previous section we described the general properties of the conformal framework. However, sometimes we would like to use only the projective plane of the conformal framework but not the null cone of this space. This will be the case when we use only rigid transformations and then we will limit ourselves to the *Affine Plane* which is a $n + 1$ dimensional subspace of the Hyperplane of reference $P(e_\infty, e_0)$.
We have chosen to work in the algebra $G_{4,1}$. Since we deal with homogeneous points the particular choice of null vectors does not affect the properties of the conformal geometry. Points in the affine plane $x \in R^{4,1}$ are formed as follows

$$x^a = \mathbf{x}_e + e_0, \tag{42}$$

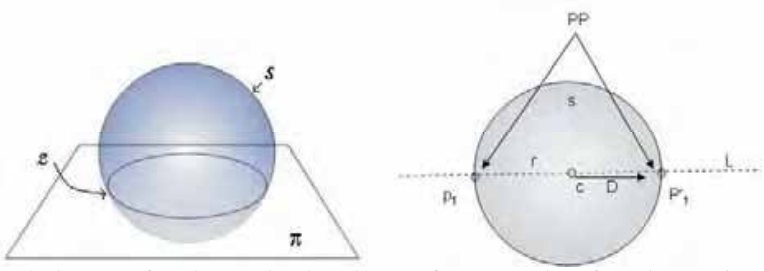


Fig. 5. a) The meet of a sphere and a plane. b) Pair of points resulting from the meet between a line and a sphere.

where $x_e \in R^3$. From this equation we note that e_0 represents the origin (by setting $x_e = 0$), similarly, $e\infty$ represents the point at infinity. Then the normalization property is now expressed as

$$e_\infty \cdot x^a = -1. \tag{43}$$

In this framework, the conformal mapping equation is expressed as

$$x_c = \mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0 = x^a + \frac{1}{2}\mathbf{x}_e^2 e_\infty. \tag{44}$$

For the case when we will be working on the affine plane exclusively, we will be mainly concerned with a simplified version of the *rejection*. Noting that $E = e_\infty \wedge e_0 = e_\infty \wedge e$, we write a equation for rejection as follows

$$\begin{aligned} P_E^{\perp}(x_c) &= (x_c \wedge E)E = (x_c \wedge E)\cdot E = (e_\infty \wedge e_0)\cdot e_0 + (x_c \wedge e_\infty)\cdot e_0, \\ \mathbf{x}_e &= -e_0 + (x_c \wedge e_\infty)\cdot e_0. \end{aligned} \tag{45}$$

Now, since the points in the affine plane have the form $x^a = xe + e0$, we conclude that

$$x^a = (x_c \wedge e_\infty)\cdot e_0, \tag{46}$$

is the mapping from the horosphere to the affine plane.

3.1 Lines and Planes

The lines and planes in the affine plane are expressed in a similar fashion to their conformal counterparts as the *join* of 2 and 3 points, respectively

$$L^a = x_1^a \wedge x_2^a, \tag{47}$$

$$\Pi^a = x_1^a \wedge x_2^a \wedge x_3^a. \tag{48}$$

Note that unlike their conformal counterparts, the line is a *bivector* and the plane is a *trivector*. As seen earlier, these equations produce a moment-direction representation thus

$$L^a = e_\infty \mathbf{d} + B, \tag{49}$$

where d is a vector representing the direction of the line and B is a bivector representing the moment of the line. Similarlywehavethat

$$\Pi^a = e_\infty \mathrm{n} + \delta e_{123} \tag{50}$$

where n is the normal vector to the plane and δ is a scalar representing the distance from the plane to the origin. Note that in any case, the direction and normal can be retrieved with $\mathrm{d} = e_\infty \cdot L^a$ and $\mathrm{n} = e_\infty \cdot \Pi^a$, respectively.

In this framework, the intersection or *meet* has a simple expression too. Let $A^a = a_1^a \wedge \ldots \wedge a_r^a$ and $B^a = b_1^a \wedge \ldots \wedge b_s^a$, then the meet is defined as

$$A^a \cap B^a = A^a \cdot (B^a \cdot \bar{I}_{A^a \cup B^a}), \tag{51}$$

where $\bar{I}_{A^a \cup B^a}$ is either $e_{12}e_\infty$, $e_{23}e_\infty$, $e_{31}e_\infty$,or $e_{123}e_\infty$, according to which basis vectors span the largest common space of A^a and B^a.

3.2 Directed distance

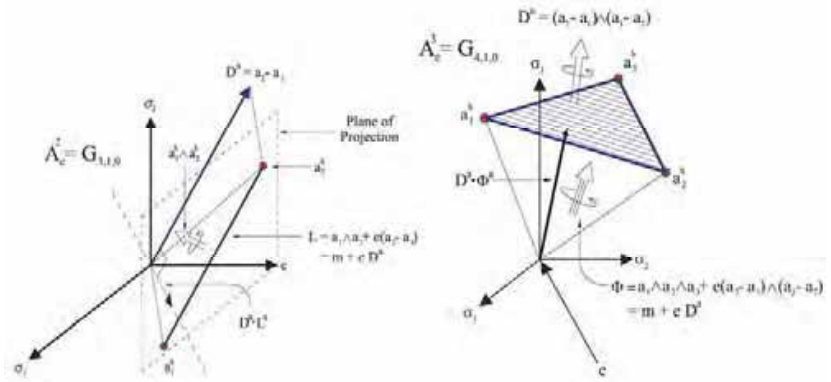


Fig. 6. a) Line in 2D affine space. b) Plane in the 3D affine space (note that the 3D space is "lifted" by a null vector e.

It is well known from vector analysis the so-called *Hessian normal form*, a convenient representation to specify lines and planes using their distance from the origin (the Hesse distance or Directed distance). In this section we are going to show how CGA can help us to obtain the Hesse distance for more general simplexes and not only for lines and planes. Figure 6(a) and (b) depict a line and a plane, respectively, that will help us to develop our equations. Let A^k be a k-line (or plane), then it consist of a *momentum* M^k of degree k and of a *direction* $D^{k-1}$of degree $k - 1$. For instance, given three Euclidean points a_1,a_2,a_3 their 2-simplex define a dual 3-plane in CGA that canbe expressedas

$$A^k \equiv \Phi = M^3 + D^2 e_0 = a_1 \wedge a_2 \wedge a_3 + (a_2 - a_1) \wedge (a_3 - a_1) e_0. \tag{52}$$

Then, the directed distance of this plane, denoted as p^k , can be obtained taking the inner product between the unit direction $D^{k-1}$and the moment M^k . Indeed, from (52) and using expressions (1) to (7), we get the direction
from $\Phi\ e_\infty = D^{k-1}$and then its unitary expression D^{k-1} dividing D^{k-1} by its magnitude. Schematically,

$$\boldsymbol{A}^k \longrightarrow \boldsymbol{A}^k \cdot e_\infty = \boldsymbol{D}^{k-1} \longrightarrow \boldsymbol{D}_u^{k-1} = \frac{\boldsymbol{D}^{k-1}}{\left|\boldsymbol{D}^{k-1}\right|}. \tag{53}$$

Finally the directed distance p^k of A^k is

$$\boldsymbol{p}^k = \boldsymbol{D}_u^{k-1} \cdot \boldsymbol{A}^k, \tag{54}$$

where the dot operation basically takes place between the direction $D^{k-1}$and the momentum of A^k . Obviously, the directed distance vector p touches orthogonally the k-plane A^k, and as we mentioned at the beginning of k this subsection, the magnitude p equals the Hesse distance. For sake of simplicity, in Figures (6.a) and (6.b) only $D^{k-1} \cdot L^k$ and $D^{k-1} \cdot \Phi^k$ are respectively shown.
Now, having this point from the first object, we can use it to compute the directed distance from the k-plane A^k parallel to the object B^k as follows

$$d[\boldsymbol{A}^k, \boldsymbol{B}^k] = d[\boldsymbol{D}^{k-1} \cdot \boldsymbol{A}^k, \boldsymbol{B}^k] = d[(e_\infty \cdot \boldsymbol{A}^k) \cdot \boldsymbol{A}^k, \boldsymbol{B}^k]. \tag{55}$$

4. Rigid Transformations

We can express rigid transformations in conformal geometry carrying out reflections between planes.

4.1 Reflection

The reflection of conformal geometric entities help us to do any other transformation. The reflection of a point x respect to the plane π is equal x minus twice the direct distance between the point and plane see the image (7), that is $x = x - 2(\pi \cdot x)\pi^{-1}$ to simplify this expression recalling the property of Clifford product of vectors $2(b \cdot a) = ab + ba$.

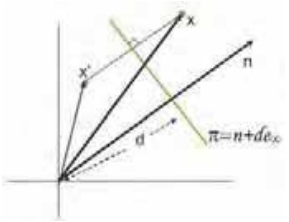


Fig. 7. Reflection of a point x respect to the plane π.

The reflection could be written

$$x' = x - (\pi x - x\pi)\pi^{-1}, \tag{56}$$

$$x' = x - \pi x\pi^{-1} - x\pi\pi^{-1} \tag{57}$$

$$x' = -\pi x\pi^{-1}. \tag{58}$$

For any geometric entity Q, the reflection respect to the plane π is given by

$$Q' = \pi Q\pi^{-1} \tag{59}$$

4.2 Translation

The translation of conformal entities can be by carrying out two reflections in parallel planes π_1 and π_2 see the image (8), that is

$$Q' = T_a\underbrace{(\pi_2\pi_1)}Q\widetilde{T_a}\underbrace{(\pi_1^{-1}\pi_2^{-1})} \tag{60}$$

$$T_a = (n + de_\infty)n = 1 + \frac{1}{2}ae_\infty = e^{-\frac{a}{2}e_\infty} \tag{61}$$

With $a = 2dn$.

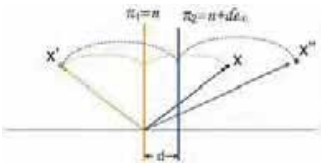


Fig. 8. Reflection about parallel planes.

4.3 Rotation

The rotation is the product of two reflections between nonparallel planes see image (9)

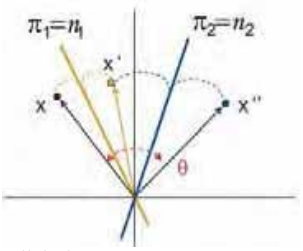


Fig. 9. Reflection about nonparallel planes.

$$Q' = R_\theta\underbrace{(\pi_2\pi_1)}Q\widetilde{R_\theta}\underbrace{(\pi_1^{-1}\pi_2^{-1})} \tag{62}$$

Or computing the conformal product of the normals of the planes.

$$R_\theta = n_2n_1 = Cos(\frac{\theta}{2}) - Sin(\frac{\theta}{2})l = e^{-\frac{\theta}{2}l} \tag{63}$$

With $l = n2 \wedge n1$,and θ twice the angle between the planes π_2 and π_1. The screw motion *called motor* related to an arbitrary axis L is $M = TRT$

4.4 Kinematic Chains

The direct kinematics for serial robot arms is a succession of motors and it is valid for points, lines, planes, circles and spheres.

$$Q' = ni = 1\prod M_iQni = 1\prod\widetilde{M}_{n-i+1} \tag{64}$$

5. Ruled Surfaces

Conics, ellipsoids, helicoids, hyperboloid of one sheet are entities which can not be directly described in CGA, however, can be modeled with its multivectors. In particular, a ruled

surface is a surface generated by the displacement of a straight line (called generatrix) along a directing curve or curves (called directrices). The plane is the simplest ruled surface, but now we are interested in nonlinear surfaces generated as ruled surfaces. For example, a circular cone is a surface generated by a straight line through a fixed point and a point in a circle. It is well known that the intersection of a plane with the cone can generate the conics. See Figure 10. In [17] the cycloidal curves can be generated by two coupled twists. In this section we are going to see how these and other curves and surfaces can be obtained using only multivectors of CGA.

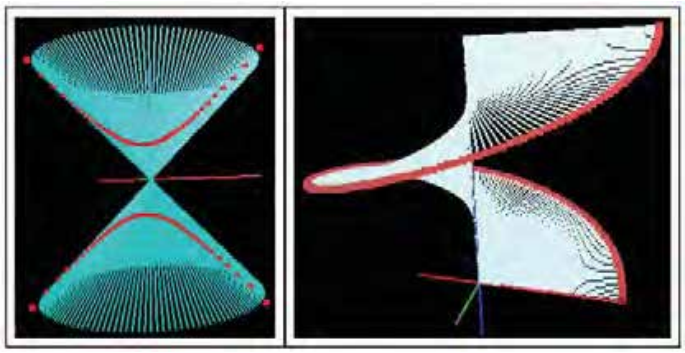


Fig. 10. (a) Hyperbola as the meet of a cone and a plane. (b) The helicoid is generated by the rotation and translation of a line segment. In CGA the *motor* is the desired multivector.

5.1 Cone and Conics

A circular cone is described by a fixed point v_0 (*vertex*), a dual circle $z_0 = a_0 \wedge a_1 \wedge a_2$ (*directrix*) and a rotor R (θ, l), $\theta \in [0, 2\pi)$ rotating the straight line $L(v_0,a_0) = v_0 \wedge a_0 \wedge e_\infty$, (*generatrix*) along the axis of the cone $l_0 = z_0 \cdot e_\infty$. Then, the cone w is generated as

$$w = R(\theta, l_0)\, L(v_0, a_0) \tilde{R}(\theta, l_0), \ \theta \in [0, 2\pi) \tag{67}$$

A conic curve can be obtained with the *meet* (17) of a cone and a plane. See Figure 10(a).

5.2 Cycloidal Curves

The family of the cycloidal curves can be generated by the rotation and translation of one or two circles. For example, the cycloidal family of curves generated by two circles of radius $r_0$and r_1 are expressed by, see Figure 11, the motor

$$M = TR_1T^4R_2 \tag{68}$$

where

$$T = T((r_0 + r_1)(sin(\theta)e_1 + cos(\theta)e_2)) \tag{69}$$

$$R_1 = R_1(\frac{r_0}{r_1}\theta) \tag{70}$$

$$R_2 = R_2(\theta) \tag{71}$$

Then, each conformal point x is transformed as MxM.

5.3 Helicoid

We can obtain the ruled surface called helicoid rotating a ray segment in a similar way as the spiral of Archimedes. So, if the axis e_3 is the directrix of the rays and it is orthogonal to them, then the translator that we need to apply is a multiple of θ, the angle of rotation. See Figure 10(b).

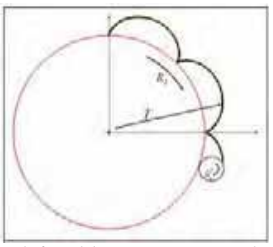


Fig. 11. The *motor* $M = TR_1T^*R_2$, defined by two rotors and one translator, can generate the family of the cycloidal curves varying the multivectors R_i and T.

5.4 Sphere and Cone

Let us see an example of how the algebra of incidence using CGA simplify the algebra. The intersection of a cone and a sphere in general position, that is, the axis of the cone does not pass through the center of the sphere, is the three dimensional curve of all the euclidean points (x, y, z) such that x and y satisfy the quartic equation

$$[x^2(1+\frac{1}{c^2}) - 2x_0x + y^2(1+\frac{1}{c^2}) - 2y_0y + x_0^2 + y_0^2 + z_0^2 - r^2]^2 = 4z_0^2(x^2+y^2)/c^2 \tag{72}$$

and x, y and z the quadratic equation

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r. \tag{73}$$

See Figure 12. In CGA the set of points q of the intersection can be expressed as the meet (17) of the dual sphere s and the cone w, (67), defined in terms of its generatrix L,that is

$$q = (s^*) \cdot [R(\theta, l_0)\, L(v_0, a_0) \tilde{R}(\theta, l_0)],\ \theta \in [0, 2\pi). \tag{74}$$

Thus, in CGA we only need (74) to express the intersection of a sphere and a cone, meanwhile in euclidean geometry it is necessary to use (72) and (73).

Fig. 12. Intersection as the *meet* of a sphere and a cone.

5.5 Hyperboloid of one sheet

The rotation of a line over a circle can generate a hyperboloid of one sheet. Figure 13(a).

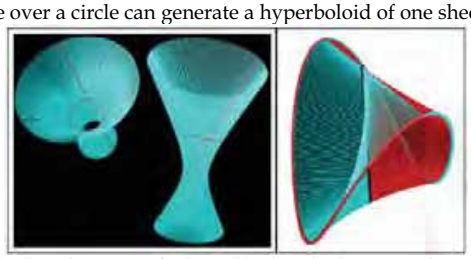


Fig. 13. (a) Hyperboloid as the rotor of a line. (b) The Plücker conoid as a ruled surface.

5.6 Ellipse and Ellipsoid

The ellipse is a curve of the family of the cycloid and with a translator and a dilator we can obtain an ellipsoid.

5.7 Plücker Conoid

The cylindroid or Pl¨ucker conoid is a ruled surface. See Figure 13(b). This ruled surface is like the helicoid where the translator parallel to the axis e_3 is of magnitude, a multiple of $cos(\theta)sin(\theta)$. The intersection curve of the conoid with a sphere will be obtained as the *meet* of both surfaces. Figure 14(a) and (b).

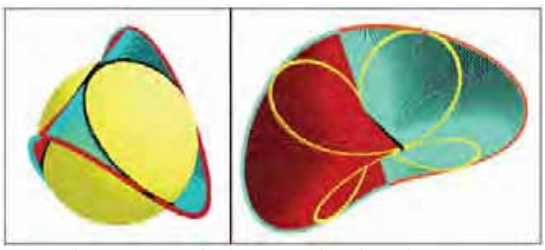


Fig. 14. The intersection between the Plücker conoid and a sphere.

6. Barrett Hand Forward Kinematics

The direct kinematics involves the computation of the position and orientation of the robot end-effector given the parameters of the joints. The direct kinematics can be easily computed if the lines of the screws' axes are given [2].

In order to introduce the kinematics of the Barrett Hand^{TM} we will show the kinematic of one finger, assuming that it is totally extended. Note that such an hypothetical position is not reachable in normal operation.

Let x_{1o}, x_{2o} be points-vectors describing the position of each joint of the finger and x_{3o} the end of the finger in the Euclidean space, see the Figure 15. If A_w, $A_{1,2,3}$ and D_w are denoting the dimensions of the finger's components

$$x_{1o} = A_{we1} + A_{1e2} + D_{we3}, \tag{75}$$

$$x_{2o} = A_{we1} + (A_1 + A_2)e_2 + D_{we3}, \tag{76}$$

$$x_{3o} = A_{we1} + (A_1 + A_2 + A_3)e_2 + D_{we3}. \tag{77}$$

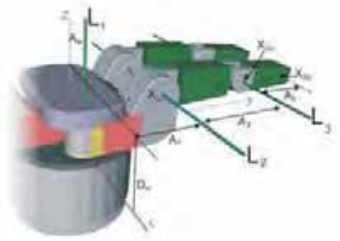


Fig. 15. Barrett hand hypothetical position.

Once we have defined these points it is quite simple to calculate the axes $L_{1o,2o,3o}$, which will be used as motor's axis. As you can see at the Figure 15,

$$L_{1o} = -A_w(e_2 \wedge e_\infty) + e_{12}, \tag{78}$$

$$L_{2o} = (x_{1o} \wedge e_1 \wedge e_\infty)\, I_c, \tag{79}$$

$$L_{3o} = (x_{2o} \wedge e_1 \wedge e_\infty)\, I_c. \tag{80}$$

When the hand is initialized the fingers moves away to the home position, this is the angle $\Phi 2 = 2.46^o$ by the joint two and the angle $\Phi 3 = 50^o$ degrees by the joint three. In order to move the finger from this hypothetical position to its home position the appropriate transformation is as follows:

$$M_{2o} = cos\,(\Phi 2/2) - sin(\Phi_2/2)L_{2o} \tag{81}$$

$$M_{3o} = cos\,(\Phi 3/2) - sin(\Phi_3/2)L_{3o}. \tag{82}$$

Once we have gotten the transformations, then we apply them to the points x_{2o} and x_{3o} in order to get the points x_2 and x_3 that represents the points in its home position, also the line L_3 is the line of motor axis in home position.

$$x_2 = M_{2o}x_{2o}\widetilde{M}_{2o}, \tag{83}$$

$$x_3 = M_{2o}M_{3o}x_{3o}\widetilde{M}_{3o}\widetilde{M}_{2o}, \tag{84}$$

$$L_3 = M_{2o}L_{3o}\widetilde{M}_{2o}. \tag{85}$$

The point $x_1 = x_{1o}$ is not affected by the transformation, the same for the lines $L_1 = L_{1o}$ and $L_2 = L_{2o}$ see Figure 16. Since the rotation angles of both axis $L2$ and $L3$ are related, we will use fractions of the angle q_1 to describe their individual rotation angles. The motors of each joint are computed using $\frac{2}{35}q_4$ to rotate around L_1, $\frac{1}{125}q_1$ around L_2 and $\frac{1}{375}q_1$ around $L3$, these specific angle coefficients where taken from the Barrett Hand user manual.

Fig. 16. Barrett hand at home position.

$$M_1 = cos(q_4/35) + sin(q_4/35)L_1, \tag{86}$$

$$M_2 = cos(q_1/250) - sin(q_1/250)L_2, \tag{87}$$

$$M_3 = cos(q_1/750) - sin(q_1/750)L_3. \tag{88}$$

The position of each point is related to the angles q_1 and q_4 as follows:

$$x_1' = M_1 x_1 \widetilde{M}_1, \tag{89}$$

$$x_2' = M_1 M_2 x_2 \widetilde{M}_2 \widetilde{M}_1, \tag{90}$$

$$x_3' = M_1 M_2 M_3 x_3 \widetilde{M}_3 \widetilde{M}_2 \widetilde{M}_1. \tag{91}$$

7. Application I: Following Geometric Primitives and Ruled Surfaces for Shape Understanding and Object Manipulation

In this section we will show how to perform certain object manipulation tasks in the context of conformal geometric algebra. First, we will solve the problem of positioning the gripper of the arm in a certain position of space disregarding the grasping plane or the gripper's alignment. Then, we will illustrate how the robotic arm can follow linear paths.

7.1 Touching a point

In order to reconstruct the point of interest, we make a back-projection of two rays extended from two views of a given scene (see Figure 17). These rays will not intersect in general, due to noise. Hence, we compute the directed distance between these lines and use the the middle point as target. Once the 3D point *pt* is computed with respect to the cameras' framework, we transform it to the arm's coordinate system.

Once we have a target point with respect to the arm's framework, there are three cases to consider. There might be several solutions (see Figs. 18.a and 19.a), a single solution (see Figure 18.b), or the point may be impossible to reach (Figure 19.b).

In order to distinguish between these cases, we create a sphere $\boldsymbol{S}_t = \boldsymbol{p}_t - \frac{1}{2}d_3^2 e_\infty$ centered at the point $\boldsymbol{p}_t$ and intersect it with the bounding sphere $\boldsymbol{S}_e = \boldsymbol{p}_0 - \frac{1}{2}(d_1 + d_2)^2 e_\infty$ of the other joints (see Figures 18.a and 18.b), producing the circle $z_s = S_e \wedge S_t$.

If the spheres S_t and S_e intersect, then we have a solution circle z_s which represents all the possible positions the point p_2 (see Figure 18) may have in order to reach the target. If the spheres are tangent, then there is only one point of intersection and a single solution to the problem as shown in Figure 18.b.

Fig. 17. Point of interest in both cameras (*pt*).

If the spheres do not intersect, then there are two possibilities. The first case is that S_t is outside the sphere S_e. In this case, there is no solution since the arm cannot reach the point p_t as shown in Figure 19.b. On the other hand, if the sphere St is inside Se, then we have a sphere of solutions. In other words, we can place the point p_2 anywhere inside S_t as shown in Figure 19.a. For this case, we arbitrarily choose the upper point of the sphere St.

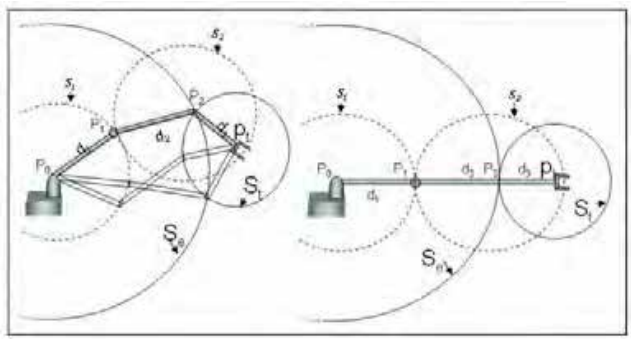


Fig. 18. a) S_e and S_t meet (infinite solutions) b) S_e and S_t are tangent (single solution).

In the experiment shown in Figure 20.a, the sphere St is placed inside the bounding sphere Se, therefore the point selected by the algorithm is the upper limit of the sphere as shown in Figures 20.a and 20.b. The last joint is completely vertical.

7.2 Line of intersection of two planes

In the industry, mainly in the sector dedicated to car assembly, it is often required to weld pieces. However, due to several factors, these pieces are not always in the same position, complicating this task and making this process almost impossible to automate. In many cases the requirement is to weld pieces of straight lines when no points on the line are available. This is the problem to solve in the following experiment.

Since we do not have the equation of the line or the points defining this line we are going to determine it via the intersection of two planes (the welding planes). In order to determine each plane, we need three points. The 3D coordinates of the points are

triangulated using the stereo vision system of the robot yielding a configuration like the one shown in Figure 21.

Once the 3D coordinates of the points in space have been computed, we can find now each plane as $\pi^* = \boldsymbol{x}_1 \wedge \boldsymbol{x}_2 \wedge \boldsymbol{x}_3 \wedge e_\infty$, and $\pi'^* = \boldsymbol{x}'_1 \wedge \boldsymbol{x}'_2 \wedge \boldsymbol{x}'_3 \wedge e'_\infty$. The line of intersection is computed via the *meet* operator $\boldsymbol{l} = \pi' \cap \pi$. In Figure 22.a we show a simulation of the arm following the line produced by the intersection of these two planes.

Once the line of intersection $\boldsymbol{l}$ is computed, it suffices with translating it on the plane $\psi = \boldsymbol{l}^* \wedge e_2$ (see Figure 22.b) using the translator $\boldsymbol{T}_1 = 1+\gamma e_2 e_\infty$, in the direction of e_2 (the y axis) a distance γ. Furthermore, we build the translator $\boldsymbol{T}_2 = 1+d_3 e_2 e_\infty$ with the same direction ($e2$), but with a separation d_3 which corresponds to the size of the gripper. Once the translators have been computed, we find the lines $\boldsymbol{l}'$ and $\boldsymbol{l}''$ by translating the line $\boldsymbol{l}$ with $\boldsymbol{l}' = \boldsymbol{T}_1 \boldsymbol{l} \boldsymbol{T}_1^{-1}$, and $\boldsymbol{l}'' = \boldsymbol{T}_2 \boldsymbol{l}' \boldsymbol{T}_2^{-1}$.

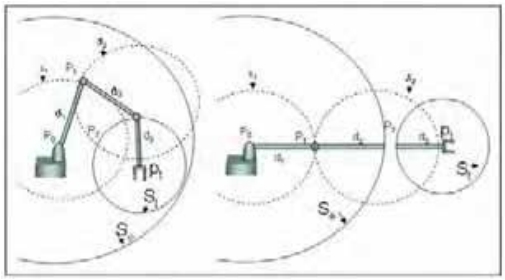


Fig. 19. a) *St* inside *Se* produces infinite solutions, b) *St* outside *Se*, no possible solution.

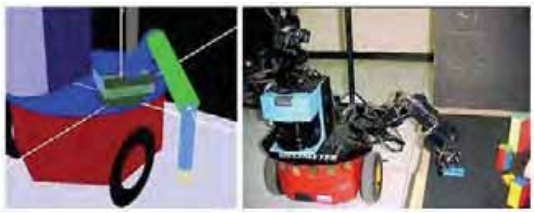


Fig. 20. a) Simulation of the robotic arm touching a point. b) Robot "Geometer" touching a point with its arm.

The next step after computing the lines, is to find the points $\boldsymbol{p}_t$ and $\boldsymbol{p}_2$ which represent the places where the arm will start and finish its motion, respectively. These points were given manually, but they may be computed with the intersection of the lines $\boldsymbol{l}'$ and $\boldsymbol{l}''$ with a plane that defines the desired depth. In order to make the motion over the line, we build a translator $\boldsymbol{T}_L = 1-\Delta_L l e_\infty$ with the same direction as l as shown in Figure 22.b. Then, this translator is applied to the points $\boldsymbol{p}_2 = \boldsymbol{T}_L \boldsymbol{p}_2 \boldsymbol{T}_L^{-1}$ and $\boldsymbol{p}_t = \boldsymbol{T}_L \boldsymbol{p}_t \boldsymbol{T}_L^{-1}$ in an iterative fashion to yield a displacement ΔL on the robotic arm.

By placing the end point over the lines and p_2 over the translated line, and by following the path with a translator in the direction of l we get a motion over l as seen in the image sequence of Figure 23.

7.3 Following a spherical path

This experiment consists in following the path of a spherical object at a certain fixed distance from it. For this experiment, only four points on the object are available (see Figure 24.a).

After acquiring the four 3D points, we compute the sphere $S^* = x^1 \wedge x^2 \wedge x^3 \wedge x^4$. In order to place the point p_2 in such a way that the arm points towards the sphere, the sphere was expanded using two different dilators. This produces a sphere that contains S^* and ensures that a fixed distance between the arm and S^* is preserved, as shown in Figure 24.b.

The dilators are computed as follows

$$D_\gamma = e^{-\frac{1}{2}\ln(\frac{\gamma+\rho}{\rho})E}, \tag{92}$$

$$D_d = e^{-\frac{1}{2}\ln(\frac{d_3+\gamma+\rho}{\rho})E}. \tag{93}$$

The spheres S_1 and S_2 are computed by dilating St:

$$S_1 = D_\gamma S_t D_\gamma^{-1}, \tag{94}$$

$$S_2 = D_d S_t D_d^{-1}. \tag{95}$$

Guiding lines for the robotic arm produced by the intersection, *meet*, of the planes and vertical translation.

We decompose each sphere in its parametric form as

$$p_t = M_1(\varphi)M_1(\phi)p_{s_1}M_1^{-1}(\phi)M_1^{-1}(\varphi), \tag{96}$$

$$p_2 = M_2(\varphi)M_2(\phi)p_{s_2}M_2^{-1}(\phi)M_2^{-1}(\varphi). \tag{97}$$

Where p_s is any point on the sphere. In order to simplify the problem, we select the upper point on the sphere. To perform the motion on the sphere, we vary the parameters φ and φ and compute the corresponding pt and p_2 using equations (96) and (97). The results of the simulation are shown in Figure 25.a, whereas the results of the real experiment can be seen in Figures 25.b and 25.c.

Fig. 21. Images acquired by the binocular system of the robot "Geometer" showing the points on each plane.

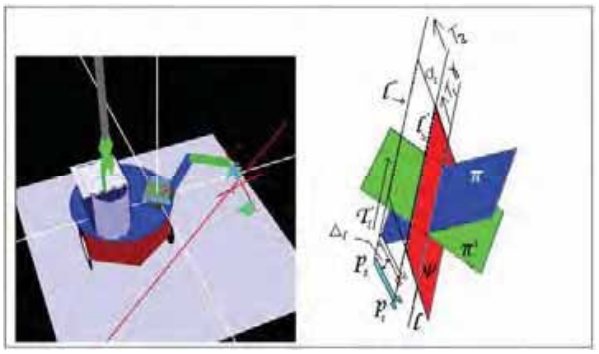


Fig. 22. a. Simulation of the arm following the path of a line produced by the intersection of two planes. b.

7.4 Following a 3D-curve in ruled surfaces

As another simulated example using ruled surfaces consider a robot arm laser welder. See Figure 26. The welding distance has to be kept constant and the end-effector should follow a 3D-curve w on the ruled surface guided only by the directrices d_1, d_2 and a guide line L. From the generatrices we can always generate the nonlinear ruled surface, and then with the meet with another surface we can obtain the desired 3D-curve. We tested our simulations with several ruled surfaces, obtaining expressions of high nonlinear surfaces and 3D-curves, that with the standard vector and matrix analysis it would be very difficult to obtain them.

Fig. 23. Image swquence of a linear-path motion.

Fig. 24. a) Points over the sphere as seen by the robot "Geometer". b) Guiding spheres for the arm's motion.

8. Aplications II: Visual Grasping Identification

In our example considering that the cameras can only see the surface of the observed objects, thus we will consider them as bi-dimensional surfaces which are embedded in a 3D space, and are described by the function

$$\vec{H}(s,t) = h_x(s,t)e_1 + h_y(s,t)e_2 + h_z(s,t)e_3 \tag{98}$$

where s and t are real parameters in the range [0, 1]. Such parameterization allows us to work with different objects like points, conics, quadrics, or even more complex objects like cups, glasses, etc. The table 2 shows some parameterized objects.

Particle	$\vec{H} = 3e_1 + 4e_2 + 5e_3$
Cylinder	$\vec{H} = cos(t)e_1 + sin(t)e_2 + se_3$
Plane	$\vec{H} = te_1 + se_2 + (3s + 4t + 2)e_3$

Table 2. Parameterized Objects.

Due to that our objective is to grasp such objects with the Barrett Hand, we must consider that it has only three fingers, so the problem consists in finding three "touching points" for which the system is in equilibrium during the grasping; this means that the sum of the forces equals to zero, and also the sum of the moments. For this case, we consider that there exists friction in each "touching point".

If the friction is being considered, we can claim that over the surface $H(s, t)$ a set of forces exist which can be applied. Such forces are inside a cone which have the normal $N(s, t)$ of the surface as its axis (as shown in Fig. 27). Its radius depends on the friction's coefficient $\|F - F_n\| \leq -\mu(|F_n|)$, where $Fn = (F \cdot N(s, t))N(s, t)$ is the normal component of F. The angle θ for the incidence of F with respect to the normal can be calculated using the wedge product, and should be smaller than a fixed $\theta\mu$

$$\frac{\|F \wedge N(s,t)\|}{F \cdot N(s,t)} \leq tan(\theta_\mu) \tag{99}$$

Fig. 25. a) Simulation of the motion over a sphere. b) and c) Two of the images in the sequence of the real experiment.

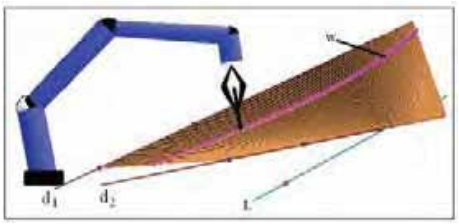


Fig. 26. A laser welding following a 3D-curve w on a ruled surface defined by the directrices d_1 and d_2. The 3D-curve w is the *meet* between the ruled surface and a plane containing the line L.

We know the surface of the object, so we can compute its normal vector in each point using

$$N(s,t) = \left(\frac{\partial \vec{H}(s,t)}{\partial s} \wedge \frac{\partial \vec{H}(s,t)}{\partial t} \right) I_e \tag{100}$$

In surfaces with lower friction, the angle θ is very small, then the value of F tends to its projection over the normal ($F \approx Fn$). To maintain equilibrium, the sum of the forces must be zero $\left(\sum_{i=1}^{3} \|F_n\| N(s_i,t_i) = 0\right)$. This fact restricts the points over the surface in which it can be applied the forces. This number of points is even more reduced if we are confronted with the case when considering the unit normal $\left(\sum_{i=1}^{3} N(s_i,t_i) = 0\right)$ the forces over the object are equal. Additionally, to maintain the equilibrium, it must be accomplished that the sum of the moments is zero

$$\sum_{i=1}^{3} H(s,t) \wedge N(s,t) = 0 \tag{101}$$

The points on the surface having the same directed distance to the center of mass of the object fulfill $H(s, t) \wedge N(s, t) = 0$. Due to the normal in such points crosses the center of mass (C_m), it does not produce any moment. Before determining the external and internal points, we must compute the center of mass as follows

$$C_m = \int_0^1 \int_0^1 \vec{H}(s,t) ds dt \tag{102}$$

Once that C_m is calculated we can establish next constraint

$$(H(s, t) - C_m) \wedge N(s, t) = 0 \tag{103}$$

The values s and t satisfying (103) form a subspace called *grasping space*. They accomplish that the points represented by $H(s, t)$ are critical on the surface (being maximums, minimums or inflections). In this work we will not consider other grasping cases like when they do not utilize extreme points other when friction cones are being considered. This issues will be treated in future work. The equation (103) is hard to fulfill due to the noise, and it is necessary to consider a cone of vectors. So, we introduce an angle called a,

$$\frac{\|(H(s,t) - C_m) \wedge N(s,t)\|}{(H(s,t) - C_m) \cdot N(s,t)} \leq tan(\alpha) \tag{104}$$

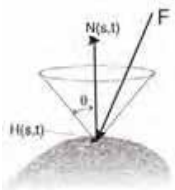


Fig. 27. The friction cone.

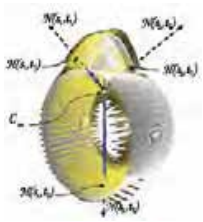


Fig. 28. Object and his normal vectors.

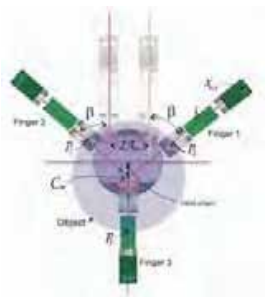


Fig. 29. Object relative position.

We use equation (104) instead of (103), because it allows us to deal with errors or data lost. The constraint imposing that the three forces must be equal is hard to fulfill because it implies that the three points must be symmetric with respect to the mass center. When such points are not present, we can relax the constraint to allow that only two forces are equal in order to fulfill the hand's kinematics equations. Then, the normals $N(s_1, t_1)$ and $N(s_2, t_2)$ must be symmetric with respect to $N(s_3, t_3)$.

$$N(s_3, t_3)N(s_1, t_1)N(s_3, t_3)^{-1} = N(s_2, t_2) \tag{105}$$

Once the three grasping points ($P_1 = H(s_1, t_1)$, $P_2 = H(s_2, t_2)$, $P_3 = H(s_3, t_3)$) are calculated, it is really easy to determine the angles at the joints in each finger. To determine the angle of the spread ($q_4 = \beta$) for example we use

$$\cos\beta = \frac{(p_1 - C_m)\cdot(C_m - p_3)}{|p_1 - c_m|\,|C_m - p_3|},\ \sin\beta = \frac{|(p_1 - C_m)\wedge(C_m - p_3)|}{|p_1 - c_m|\,|C_m - p_3|}, \tag{106}$$

or it is possible to implement a control law which will allow to move the desired finger without the need of solving any kind of inverse kinematics equations [1]. Given the differential kinematics equation

$$\dot{X}'_3 = \begin{bmatrix} \frac{1}{125}X'_3 \cdot L'_2 + \frac{1}{375}X'_3 \cdot L'_3 & -\frac{2}{35}X'_3 \cdot L'_1 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_4, \end{bmatrix} \tag{107}$$

If we want to reach the point $H(s_1, t_1)$, we require that the suitable velocity at the very end of the finger should be proportional to the error at each instance $V_i = -0.7(X'_3 - H(s_1, t_1))$. This velocity is mapped into the phase space by means of using the Jacobian inverse. Here we use simply the pseudo-inverse with $j_1 = \frac{1}{125}X'_3 \cdot L'_2 + \frac{1}{375}X'_3 \cdot L'_3$ and $j_2 = -\frac{2}{35}X'_3 \cdot L'_1$

Applying this control rule, one can move any of the fingers at a desired position above an object, so that an adequate grasp is accomplish.

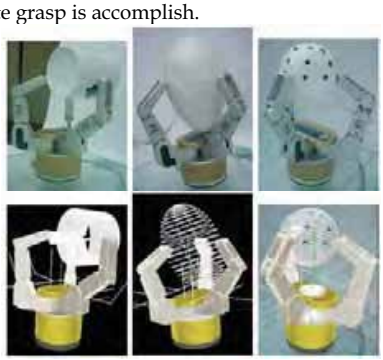


Fig. 30. Grasping some objects.

9. Conclusion

In this chapter the authors have used a single non–standard mathematical framework, the Conformal Geometric Algebra, in order to simplify the set of data structures that we usually use with the traditional methods. The key idea is to define and use a set of products in CGA that will be enough to generate conformal transformations, manifolds as ruled surfaces and develop incidence algebra operations, as well as solve equations and obtain directed distances between different kinds of geometric primitives. Thus, within this approach, all those different mathematical entities and tasks can be done simultaneously, without the necessity of abandoning the system.

Using conformal geometric algebra we even show that it is possible to find three grasping points for each kind of object, based on the intrinsic information of the object. The hand's kinematic and the object structure can be easily related to each other in order to manage a natural and feasible grasping where force equilibrium is always guaranteed. These are only some applications that could show to the robotic and computer vision communities the useful insights and advantages of the CGA, and we invite them to adopt, explore and implement new tasks with this novel framework, expanding its horizon to new possibilities for robots equipped with stereo systems, range data, laser, omnidirectional and odometry.

10. References

[1] Carlos Canudas de Wit, Georges Bastin, Bruno Siciliano. (1996) Theory of Robot Control, Springer.

[2] Bayro-Corrochano E. and Kähler D (2000). Motor Algebra Approach for Computing the kinematics of robot manipulators. *Journal of Robotics Systems*, 17(9):495-516.

[3] Bayro-Corrochano E. 2001. *Geometric Computing for Perception Action Systems*, Springer Verlag, Boston.

[4] Bayro-Corrochano E. 2005. Robot perception and action using conformal geometric algebra. In Handbook of Geometric Computing. Applications in Pattern

Recognition, Computer Vision, Neuralcomputing and Robotics. Eduardo Bayro-Corrochano (Ed.), Springer Verlag, Heidelberg, Chap.13, pp. 405-458.

[5] Bayro-Corrochano, E. and Lasenby, J. 1995. Object modelling and motion analysis using Clifford algebra. In Proceedings of Europe-China Workshop on *Geometric Modeling and Invariants for Computer Vision*, Ed. Roger Mohr and Wu Chengke, Xi'an, China, April 27-29, pp. 143-149.

[6] Bayro-Corrochano, E., Lasenby, J. and Sommer, G. 1996. Geometric Algebra: a framework for computing point and line correspondences and projective structure using n–uncalibrated cameras. *Proceedings of the International Conference on Pattern Recognition (ICPR'96), Vienna, August 1996.*, Vol.I, pp. 393-397.

[7] Laseby J. and Bayro–Corrochano E. 1999. Analysis and Computation of Projective Invariants from Multiple Views in the Geometric Algebra Framework. In Special Issue on Invariants for Pattern Recognition and Classification, ed. M.A. Rodrigues. Int. Journal of Pattern Recognition and Artificial Intelligence, Vol 13, No 8, December 1999, pp. 1105–1121.

[8] Bayro–Corrochano E., Daniilidis K. and Sommer G. 2000. Motor algebra for 3D kinematics. The case of the hand–eye calibration. *Journal of Mathematical Imaging and Vision*, vol. 13, pag. 79-99, 2000.

[9] Hestenes, D. 1966. Space–Time Algebra. *Gordon and Breach*.

[10] Hestenes, D. and Sobczyk, G. 1984. Clifford Algebra to Geometric Calculus: A unified language for mathematics and physics. *D. Reidel*, Dordrecht.

[11] Hestenes D., Li H. and Rockwood A. 2001. New algebraic tools for classical geometry. *In Geometric Computing with Clifford Algebra*, G. Sommer (Ed.). Springer-Verlag, Berlin Heidelberg, Chap.I, pp. 3-23.

[12] Hestenes, D. and Ziegler, R. 1991. Projective Geometry with Clifford Algebra. *Acta Applicandae Mathematicae*, 23, pp. 25–63.

[13] Lasenby, J., Lasenby, A.N., Lasenby, Doran, C.J.L and Fitzgerald, W.J. 1998. 'New geometric methods for computer vision – an application to structure and motion estimation'. *International Journal of Computer Vision*, 26(3), 191-213.

[14] Csurka G. and Faugeras O. Computing three dimensional project invariants from a pair of images using the Grassmann–Cayley algebra. 1998. *Journal of Image and Vision Computing*, 16, pp. 3-12.

[15] Lounesto P. *Clifford Algebra and Spinors*, 2nd ed. Cambridge University Press, Cambridge, UK, 2001.

[16] Needham T. *Visual Complex Analysis*. Oxford University Press. Reprinted 2003.

[17] Rosenhahn B., Perwass C., Sommer G. *Pose estimation of 3D free-form contours*. Technical Report 0207, University Kiel, 2002.

[18] J.M. Selig. *Clifford algebra of points, lines and planes*. South Bank University, School of Computing, Information Technology and Maths, Technical Report SBU–CISM–99–06, 1999.

[19] White N. Geometric applications of the Grassman–Cayley algebra. In J.E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*, CRC Press, Floridam 1997.

3

One Approach to the Fusion of Inertial Navigation and Dynamic Vision

Stevica Graovac
School of Electrical Engineering, University of Belgrade
Serbia

1. Introduction

The integration of different types of navigation systems is frequently used in the automatic motion control systems due to the fact that particular errors existing in anyone of them are usually of different physical natures. The autonomous navigation systems are always preferred from many of reasons and the inertial navigation systems (INS) are traditionally considered as the main representative of this class. The integration of such systems based on the inertial sensors (rate gyros and linear accelerometers) and other navigation systems is very popular nowadays, especially as a combination with global positioning systems [Farrel & Barth, 1999], [Grewal et al., 2001]. The vision based navigation systems (VNS) are also of autonomous type and there is a reasonable intention to make the fusion of these two systems in some type of integrated INS/VNS system. This paper is oriented toward the possibility of fusion of data originated from a strap-down INS on one side, and from a dynamic vision based navigation system (DVNS), on the other. Such an approach offers the wide area of potential applications including the mobile robots and a number of automatically controlled ground, submarine, and aerial vehicles.

The most usual approach in navigation systems integration is of "optimal filter" type (typical INS/VNS examples are given in [Kaminer et al., 1999] and [Roumeliotis et al., 2002]) In such an approach one of the systems is considered as the main one and the other supplies less frequently made measurements (corrupted by the noise, but still considered as the more precise) used in order to estimate in an optimal fashion the navigation states as well as the error parameters of the main system's sensors.

The approach adopted in this paper considers both systems in an equal way. It is based on the weighted averaging of their outputs, allowing some degrees of freedom in this procedure regarding to the actually estimated likelihood of their data. These estimates are based on reasoning related to the physical nature of system errors. The errors characterizing one typical strap-down INS are of slowly varying oscillatory nature and induced by the inaccuracies of inertial sensors. On the other hand, the errors in any VNS are mainly due to a finite resolution of a TV camera, but there is a significant influence of the actual scene structure and visibility conditions, also. In other words, it could be said that the accuracy of an INS is gradually decreasing in time while it is not affected by the fact where the moving object actually is. The accuracy of a DVNS is generally better in all situations where the

recognizable referent landmarks are inside the camera's field of view, occupying larger extent of it. Because the DVNS is based on processing of a sequence of images, the larger relative motion of the landmarks in two consecutive frames is preferable too.
Having in minds these basic features of INS and DVNS, their integration could be considered in two basic ways:

1. An INS is a kind of "master" navigation system while the corrections produced by DVNS are made in time when a moving object is passing the landmarks located around the trajectory. This approach is typically applicable in case of the flight control of remotely piloted vehicles and in the similar "out-door" applications;
2. A VNS is a basic navigation system assuming that the reference scene objects always exist in the scene, while an INS provides the required data related to the absolute motion of an object during the interval between two frames. This approach is oriented toward mobile robot "in-door" applications as well as in case of automatic motion control of the road vehicles.

The next chapter of paper introduces the fundamentals of INS and VNS in the extent required to understand their integration. In Chapter 3. the general case of suggested fusion procedure is presented. A number of particular implementation schemes including reduced set of sensors and/or reduced amount of calculations could be specified based on this general case. The next two chapters consist of the illustrative examples of application: A vision aided INS in the simulated case of remotely piloted vehicle's flight (Chapter 4), [Graovac, 2004]; and a VNS assisted by the acceleration measurements provided by an INS, for the robot control applications (Chapter 5), [Graovac, 2002].
The results related to "out-door" applications are obtained using the full 6-DOF simulation of object's motion and the model of the INS work. The computer-generated images of terrain and ground landmarks have been used during the tests of a DVNS algorithm. These images have been additionally corrupted by noise and textured. The "in-door" applications are illustrated using the laboratory experiments with an educational robot equipped with a TV camera.

2. Basic Concepts of INS and VNS

2.1 Fundamentals of an INS

Estimation of a position of moving object, $\vec{R}_I = [x_I \quad y_I \quad z_I]^T$, relative to an *inertial coordinate frame* (ICF) could be done according to the basic navigation equations as

$$\begin{bmatrix} \dot{V}_{XI} \\ \dot{V}_{YI} \\ \dot{V}_{ZI} \end{bmatrix} = \begin{bmatrix} A_{XI} \\ A_{YI} \\ A_{ZI} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}, \quad \begin{bmatrix} V_{XI}(0) \\ V_{YI}(0) \\ V_{ZI}(0) \end{bmatrix} = \vec{V}_I(0)$$
$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{z}_I \end{bmatrix} = \begin{bmatrix} V_{XI} \\ V_{YI} \\ V_{ZI} \end{bmatrix} = \vec{V}_I, \quad \begin{bmatrix} x_I(0) \\ y_I(0) \\ z_I(0) \end{bmatrix} = \vec{R}_I(0) \tag{1}$$

Acceleration vector in ICF $\vec{A}_I$, on the right hand side, is obtained by transformation of the acceleration measurement vector $\vec{A}_B$. These measurements are made by a triad of linear accelerometers rigidly fixed to the body of moving object and they are referenced to the *body fixed coordinate frame* (BCF):

$$\vec{A}_I = \begin{bmatrix} A_{XI} \\ A_{YI} \\ A_{ZI} \end{bmatrix} = T_{I/B}\vec{A}_B = T_{I/B}\begin{bmatrix} A_{XB} \\ A_{YB} \\ A_{ZB} \end{bmatrix}. \quad (2)$$

Matrix transform $T_{I/B}$ is defined via Euler angles of pitch, yaw, and roll (ϑ, ψ, ϕ) as

$$T_{I/B} = T_3^T(\psi)T_2^T(\vartheta)T_1^T(\phi). \quad (3)$$

where T_1, T_2, T_3, represent the elementary matrix transformations due to rotation around coordinate axes. Actual values of Euler angles are obtained by numerical integration of a set of differential equations:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\vartheta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \omega_{VX} + \tan\vartheta(\omega_{VY}\sin\phi + \omega_{VZ}\cos\phi) \\ \omega_{VY}\cos\phi - \omega_{VZ}\sin\phi \\ \sec\vartheta(\omega_{VY}\sin\phi + \omega_{VZ}\cos\phi) \end{bmatrix}. \quad (4)$$

where $\omega_{VX}, \omega_{VY}, \omega_{VZ}$, represent the projections of the angular velocity of a moving object in ICF onto the axes of BCF. These are measured by the set of rate gyros rigidly fixed to the body of the moving object.

The measurements of linear accelerations and angular velocities in BCF are inaccurate due to slowly varying bias introduced by a number of physical phenomena inside the inertial instruments. These are results of the complex motion of an object (with six degrees of freedom) as well as of sensor imperfections. Sensor signals are additionally corrupted by high frequency measurement noise caused by internal imperfections and by external influences due to the air turbulence, vibrations of vehicle, etc. A specific type of error associated to this particular mechanization (known as a *strap-down inertial navigation system* - SDINS) in case of flying object is a result of rectification introduced by multiplication shown in (2). The elements of matrix $T_{I/B}$ as well as of vector $\vec{A}_B$ include the oscillatory components on natural frequency of body oscillations.

The inertial instruments analyzed here are of medium quality (typically used for the flight stabilization purposes). The numerical data illustrating their accuracy are:

Rate gyros: Bandwidth - 80 Hz;
Bias - 10^0/hour; G-sensitive drift - 10^0/hour/g;
Scale factor error - 1%;
Measurement noise: white, Gaussian, zero mean value, $\sigma = 1^0$/s;

Accelerometers: Bandwidth - 150 Hz;
Bias - 0.1 m/s^2; Resolution - 0.05 m/s^2;
Scale factor error - 1%;
Measurement noise: white, Gaussian, zero mean value, $\sigma = 0.1$ m/s^2;

The accuracy of an INS was analyzed using the complete 6-DOF horizontal flight simulation. As a way of on-line accuracy improvement, the Kalman filter was applied in order to make the filtration of rate gyro signals. This one was based on the linearized dynamic models in pitch and yaw channels. The results of the Kalman filter application in the estimation of pitch rate and pitch angle during the interval of ten seconds of horizontal flight are illustrated in Figure 1.

2.2 Fundamentals of a Dynamic Vision Based Navigation

The linear position of a moving object carrying a TV camera on-board relative to the environmental elements can be reconstructed either from one frame or from a sequence of frames. In the first case, a number of characteristic scene objects' features should be

extracted. The other approach, known as a *dynamic vision*, generally allows usage of a lower number of extracted and tracked features. If some additional information about linear and angular velocities or about angular orientation are known, the task can be radically simplified, allowing the tracking of just one reference object's feature [Frezza et al., 1994], [Menon et al., 1993]. In both cases, if the absolute position of a reference object in ICF is a priori known, the whole method can be interpreted as a reconstruction of the absolute position of a moving object - *visual navigation.*

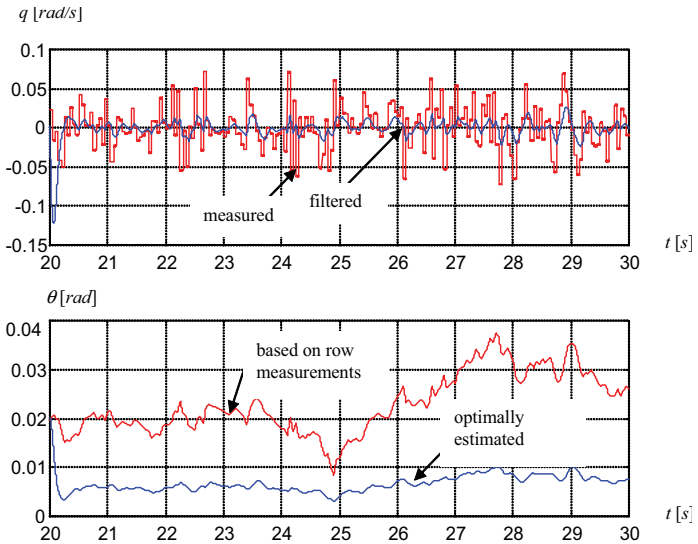


Fig. 1. The effects of application of Kalman filter in the estimation of pitch rate and pitch angle.

The dynamic vision method has been applied in this paper. Supposing that external information about linear and angular velocities of a moving object exists (supplied by an INS), the number of tracked features is reduced to one. In the case of an autonomously guided flying vehicle, typical ground reference objects could be bridges, airport runways, cross-roads, distinguishable buildings, or other large stationary landmarks located at known absolute positions. The task of a VNS consists in extracting the image of landmark itself and after that, calculating the position of some easily recognizable characteristic point (e.g., a corner). If the image of a whole landmark occupies just a small portion of the complete image, it is more reasonable to calculate the position of its centroid instead.
Primary detection (recognition) of a landmark is the most critical step. It is supposed that this task is accomplished using a bank of reference landmark images made separately in advance, under different aspects, from different distances, and under different visibility conditions. Once primary automatic detection has been done, the subsequent recognition is highly simplified. The recognized pattern from the actual image becomes the reference one for the next one, and so on.

In an ideal case, the only required information is regarding the shift of the characteristic point between two consecutive images. The existence of image noise and different other reasons may cause an erroneous calculation of a characteristic point location inside the picture. In order to minimize these effects, a greater number of characteristic points and/or consecutive frames should be analyzed.
Dynamic equations describing the stated approach are the following:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \qquad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{x_2}{x_1} \\ \frac{x_3}{x_1} \end{bmatrix} \quad x_1 \neq 0 \tag{5}$$

State vector $\vec{x} = [x_1 \quad x_2 \quad x_3]^T$ represents the position of a reference point with respect to viewer frame, while coefficient vectors $\vec{v} = [v_1 \quad v_2 \quad v_3]^T$ and $\vec{\omega} = [\omega_1 \quad \omega_2 \quad \omega_3]^T$ represent relative linear and angular velocities, also expressed in the coordinate frame fixed to the moving object. Measured outputs of this nonlinear dynamic system consist of two projections of the reference point onto the image plane (*picture coordinate frame* - PCF) which is perpendicular to the x_1 axis, at a conventional distance $f = 1$ from the origin. If the relative positions are known, the task consists of motion parameter estimation (coefficient vectors identification). If the motion parameters are known, the task is of state estimation nature (structure reconstruction). The second case is considered here.
If in some following moment of time (e.g., in the next frame) the state vector has the value $[x_1 + \Delta x_1 \quad x_2 + \Delta x_2 \quad x_3 + \Delta x_3]^T$, there would exist the shift of an image of reference point in PCF given as

$$\Delta y_1 = f\frac{\Delta x_2}{x_1 + \Delta x_1} - y_1\frac{\Delta x_1}{x_1 + \Delta x_1} \quad \Delta y_2 = f\frac{\Delta x_3}{x_1 + \Delta x_1} - y_2\frac{\Delta x_1}{x_1 + \Delta x_1} \tag{6}$$

The variation of position $\Delta\vec{x} = [\Delta x_1 \quad \Delta x_2 \quad \Delta x_3]$ is produced by the linear motion of a moving object $\Delta\vec{x}_I$ as well as by its change of angular orientation, defined now by matrix transformation $T_{I/B} + \Delta T_{I/B}$ instead of the previous $T_{I/B}$. After appropriate geometrical recalculations it could be shown that the variation of the relative linear position is represented as

$$\Delta\vec{x} = T_C\left[\Delta T_{I/B} T_{I/B}^T\left(T_C^T\vec{x} + \vec{l}\right) - \left(T_{I/B} + \Delta T_{I/B}\right)\Delta\vec{x}_I\right]. \tag{7}$$

where the linear position of the camera relative to the center of gravity of a moving object is denoted as $\vec{l}$, while the angular orientation of a camera relative to the BCF axes is represented via transformation matrix T_C. Both these parameters are known because they are either constant ones or can be measured easily.
After division of both sides of (7), one obtains

$$\begin{bmatrix} \Delta x_1/x_1 \\ \Delta x_2/x_1 \\ \Delta x_3/x_1 \end{bmatrix} = T_C\Delta T_{I/B} T_{I/B}^T T_C^T\begin{bmatrix} 1 \\ y_1/f \\ y_2/f \end{bmatrix} + T_C\Delta T_{I/B} T_{I/B}^T\begin{bmatrix} l_1/x_1 \\ l_2/x_1 \\ l_3/x_1 \end{bmatrix} - T_C\left(T_{I/B} + \Delta T_{I/B}\right)\begin{bmatrix} \Delta x_I/x_1 \\ \Delta y_I/x_1 \\ \Delta z_I/x_1 \end{bmatrix}. \tag{8}$$

Supposing that T_C, f, and $\vec{l}$ are known and that $T_{I/B}$ and $T_{I/B} + \Delta T_{I/B}$ are supplied externally as well as $\Delta\vec{x}_I$ (e.g., by an INS), the task of VNS consists in determining the pair of coordinates in PCF (y_1, y_2) and at least one of the displacement components (6). Combining three scalar equations (8) with the proper one in (6), it is now possible to determine four unknown variables $(\Delta x_1, \Delta x_2, \Delta x_3, x_1)$. Once x_1 is calculated, one can reconstruct the remaining

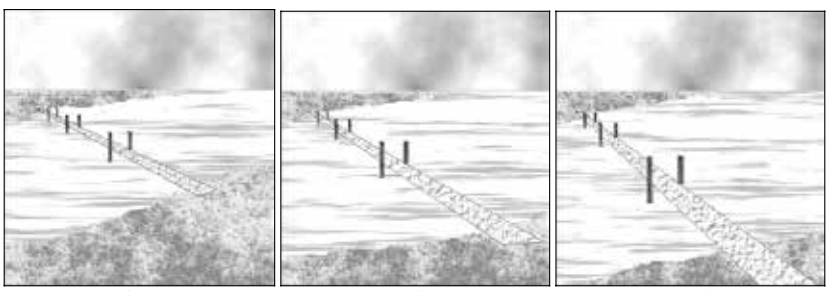
two components of the relative position vector (x_2 and x_3) from the output part of (5). The knowledge of relative position vector $\vec{x}$ and of the absolute position of the characteristic point in ICF is sufficient to reconstruct the absolute position of a moving object.

The crucial problem from an image processing point of view is how to determine the locations of characteristic points in PCF as accurately as possible. There exist a number of methods of distinguishing objects of interest inside the image. Practically all of them are application dependent. Various sequences of image enhancement/digital filtration procedures, segmentation approaches using multilevel gray or binarized picture, morphological filtration algorithms, etc. making these procedures, must be carefully chosen according to the actual scene contents.

Computer generated images of ground landmarks are used throughout this work. A relatively simple correlation technique consisting of matching of actual image contents and a reference pattern has appeared as the most robust one. It is based on calculation of a sum of absolute differences of light intensities inside the window scanning across the image. The feature has been defined as the light intensity distribution inside the rectangular window of $n_T = n_X n_Y$ pixels around the characteristic point. The displacement of characteristic point $(\Delta y_1, \Delta y_2)$ is calculated by maximizing the similarity of the actual image and the previous one, i.e., minimizing the criterion given as a sum of absolute values of differences (MAD) of light intensity I_N:

$$L = \frac{1}{n_T} \sum \left| I_N(y_1 + \Delta y_1, y_2 + \Delta y_2) - I_{N-1}(y_1, y_2) \right| . \tag{9}$$

The efficiency of the stated algorithm will be illustrated using the sequence of textured images of a bridge. The nearest holder has been used as a reference object while its crossing with the left edge of a runway was selected as a characteristic point (Figure 2.) Figure 3. illustrates matching results obtained for the multiple level gray and binarized pictures. The brightest points inside the black window are pointing to the locations of maximal similarity. The reference window was of dimensions 25 X 25 pixels. Higher sharpness of candidate area in case (a) suggests that one could expect better results if the multiple level gray pictures were used.

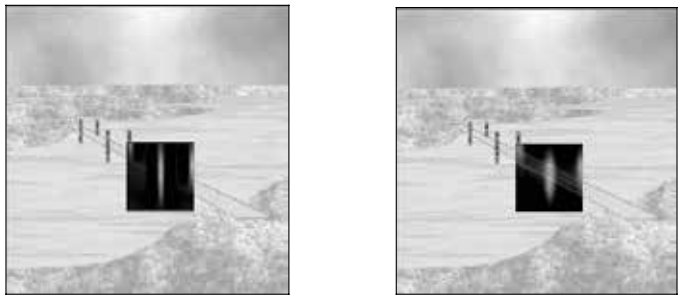


1) **2)** **3)**

Fig. 2. Sequence of textured images of a bridge used as a landmark.

When the sequence of frames shown in Figure 2. was used for navigation purposes, the results given in Table 1. have been obtained. It is supposed that the angular position of the camera is constant during the observed time period (yaw angle, 12°, pitch angle, -5°, roll

angle, 0°). The linear velocities of the moving object are exactly known and constant also (V_X = 500 m/s, V_Y = 100 m/s, and V_Z = 0 m/s). Position estimates given in Table 1. were obtained using the pairs of frames 1-2, 1-3, and 2-3.



(a) **(b)**

Fig. 3. Extraction of a characteristic point inside: (a) multiple level gray and (b) binarized picture.

	Position in frame 1.			*Position in frame 2.*	
	Exact	*Estimate*		*Exact*	*Estimate*
		Based on pair1-2	*Based on pair1-3*		*Based on pair2-3*
X (m)	*600*	*606*	*535*	*400*	*357*
Y (m)	*200*	*199*	*187*	*160*	*152*
Z (m)	*100*	*100*	*94*	*100*	*96*

Table 1. Moving object position estimation using dynamic vision from sequence shown in Figure 2.

2.3 Fundamentals of an Autonomous VNS

The fundamental step in an autonomous VNS based on the processing of just one image consists of the calculation of the relative distance and angular orientation of a camera relative to the reference object (landmark) located in a horizontal plane of ICF (Figure 4.) [Kanatani, 1993].

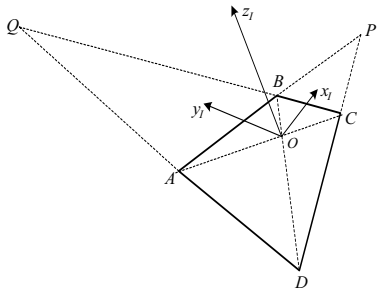


Fig. 4. Reference object in the field of view of TV camera.

Supposing a landmark of rectangular shape of known dimensions and the cross-section of its diagonals as a reference point adopted as the origin of ICF, the position of the camera relative to this point could be obtained from just one image by the following algorithm:

1. Calculate the normalized vectors for all four projections in PCF of corners *A, B, C,* and *D*. Each one of these *m-vectors* is a vector representing projection of the appropriate point $\vec{y}_i = [1 \quad y_{i1} \quad y_{i2}]^T$, divided by its Euclidean norm.
2. Calculate the normalized vectors for all four projections of the edges *AB, BC, CD, DA*, as the normalized cross-products of m-vectors above. Elements of these *n-vectors* specify the equations of image lines encompassing the projections of the appropriate pair of corners.
3. Calculate the m-vectors of *vanishing points P* and *Q*, as cross-products of n-vectors above, representing the projections of parallel edges of a reference object.
1. Calculate the n-vectors of diagonals *AC* and *BD* as in case of image lines representing the edges (2).
2. Calculate the m-vector $\vec{m}_O$ of the point at the cross-section of diagonals *O*, as in case of vanishing points (3).
3. Choosing any one of the corners as the point at known distance *d* from the reference point *O*, calculate the *scene depth:*

$$|\vec{R}| = \frac{\vec{m}_I \vec{e}_{I3} \cdot d}{\left\| \vec{m}_O \vec{e}_{I3} \cdot \vec{m}_I - \vec{m}_I \vec{e}_{I3} \cdot \vec{m}_O \right\|}. \tag{10}$$

representing the distance between camera's sensitive element and the reference point *O*. The m-vectors $\vec{m}_O$ and $\vec{m}_I$ are related to the reference point *O* and the chosen corner *I*. The m-vectors of vanishing points $\vec{m}_P$ and $\vec{m}_Q$ are at the same time the basis vectors of reference coordinate frame with its origin at *O*. The ort of direction perpendicular to the plane containing the reference rectangle is obtained as $\vec{e}_{I3} = \vec{m}_P \times \vec{m}_Q$.

4. Calculate the position of a camera relative to the reference point as

$$\vec{R} = -|R| \cdot T_O \vec{m}_O . \tag{11}$$

where $T_O = [\vec{e}_{I1} \quad \vec{e}_{I2} \quad \vec{e}_{I3}]^T = [\vec{m}_P \quad \vec{m}_Q \quad \vec{m}_P \times \vec{m}_Q]^T$ represents the transformation matrix due to rotation of the frame fixed to a camera (BCF) in respect to the coordinate frame fixed to the reference object (ICF).

The above explained algorithm reflects just a *geometrical* aspect of a problem. Much more computational efforts are associated with the *image processing* aspect, i.e., with the problem how to distinguish the reference object and its characteristic points from the actual contents of an image. It should be noted that the final effect of this process consists of some deteriorated accuracy in the determination of image coordinates of the reference points. A lot of *scene dependent* conditions affect the extraction as well as some system parameters (image noise, level quantization, space resolution). An image noise is dominantly associated to TV camera itself and it is usually considered as a zero-mean, Gaussian, white noise with specified standard deviation (expressed in number of intensity quanta). The later two systematic sources of inaccuracy are due to the process of image digitization. While the effects of the finite word length of a video A/D converter are of the same nature as the effects of image noise, the finite space resolution has the direct influence onto the final accuracy of position estimation, even when the reference object is ideally extracted. The

finite number of picture elements, *pixels*, along the horizontal and vertical directions inside the image, makes the limits for the final accuracy. However, this effect is strongly coupled with the size of camera's field of view and the actual distance between a camera and the reference point. The error expressed in pixels has its angular and linear equivalents in dependence on these parameters.
The redundancy in geometrical computations is generally suggested for the VNS accuracy improvement. Instead of a theoretically minimal number of considered points required for some calculation, the number of points is increased and some appropriate optimization procedure is usually used in order to filter out the effects of noise in determination of a location of any particular characteristic point. For example, instead of starting with the determination of the positions of corners in above mentioned algorithm, one can start with the detection of edges and find the equations of image lines by the best fitting procedure considering the whole set of edge points (not by using just two of them as before). Now the m-vectors of corners are obtained as cross-products of the appropriate n-vectors and the remainder of algorithm is the same. Similarly, the final accuracy can be significantly improved if one repeats the explained algorithm using different corners as reference ones and finds the weighted average of results. All these methods used for the accuracy improvement increase the overall computational effort. Therefore, it is of a great importance to find the way how to obtain the same or better accuracy using a less number of considered points or the less complex image processing algorithms.

3. Principles of Data Fusion

Main conclusions related to the quality of information about linear and angular positions of a moving object relative to ICF, obtained by INS and VNS separately, are the following:
The accuracy of the SDINS based algorithm

- depends on a slowly varying bias (drift) and a measurement noise of inertial sensors;
- decreases in time due to cumulative effect produced by these errors;
- depends on errors in initial condition estimation (angular and linear positions and velocities);
- could be improved by recursive optimal state estimation; and
- is affected by slowly varying bias introduced by rectification.

The accuracy of the VNS based algorithm

- depends on the reference object's visibility conditions;
- depends on TV image noise as well as on quantization made by video signal digitization;
- depends on the relative size of a reference object inside the field of view (increases while the moving object approaches the reference one);
- depends on the shift(s) of the characteristic point(s) between two consecutive frames and increases in the case of larger ones; and
- could be improved by increasing the number of tracked points and/or analyzed frames.

Having in mind the fact that the error sources inside these two systems are different and independent, the possibility of their fusion is considered. The combined algorithm of linear and angular position estimation is based on a suitable definition of a criterion specifying the likelihood of partial estimations.

It is supposed in the general case that both algorithms are active simultaneously and autonomously. *Autonomous* estimations from one algorithm are being passed to another one in order to obtain new (*assisted*) estimations. Resultant estimation on the level of a combined algorithm is always obtained as the weighted average value of separate ones. The weighting factors are calculated according to the adopted criteria about partial estimation likelihood.
The following formalism has been adopted:

- the transformation matrix connecting BCF and ICF, generally noted as $T_{I/B}$ will be represented just as T_I;
- all vectors representing angular and linear velocities and linear positions are relative to ICF;
- lower indexes I and V are referencing the estimated variables to the estimation originating system (I - inertial, V - visual);
- autonomously estimated variables are noted by upper index ' ;
- upper index " stands for the estimate based on the information obtained from other system (assisted one);

The general procedure consists of the following steps:

1. SDINS generates its autonomous estimates of angular rate vector $\vec{\omega}_I'$, transformation matrix T_I', linear velocity vector $\vec{V}_I'$, and space position $\vec{x}_I'$.
2. Based on $\vec{x}_I'$, T_I', and a priori known position of a reference object in ICF, VNS searches the field of view inside the expected region. It finds the image of the reference object and calculates the coordinates of characteristic points in PCF.
3. Adopting $\vec{x}_I'$ as a priori known initial position estimation (scene structure), VNS identifies from the sequence of frames the angular rate vector $\vec{\omega}_V''$ and linear velocity vector $\vec{V}_V''$.
4. Adopting the estimations $\vec{\omega}_I'$ and $\vec{V}_I'$ as accurate ones and on the basis of landmark's image position measurements in the sequence of frames, VNS estimates the position vector $\vec{x}_V''$.
5. VNS autonomously generates its estimation of $\vec{x}_V'$ and T_V' by tracking of more characteristic points in one frame.
6. INS takes the estimation T_V' from VNS and applying it onto the vector of measured accelerations in BCF and by double integration calculates the new estimation of position $\vec{x}_I''$.
7. Inside INS, the resultant moving object position estimate is obtained as
$$\vec{x}_{IR} = K_{II}\vec{x}_I' + (1 - K_{II})\vec{x}_I''. \tag{12}$$
8. Inside VNS, the resultant moving object position estimate is obtained as
$$\vec{x}_{VR} = K_{VV}\vec{x}_V' + (1 - K_{VV})\vec{x}_V''. \tag{13}$$
9. The resultant estimates on the level of a combined algorithm are obtained as
$$\begin{aligned} \vec{x} &= K_I\vec{x}_{IR} + (1 - K_I)\vec{x}_{VR} \\ \vec{\omega} &= K_I\vec{\omega}_I' + (1 - K_I)\vec{\omega}_V'' \\ \vec{V} &= K_I\vec{V}_I' + (1 - K_I)\vec{V}_V'' \\ T &= K_I T_I' + (1 - K_I)T_V' \end{aligned}. \tag{14}$$

One can note that inside the VNS part, autonomous estimation of linear and angular velocities has been omitted, supposing that in "out-door" applications a lot of points and frames must be used for this purpose, introducing a computational burden this way. For "in-door" applications, where it is possible to process just a relatively small number of points, there exists the reason to produce these estimates also. Based on this additional information, the calculation in the INS part could be extended in order to include T_I'' (based on $\vec{\omega}_V'$) and another calculation of $\vec{x}_I$ (based on $\vec{V}_V'$), increasing the total number of position estimates in INS from two to four.

Besides the general limitations regarding the computing time required to implement this combined algorithm, as the most important step, one should analyze the likelihood of partial estimates. As the practical measure of decreasing of computing time, one can always consider the possibility to exclude some of the steps 1. - 9. completely, especially if it is a priori possible to predict that their results would be of insufficient accuracy.

Generally speaking, the choice of weighting factors in (14) is a critical step in the whole combined algorithm. It is possible by an improper choice to obtain the resultant estimates worse than in the case of application of a separate algorithm (better among two). While the specification of weighting factor variations is in large extent application dependent, there exists the interest to define some basic principles and adaptation mechanisms, having in mind the nature of errors in INS and VNS. In the particular case of extremely short working time of inertial sensors and good visibility conditions, one can specify constant values of weighting factors, but in the general case it is more adequate to assume that accuracy of separate estimates is changeable and that values of weighting factors should be adapted accordingly.

The first principle regards to the general conclusions about the overall accuracy of INS and VNS stated above. While the accuracy of position estimates in INS is always decreasing in time, the evaluation of accuracy of results obtained by VNS is more complex. As the first, there exists the possibility that this algorithm could not be applied at all (e.g., when a reference object is outside the field of view, for the case when it could not be distinguished uniquely between a few potential ones, etc.) This situation is not possible in an INS except in case that some of the inertial sensors completely failed. As a second, assuming that moving object equipped by a TV camera approaches the reference one in time, it is realistic to expect that overall accuracy increases. However, this increasing is not a continuous one. While by approaching, the relative errors really are smaller for the fixed value of absolute error in determination of characteristic points' coordinates in PCF (expressed in pixels), one can not guarantee that the last ones could not be larger in the next frame. For example, partial occluding of a reference object after it has been detected and tracked in a number of frames could deteriorate the accuracy in large extent. According to this reason, it is assumed that the accuracy of VNS estimates increases in time linearly, from a minimal to maximal one. Simultaneously, using the mechanism of monitoring of VNS estimates, the basic principle is corrected occasionally.

There follows the procedure of adaptation of weighting factor K_I :

1. Before the reliable detection of a reference object inside VNS it is set to: $K_I = 1$.
2. Reliable detection criterion is based on similarity measure between the actual scene contents and the memorized reference pattern. Based on the estimated linear position and transformation matrix obtained by INS, the part of algorithm belonging to VNS makes the required rescaling and rotating of memorized pattern.

It is required that in the window of specified dimension, the value of functional L in the MAD algorithm (Eq. 9) should be lower than the specified threshold value, in N frames continuously. Let's assume that inside the window of dimension 25 × 25, the threshold value is L_{max} = 5. In this case, if the average absolute difference in light intensities for 625 observed points is less than approximately two quanta, it is supposed that the reference object is detected. After that it is assumed that both autonomous and assisted estimations could be calculated in VNS reliably.

3. VNS starts with tracking of characteristic points of a reference object. The autonomous estimates $\vec{x}'_V$ and T'_V as well as the assisted ones ($\vec{x}''_V$, $\vec{\omega}''_V$, and $\vec{V}''_V$) are being calculated. The scene content inside the window around the characteristic point becomes the reference pattern for the next frame analysis.
4. After reliable detection of the reference object in VNS, weighting factor K_I starts to decrease linearly in time.
5. The minimal value of this factor K_{Imin} should be specified for any particular application.
6. The time required for K_I to reach the minimal value is calculated in the VNS part at the beginning of tracking. This calculation based on initial data obtained by INS (position, angular and linear velocities) gives the estimated time of existence of a reference object inside the TV camera's field of view. It is assumed that the moving object approaches the reference one.
7. The similarity measure L is monitored during the tracking of characteristic points. If in the actual frame this one is larger than in the previous one, weighting factor K_I holds the previous value.
8. If at any of the frames the similarity measure is worse than the critical one used as a criterion of reliable detection ($L > L_{max}$), weighting factor K_I is to be reset to value 1 and a detection procedure of a whole reference object is repeated again (back to step 1.).
9. If the conditions of losing the reference object from the field of view are the regular ones (the estimated time of existence has been expired), weighting factor K_I is also reset to the value 1, but the new acquisition of the actual reference object is not going to start. VNS starts a state of waiting on the new reference object recognition.

As it is obvious, parameters like window dimensions, similarity measure threshold, L_{max}, number of frames used for reliable detection, and minimal value of weighting factor for INS estimates (it is at the same time the maximal value of weighting factor for VNS estimates), should be considered as the free ones. They should be carefully specified for the particular application.

It can be noted that in the described procedure the INS part is considered as "master" while the VNS algorithm autonomously evaluates the accuracy of its own estimates at the primary level (correlation between the actual window contents and the actual reference pattern). The described procedure regards to the most general case: flight of a moving object along the specified trajectory with the existence of a number of reference ground objects as landmarks. All cases where the reference object is always present in the field of view are considered as particular ones. For these cases it is reasonable to consider the maximal value of weighting factor K_I as free parameter also (to adopt it as less than 1).

In the expressions (12) and (13) there are the weighting factors affecting autonomous and assisted estimates inside both parts of the algorithm. The way of their adaptation should be considered also.

The position estimate in INS is expressed via (12). If the detection of a reference object in VNS does not exist ($K_I = 1$), the only available estimate is the autonomous one ($K_{II} = 1$). As the assisted estimation $\vec{x}_I''$ is obtained applying the transformation matrix T_V' onto the accelerometers' outputs directly, validity of this matrix affects the weighting factor for the assisted estimate. The simple validity measure for the transformation matrix is its deviation from orthogonality. If its determinant $\det T_V'$ is outside the specified tolerances (e.g., 1 ± 0.1), it is reasonable to assume that this estimate is not valid and to reject it ($K_{II} = 1$). If T_V' is approximately orthogonal, two cases are possible in general. In the first one, the estimates $\vec{x}_I'$ and $\vec{x}_I''$ are close (e.g., differences between all coordinates are below the specified values). According to that, K_{II} decreases linearly from the value of 1 to 0.5 minimally, depending on $\det T_V'$. The minimal value (equal weighting of autonomous and assisted estimations) is approached when $\det T_V'$ is inside the small tolerances (e.g., 1 ± 0.01). The second possibility is that T_V' is approximately orthogonal, but the differences between estimations $\vec{x}_I'$ and $\vec{x}_I''$ are outside the specified tolerances. This is the case when one can assume that the likelihood of an assisted estimate is higher than that of the autonomous one. As a result of this K_{II} should also decrease as previously, but now from maximal value of 1 to $K_{IImin} < 0.5$, depending again on the value of $\det T_v'$. Whatever weight is assigned to $\vec{x}_I''$ one should note that the resultant position estimate in INS is always dominantly dictated by inertial instruments. At this step, just position increments are being calculated in the combined manner, while the initial conditions are determined by previous results of INS only.

The expression (13) defines the fusion of data on the VNS level. The basic condition for its application is the reliably detected reference object ($K_I < 1$). If the autonomous VNS estimate of angular position is bad (i.e., $\det T_V'$ is outside the specified tolerances) the autonomous linear position estimate $\vec{x}_V'$ is going to be bad also, and accordingly, weighting factor K_{VV} takes the minimal value close to zero. While $\det T_V'$ approaches the value 1, this weighting factor increases linearly up to the maximal value K_{VVmax}. It should be noted again that whatever weighting is assigned to $\vec{x}_V''$, calculations in the VNS part are basically dependent on the angular position estimate. Possible invalid estimates $\vec{x}_I'$ due to accumulated inertial sensors' errors are of no importance here, having in mind that calculation of $\vec{x}_V''$ is based on actual filtered signals $\vec{\omega}_I'$ and $\vec{V}_I'$.

4. Simulation Results – Vision Aided INS

The part of an airport runway was considered as a reference ground landmark in vicinity of nominal trajectory. The middle point of a nearer edge is located at known position $\vec{x}_{RO} = [50000 \quad 500 \quad 0]^T$ in ICF. The interval from t = 77s to t = 83s, in which the existence of a landmark could be expected inside the field of view, is predicted on the basis of known fixed parameters of an electro-optical system: angular position of camera relative to BCF ($[\varepsilon_1 \quad \varepsilon_2 \quad \varepsilon_3]^T = [10^0 \quad -5^0 \quad 0^0]^T$, focal length, f = 1, field of view width, $\varepsilon_{\max} = 15^0$). Acquisition of

a reference object is done during the first second of this interval (ten frames). The assumed image of a landmark is obtained using erroneous data from INS. These data are obtained after optimal filtration of the rate gyro signals made inside INS and the linear position errors on observed interval are quantitatively represented in Figure 5.

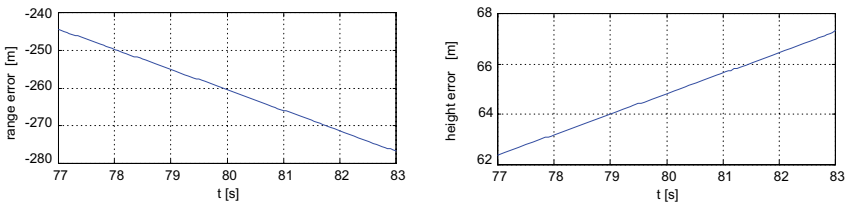


Fig. 5. Position errors of unaided INS during correction interval.

The quality of pitch angle estimation made by INS is illustrated in Figure 6.

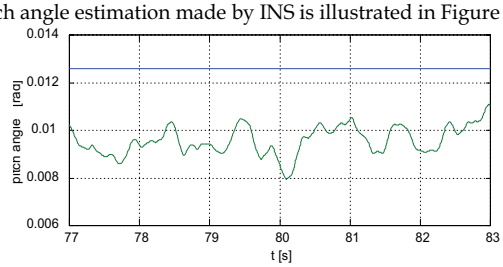


Fig. 6. The exact and estimated values of pitch angle during the correction interval.

Since the estimated values of pitch angle have been obtained using one possible realization of simulated sensor noise, their generation for the purposes of combined navigation method illustration is made using statistical properties of observed time history. The parameters calculated from Figure 6. are: mean value, $\hat{\vartheta}_{sr}$ = 9.6 mrad, and standard deviation, σ_{ϑ} = 0.59 mrad. Approximately the same standard deviations have been obtained for other two Euler angles, while their mean vales are equal to zero. The transformation matrix T_O is generated using the Euler angle estimates generated stochastically.

The fact that in the first frame there is a difference between the actual image of reference object and the expected one is illustrated in Figure 7.

The window of dimensions 25×25 pixels around the lower left corner of a runway was used as a reference pattern. During the first ten frames inside a correction interval, maximum similarity measures formed as a sum of light intensity absolute differences are shown in Figure 8.

Minimum values of MAD criterion are lower than the adopted threshold value of L_{max} = 5 in five consecutive frames at t = 78 s. It is assumed that the acquisition phase was finished at this time. The next step consists in calculation of the expected time of presence of the characteristic point inside the field of view. It is equal to 5 s for this particular example (under the assumption that linear and angular velocities would be constant). Figure 9.

illustrates the expected contents of the field of view in the interval of next five seconds after the approval of acquisition.

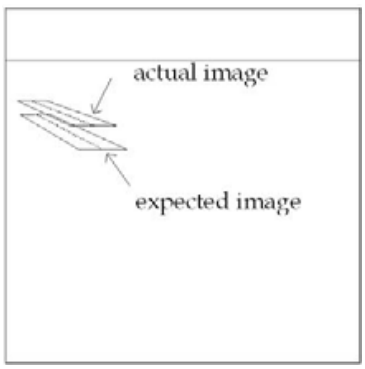


Fig. 7. The expected and actual images of a reference object at the beginning of correction interval.

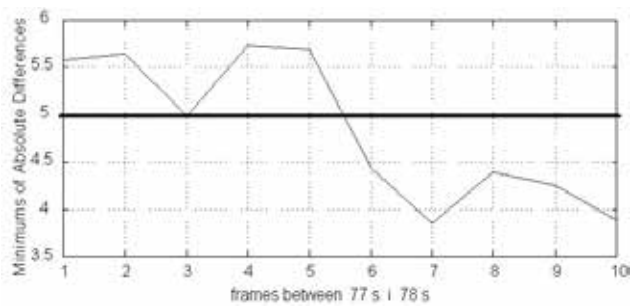


Fig. 8. Maximums of similarity measures during the first second of acquisition interval.

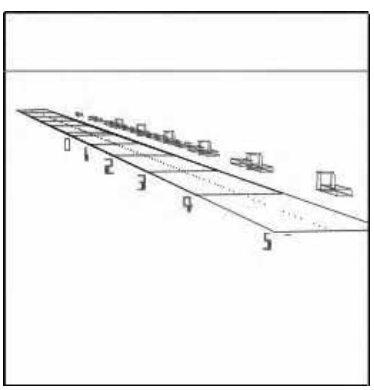


Fig. 9. Expected motion of a reference object during the interval of correction.

According to this, the variation of weighting factor K_I is defined as

$$K_I = 1 - \frac{1-0.1}{5}(t-78). \tag{15}$$

The effects of the combined algorithm will be illustrated in the final portion of the correction interval. The last five frames are shown in Figure 10.

t = 82.2 s t = 82.4 s t = 82.6 s t = 82.8 s t = 83.0 s

Fig. 10. Sequence of runway images at the end of correction interval.

The exact and estimated positions of a moving object are given in Table 2.

	t = 82.2 s		t = 82.4 s		t = 82.6 s		t = 82.8 s	
	x [m]	z [m]	x [m]	z [m]	x [m]	z [m]	x [m]	z [m]
Exact	*48022*	*200*	*48135*	*200*	*48250*	*200*	*48364*	*200*
INS	*48295*	*267*	*48409*	*267*	*48525*	*267*	*48640*	*267*
VNS	*47997*	*205*	*48204*	*199*	*48236*	*203*	*48362*	*200*
Comb.	*48070*	*220*	*48247*	*213*	*48286*	*214*	*48400*	*209*

Table 2. The exact and estimated positions of an object in ICF.

The same results are shown graphically in Figure 11.

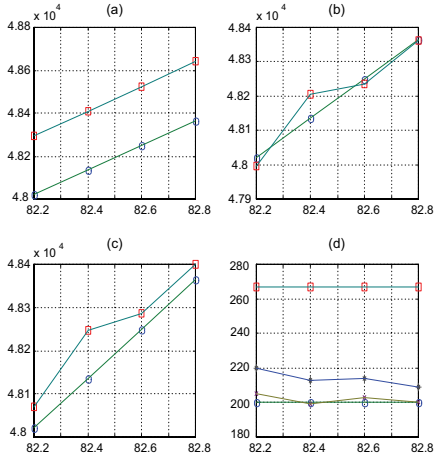


Fig. 11. Comparison of position estimations at the end of correction interval: (a) - range, INS versus exact, (b) - range, VNS versus exact, (c) - range, combined versus exact, (d) - height, exact (circle), INS (square), VNS (cross), combined (star).

As it is obvious from (a), the error in INS is very slowly increasing. From (b) one can see very accurate estimates of VNS while quantization errors as well as errors in determination of characteristic point location occasionally introduce some larger position errors (t = 82.4 s). Because of maximal weighting of VNS estimations at the end of the correction interval, beneficial effects of combined algorithm are obvious from (c). Analyzing the results of height estimation (d) one can conclude that the VNS algorithm is extremely accurate, making the results of combined algorithm satisfactory also (suggesting that it is possible to assume an even lower minimum value than K_{Imin} = 0.1).

5. Simulation Results – VNS Assisted by INS

5.1 The Definition of Navigation Tasks

Two particular "in door" navigation tasks have been specified in order to compare the results of application of an autonomous VNS and a dynamic vision navigation algorithm representing a VNS assisted by the acceleration measurements produced by reduced INS:

(Task A): Moving object has got three linear degrees of freedom. In forward direction it moves with the constant velocity (10 m/s). The camera is mounted as forward looking. The initial position of object is assumed as 5 m out of navigation line in lateral direction and 5 m above. The navigation line consists of sequence of rectangular shapes located in the ground plane. The dimension of these landmarks is 1.5 m X 0.15 m with 1.5 m distance between them. The linear velocities in lateral (V_y) and vertical (V_z) directions are controlled and limited to 5 m/s maximally. As a result of a navigation algorithm the actual commanded values of these velocities are calculated as proportional to the estimated distance from center-line of navigation line. The task consists in approaching the navigation line in lateral direction and following of it further. At the same time, a moving object should approach the ground plane (camera at the fixed distance of 1 m above) and continue to move at this height.

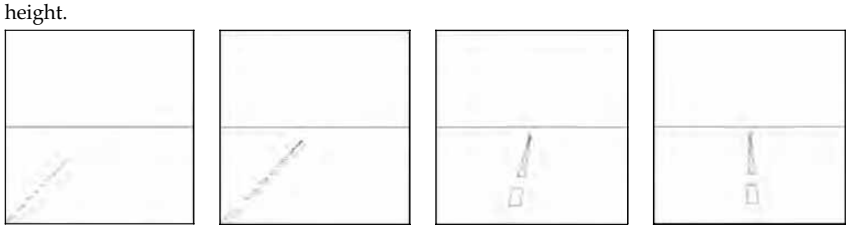


Fig. 12. Sequence of navigation line views **(A).**

The contents of camera's field of view are computer generated. Figure 12. illustrates the sequence of frames generated at 0.6 s inter-frame interval assuming the ideal work of the VNS algorithm (noise-free images, infinite resolution, without camera vibrations).

(Task B): This task consists of the planar motion control - tracking and following of the curved navigation line. This line consists from two connected circular segments. Their dimensions have been adopted according to the available equipment (see 5.3). Eight approximately rectangular black landmarks of known dimensions are equidistantly placed along the line. A TV camera is forward looking and mounted at 50 mm above the ground plane (Figure 13.)

This navigation task consists in the following:

1) Determine the distance from a camera to the reference point belonging to the landmark;
2) Determine the orientation of the landmark in the ground plane; 3) Generate the commands for the translation up to the point above the tracked one and for the rotation of a camera around the vertical axis in order to follow the sequence of landmarks.

The autonomous VNS algorithm uses two nearer marker corners for the calculation of scene depth. The determination of marker orientation requires that all four marker's corners should be visible in one frame. If the analyzed marker is partially visible, the algorithm automatically rejects it and analyzes the next one.

In the algorithm based on dynamic vision the linear velocity components are obtained via integration of the accelerometers' signals. In comparison to the autonomous VNS algorithm, the advantage is in the fact that it is enough to track just one landmark corner here. On the other hand, at least two consecutive frames containing the tracked point are required in order to estimate the distance. As a result of this, any selection of the reference point must be preceded by the prediction whether this one would be visible in the next frame. The other disadvantage resulting from the same reason consists of the fact that the navigation should be initialized by a priori known motion in order to acquire the initial information about marker position and after that to adjust the control action according to it. In other words, in comparison to autonomous VNS there will be always "one frame delay" whenever the tracked point is changed.

Additionally, while the autonomous VNS algorithm has the ability of autonomous estimation of landmark's orientation (needed in order to improve the conditions of landmark distinguishing), there is no such possibility in the case of a combined algorithm. As a result of this, the command for rotational motion is generated here on the basis of a ratio of velocity components and the camera is oriented in direction of a velocity vector (which is directed toward the tracked point - *direct guidance*).

5.2 Simulation Methodology

The estimated positions of characteristic points in the image plane are obtained as the ideal ones additionally corrupted by white, Gaussian, zero-mean noise with standard deviation *sigma* = 1 pixel. The camera vibrations are also simulated as a noisy process. The pitch, yaw, and roll angles are simulated as white, Gaussian, zero-mean noise with standard deviation *sigma* = 1^0.

For the task **(B)** the algorithm explained in Section 4.3 is applied always for the nearest, completely visible element of a navigation line (four corner points). The algorithm presented in Section 4.2 is based on the tracking of one of the corners of the same part of navigation line as in the previous case.

5.3 Experimental Rig Set-up

The experimental acquisition of images is done using the educational robot "Kestrel" with three degrees of freedom in linear and one degree of freedom in rotational motion, driven by step motors, equipped with CCD TV camera (KAPPA CF16/4 P) and using the frame grabber with resolution of 512 X 512 pixels ("Targa")

A photograph of the experimental rig set-up is shown in Figure 13.

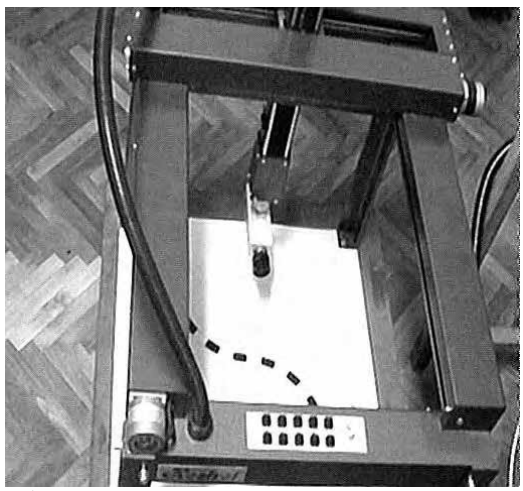


Fig. 13. Experimental rig set-up.

The initial sequence of frames produced by a TV camera during the navigation task **(B)** is presented in Figure 14.

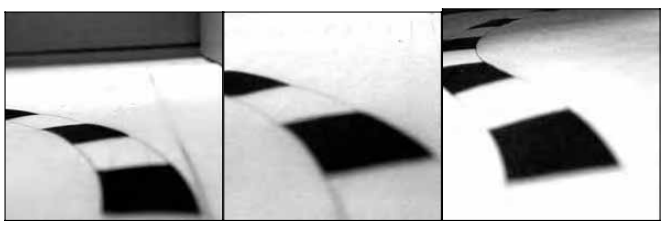
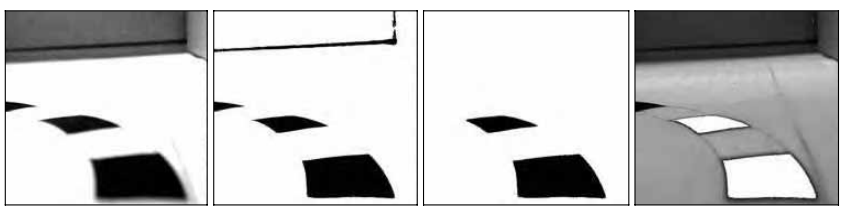


Fig. 14. Experimental sequence of frames produced during the navigation task **(B).**

The processing phases for the first frame of Figure 15. are illustrated in Figure 15.



(a) (b) (c) (d)

Fig. 15. Results of a processing of frame (1) from Figure 14.

(a) Result of a preprocessing (equalization of image histogram, noise elimination, sharpening); (b) Result of a segmentation based on intensity level; (c) Result of a morphological filtration (cleaning of edges, erosion, dilatation); (d) Inverted result of the

image processing superimposed to the original image in order to notice the differences between the actual landmark and determined one.

5.4 Results

The results obtained by the application of algorithms of the autonomous VNS and with assistance of the linear acceleration measurements (dynamic vision) for the task **(A)** under the methodology defined at 5.2 are shown in Figures 16. and 17. The dashed lines represent in both cases the ideal trajectories obtained for the ideal position estimates (exactly calculated positions of characteristic points, infinite space resolution). The solid lines illustrate the trajectories obtained by the simulated application of described algorithms (through circular symbols representing the actual positions of moving object). The square symbols are used to mark the positions estimated by the navigation algorithms.

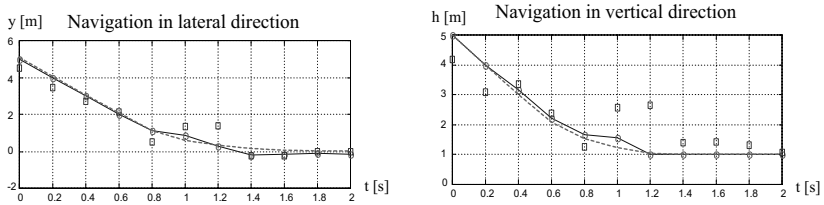


Fig. 16. Simulation results obtained using an autonomous VNS algorithm **(A).**

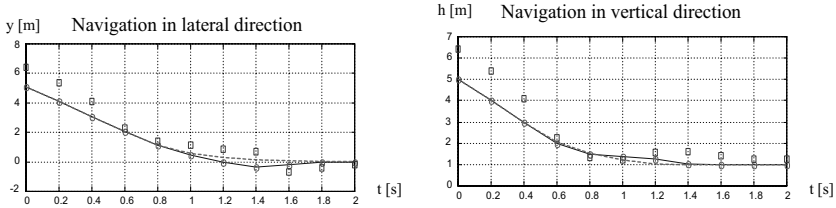


Fig. 17. Simulation results obtained using the dynamic vision algorithm **(A).**

Comparing these results one can conclude that the trajectories obtained in lateral directions are of similar quality. The advantage of the dynamic vision algorithm is noticeable in the case of navigation in vertical direction. Superior quality of position estimates made that the reaching of the final height of 1 m is much "softer" in comparison with the result obtained via autonomous VNS (landing velocity of 1 m/s in comparison to 3.1 m/s). The inherent disadvantage of a dynamic vision algorithm consisting of "one frame delay" is slightly visible. Its repercussion is the requirement to include in image processing algorithm the additional prediction whether the considered marker would be present in the next frame in the field of view. In spite of this, the image processing part of algorithm remains here less time consuming in the comparison to the autonomous VNS algorithm.

The results illustrating navigation task (**B**) for both autonomous VNS algorithm and combined one are as follows.

Figure 18. illustrates very good results in the estimation of landmarks' angular orientations obtained via autonomous VNS algorithm. The solid line through the circular symbols

represents the actual average values of orientation angle while the dashed line through the square symbols represents the estimated ones.

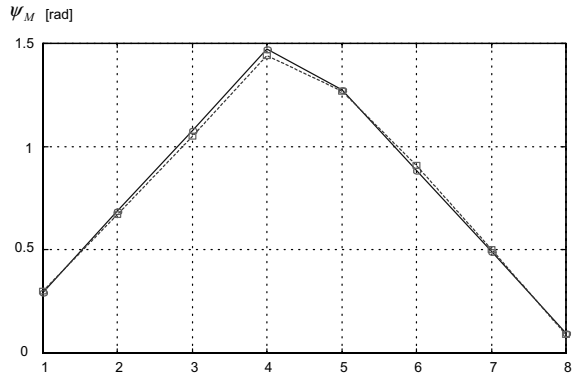


Fig. 18. Estimates of landmarks' angular orientations obtained by the autonomous VNS algorithm **(B).**

As a result of these accurate estimates, the landmarks are always visible in approximately the same way in a camera's field of view. The trajectory obtained as a result of navigation line following in this case is presented in Figure 19. The initial location was 60 mm behind the coordinate origin. The autonomous VNS algorithm generates the commands positioning the moving object at the location behind the next landmark, at the same distance and along the direction of its orientation. Circular symbols represent the positions of tracked corners of eight landmarks while the square ones represent the consecutive positions of a moving object.

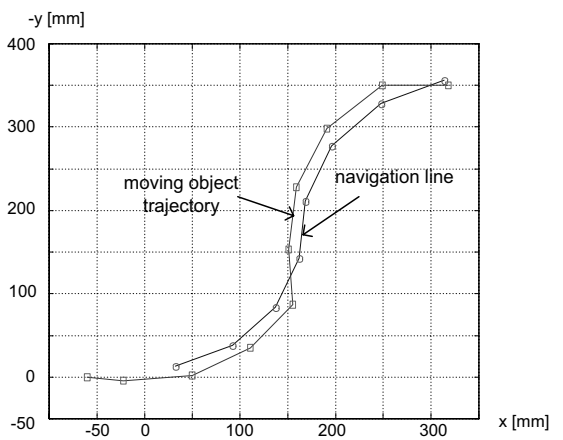


Fig. 19. Following of the navigation line by the autonomous VNS algorithm **(B).**

In the case of a combined algorithm it is more appropriate to track the further corners of the nearest landmark (they should be present in the field of view in two or more frames). At the very beginning, the motion is started in pre-specified direction in order to acquire the information about the position of a tracked point in the first two frames. After that, the commands moving the object above the tracked point are generated, while the camera is rotated in order to be oriented in direction of a motion. When the new object of "upper left corner" type is detected inside the image, its relative position is calculated in two next frames and the new commands for the linear and angular motion are generated.
Figure 20. illustrates the result of application of a combined algorithm in navigation task **(B)**.

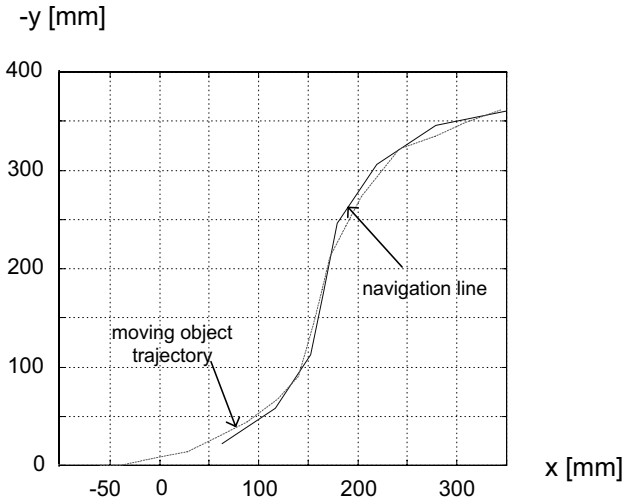


Fig. 20. Following of the navigation line by the dynamic vision algorithm (B).

Due to more meaningful changes in the image contents, one can not recognize here the regularity characterizing the previous case. On the other hand, the reduced amount of calculations allows the higher sampling frequency. As a result of this, a quality of following of the navigation line was slightly better in comparison to the result shown in Figure 19.

6. Conclusion

The algorithm of fusion of data originating from the strap-down inertial navigation system (INS) and the dynamic vision based visual navigation system (VNS) has been suggested for the general case when the appropriate landmarks are in the field of a TV camera's view. The procedure is of weighted averaging type, allowing the adjustment of weighting factors having in mind the physical nature of errors characterizing both systems and according to the self-evaluation of some intermediate estimates made inside the VNS. The overall procedure could be reasonably reduced according to the particular application and to some

a priori knowledge about the possible system errors by excluding some of the possible autonomous or assisted estimates.

Two particular examples have been used in order to illustrate this approach. In the first one, typical for the aerial vehicle motion control, the scenario was constructed under the realistic assumptions about the technical realization of a system, visibility conditions, and the noise levels inside the inertial sensors and in a TV image. It can be concluded that the INS position estimates could be efficiently improved by using the assisted estimates produced by the VNS. While for the height corrections in the INS one can always use a barometric sensor as a simpler solution, the actual benefits of this type of combined algorithm are mostly in the improvements of the position estimates in a horizontal plane (range, lateral deviation from nominal trajectory).

The second example is typical for the mobile robot applications as well as for the automatic motion control of the road vehicles. It represents the integration of a VNS and a reduced INS (just the acceleration measurements integrated in order to enable a dynamic vision based algorithm). This system has shown the advantages in comparison to the autonomous VNS. These consist mainly in the reductions of the computations regarding the distinguishing of the characteristic points of a reference object as well as in some improvements of a position estimation accuracy also (as a consequence of a relaxing the overall accuracy dependence on the results of image processing part of algorithm only). Finally, it should be pointed out that a VNS algorithm assisted by the INS data requires no a priori information about the shape and dimensions of the reference objects, which is beneficial also.

The analyzed examples are relatively simple ones but still meaningful for the vehicular motion control applications. More complex tasks including the rotational degrees of freedom should be considered as the more general cases of a fusion of VNS and INS, where the set of inertial instruments can be extended by using of the rate gyros. This way, the complex and noise sensitive procedures of determining of an angular orientation of a mobile robot or a vehicle, based on a machine vision alone, can be replaced by the usage of the inertial sensors' data. The future research is going to be oriented in this way.

7. References

Farrell, J.A., Barth, M. (1999): "*The Global Positioning System & Inertial Navigation*", McGraw Hill, 1999.

Frezza, R.; Perona, P.; Picci, G.; Soatto, S. (1994): "System-Theoretic Aspects of Dynamic Vision", in "*Trends in Control*", A. Isidori, (Ed.): Springer, Berlin, 1994., pp. 349-383.

Graovac, S. (2002): "A New Visual Navigation Algorithm Using Linear Acceleration Measurements", *Proceedings of the 10th Mediterranean Conference on Control and Automation*, Lisbon, Portugal, July 2002.

Graovac, S. (2004): "Principles of Fusion of Inertial Navigation and Dynamic Vision", *Journal of Robotic Systems*, Vol. 21, No1, pp. 13-22, January 2004.

Grewal, M. S., Weill, L. R. Andrews, A.P. (2001): "*Global Positioning Systems, Inertial Navigation, and Integration*", John Wiley & Sons, Inc., 2001.

Kaminer, I.; Pascoal, A.; Kang, W. (1999): "Integrated Vision/Inertial Navigation System Design Using Nonlinear Filtering", *Proceedings of the American Control Conference,* Vol. 3, pp. 1910 – 1914, San Diego, California, USA, June 1999.

Kanatani, K. (1993): "*Geometric Computation for Machine Vision*", Clarendon Press, Oxford, 1993.

Menon, P.K.A., Chatterji, G.B. Sridhar, B. (1993): "Electro-Optical Navigation for Aircraft", *IEEE Trans. On Aerospace and Electronic Systems*, Vol. AES-29, July 1993, pp. 825-832.

Roumeliotis, S.; Johnson, A.; Montgomery, J. (2002): "Augmenting Inertial Navigation with Image-Based Motion Estimation" *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002*, Vol. 4., pp. 4326-4333, May 2002, Washington DC, USA.

4

Sonar Sensor Interpretation and Infrared Image Fusion for Mobile Robotics

Mark Hinders, Wen Gao and William Fehlman
Department of Applied Science, The College of William & Mary in Virginia
NDE Lab @ 106 Jamestown Road, Williamsburg, VA 23187-8795

1. Introduction

The goal of our research is to give mobile robots the ability to deal with real world instructions *outdoors* such as "Go down to the big tree and turn left." Inherent in such a paradigm is the robot being able to recognize a "big tree" and do it in unstructured environments without too many pre-mapped fixed landmarks. Ultimately we envision mobile robots that unobtrusively mix in with pedestrian traffic and hence traveling primarily at a walking pace. With regard to sensors this is a different problem from robots designed to drive on roadways, since the necessary range of sensors is tied to the speed of the robot. It's also important to note that small mobile robots that are intended to mix with pedestrian traffic must normally travel at the same speed as the pedestrians, even if they occasionally scurry quickly down a deserted alley or slow way down to traverse a tricky obstacle, because people resent having to go around a slow robot while they are also easily startled by machines such as Segways that overtake them without warning. At walking speeds the range of sonar at about 50kHz is optimal, and there are none of the safety concerns one might have with lidar, for example. This type of sonar is precisely what bats use for echolocation; the goal of our research is to employ sonar sensors to allow mobile robots to recognize objects in the everyday environment based on simple signal processing algorithms tied to the physics of how the sonar backscatters from various objects. Our primary interest is for those sensors that can function well outdoors without regard to lighting conditions or even in the absence of daylight. We have built several 3D sonar scanning systems packaged as sensor heads on mobile robots, so that we are able to traverse the local environment and easily acquire 3D sonar scans of typical objects and structures. Of course sonar of the type we're exploring is not new. As early as 1773 it was observed that bats could fly freely in a dark room and pointed out that hearing was an important component of bats' orientation and obstacle avoidance capabilities (Au, 1993). By 1912 it was suggested that bats use sounds inaudible to humans to detect objects (Maxim, 1912) but it wasn't until 1938 that Griffin proved that bats use ultrasound to detect objects (Griffin, 1958).

More recently, Roman Kuc and his co-workers (Barshan & Kuc, 1992; Kleeman & Kuc, 1995) developed a series of active wide-beam sonar systems that mimic the sensor configuration of echolocating bats, which can distinguish planes, corners and edges necessary to navigate indoors. Catherine Wykes and her colleagues (Chou & Wykes, 1999) have built a prototype integrated ultrasonic/vision sensor that uses an off-the-shelf CCD camera and a four-element sonar phased array sensor that enables the camera to be calibrated using data from

the ultrasonic sensor. Leslie Kay (Kay, 2000) has developed and commercialized high-resolution octave band air sonar for spatial sensing and object imaging by blind persons. Phillip McKerrow and his co-workers (McKerrow & Harper, 1999; Harper & McKerrow, 2001; Ratner & McKerrow, 2003) have been focused on outdoor navigation and recognition of leafy plants with sonar. Other research groups are actively developing algorithms for the automatic interpretation of sensor data (Leonard & Durrant-Whyte, 1992; Crowley, 1985; Dror et al., 1995; Jeon & Kim, 2001). Our goal is to bring to bear a high level knowledge of the physics of sonar backscattering and then to apply sophisticated discrimination methods of the type long established in other fields (Rosenfeld & Kak, 1982; Theodoridis & Koutroumbas, 1998; Tou, 1968).

In section 2 we describe an algorithm to distinguish big trees, little trees and round metal poles based on the degree of asymmetry in the backscatter as the sonar beam is swept across them. The asymmetry arises from lobulations in the tree cross section and/or roughness of the bark. In section 3 we consider extended objects such as walls, fences and hedges of various types which may be difficult to differentiate under low light conditions. We key on features such as side peaks due to retroreflectors formed by the pickets and posts which can be counted via a deformable template signal processing scheme. In section 4 we discuss the addition of thermal infrared imagery to the sonar information and the use of Bayesian methods that allow us to make use of a priori knowledge of the thermal conduction and radiation properties of common objects under a variety of weather conditions. In section 5 we offer some conclusions and discuss future research directions.

2. Distinguishing Trees & Poles Via Ultrasound Backscatter Asymmetry

In this section we describe an algorithm to automatically distinguish trees from round metal poles with a sonar system packaged on a mobile robot. A polynomial interpolation of the square root of the backscattered signal energy vs. scan angle is first plotted. Asymmetry and fitting error are then extracted for each sweep across the object, giving a single point in an abstract phase space. Round metal poles are nearer to the origin than are trees, which scatter the sonar more irregularly due to lobulations and/or surface roughness. Results are shown for 20 trees and 10 metal poles scanned on our campus.

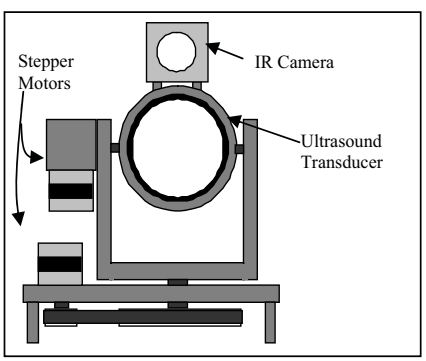


Fig. 1. Diagram of the scanner head. The dotted lines represent the two axes that the camera and ultrasound transducer rotate about.

Fig. 1 above shows a schematic of our biaxial sonar scan head. Although a variety of transducers can be used, all of the results shown here use narrow-beam AR50-8 transducers from Airmar (Milford, NH) which we've modified by machining a concavity into the matching layer in order to achieve beam half-widths of less than 8°. The transducers are scanned over perpendicular arcs via stepper motors, one of which controls rotation about the horizontal axis while the other controls rotation about the vertical axis. A custom motor controller board is connected to the computer via serial interface. The scanning is paused briefly at each orientation while a series of sonar tonebursts is generated and the echoes are recorded, digitized and archived on the computer. Fig. 2 shows the scanner mounted atop a mobile robotic platform which allows out-of-doors scanning around campus.

Fig. 2. Mobile robotic platform with computer-controlled scanner holding ultrasound transducer.

At the frequency range of interest both the surface features (roughness) and overall shape of objects affect the back-scattered echo. Although the beam-width is too broad to image in the traditional sense, as the beam is swept across a finite object variations in the beam profile give rise to characteristically different responses as the various scattering centers contribute constructively and destructively. Here we consider two classes of cylindrical objects outside, trees and smooth circular poles. In this study we scanned 20 trees and 10 poles, with up to ten different scans of each object recorded for off-line analysis (Gao, 2005; Gao & Hinders, 2005).

All data was acquired at 50kHz via the mobile apparatus shown in Figs. 1 and 2. The beam was swept across each object for a range of elevation angles and the RF echoes corresponding to the horizontal fan were digitized and recorded for off-line analysis. For each angle in the horizontal sweep we calculate the square root of signal energy in the back-scattered echo by low-pass filtering, rectifying, and integrating over the window corresponding to the echo from the object. For the smooth circular metal poles we find, as expected, that the backscatter energy is symmetric about a central maximum where the incident beam axis is normal to the surface. Trees tend to have a more complicated response due to non-circular cross sections and/or surface roughness of the bark. Rough bark can give enhanced backscatter for grazing angles where the smooth poles give very little response. We plot the square root of the signal energy vs. angular step and fit a 5th order

polynomial to it. Smooth circular poles always give a symmetric (bell-shaped) central response whereas rough and/or irregular objects often give responses less symmetric about the central peak. In general, one sweep over an object is not enough to tell a tree from a pole. We need a series of scans for each object to be able to robustly classify them. This is equivalent to a robot scanning an object repeatedly as it approaches. Assuming that the robot has already adjusted its path to avoid the obstruction, each subsequent scan gives a somewhat different orientation to the target. Multiple looks at the target thus increase the robustness of our scheme for distinguishing trees from poles because trees have more variations vs. look angle than do round metal poles.

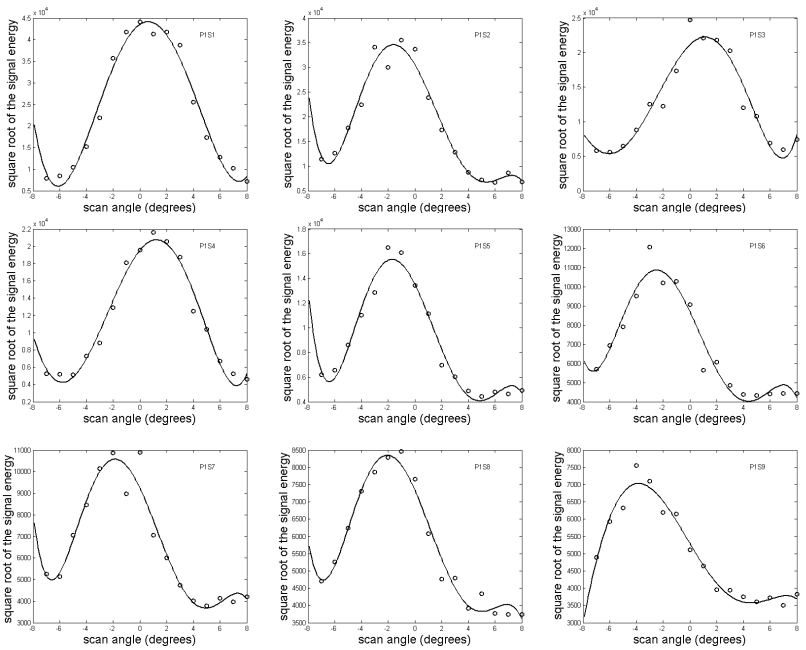


Fig. 3. Square root of signal energy plots of pole P1 when the sensor is (S1) 75cm (S2) 100cm (S3) 125cm (S4) 150cm (S5) 175cm (S6) 200cm (S7) 225cm (S8) 250cm (S9) 275cm from the pole.

Fig. 3 shows the square root of signal energy plots of pole P1 (a 14 cm diameter circular metal lamppost) from different distances and their 5th order polynomial interpolations. Each data point was obtained by low-pass filtering, rectifying, and integrating over the window corresponding to the echo from the object to calculate the square root of the signal energy as a measure of backscatter strength. For each object 16 data points were calculated as the beam was swept across it. The polynomial fits to these data are shown by the solid curve for 9 different scans. All of the fits in Fig. 3 are symmetric (bell-shaped) near the central scan angle, which is characteristic of a smooth circular pole. Fig. 4 shows the square root of signal energy plots of tree T14, a 19 cm diameter tree which has a relatively smooth surface. Nine scans are shown from different distances along with their 5th order

polynomial interpolations. Some of these are symmetric (bell-shaped) and some are not, which is characteristic of a round smooth-barked tree.

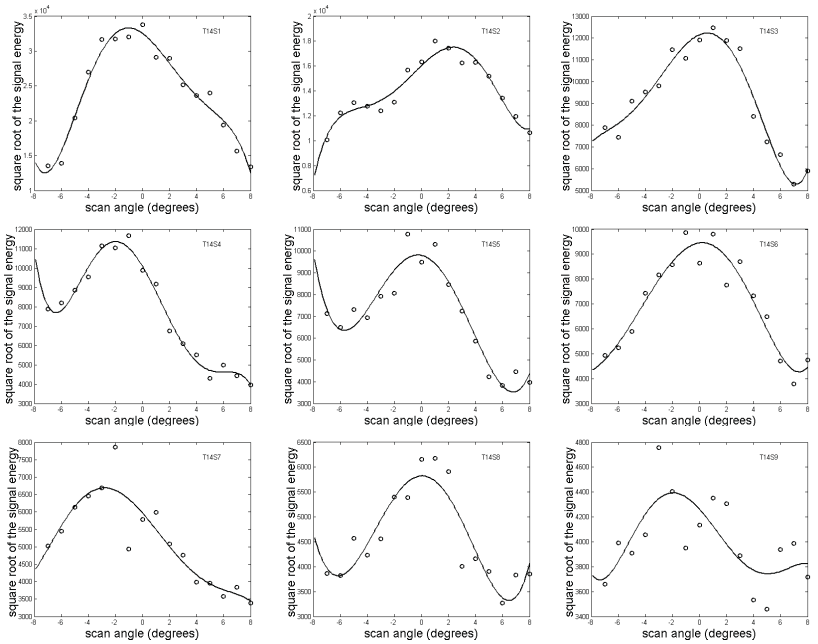


Fig. 4. Square root of signal energy plots of tree T14 when the sensor is (S1) 75cm (S2) 100cm (S3) 125cm (S4) 150cm (S5) 175cm (S6) 200cm (S7) 225cm (S8) 250cm (S9) 275cm from the tree.

Fig. 5 on the following page shows the square root of signal energy plots of tree T18, which is a 30 cm diameter tree with a rough bark surface. Nine scans are shown, from different distances, along with their 5th order polynomial interpolations. Only a few of the rough-bark scans are symmetric (bell-shaped) while most are not, which is characteristic of a rough and/or non-circular tree. We also did the same procedure for trees T15-T17, T19, T20 and poles P2, P3, P9, P10. We find that if all the plots are symmetric bell-shaped it can be confidently identified as a smooth circular pole. If some are symmetric bell-shaped while some are not, it can be identified as a tree.

Of course our goal is to have the computer distinguish trees from poles automatically based on the shapes of square root of signal energy plots. The feature vector $\boldsymbol{x}$ we choose contains two elements: Asymmetry and Deviation. If we let x_1 represent Asymmetry and x_2 represent Deviation, the feature vector can be written as $\boldsymbol{x}=[x_1, x_2]$. For example, Fig. 6 on the following page is the square root of signal energy plot of pole P1 when the distance is 200cm. For x_1 we use Full-Width Half Maximum (FWHM) to define asymmetry. We cut the full width half-maximum into two to get the left width L1 and right width L2. Asymmetry is defined as the difference between L1 and L2 divided by FWHM, which is $|L1-L2|/|L1+L2|$. The Deviation x_2 we define as the average distance from the experimental

data point to the fitted data point at the same *x*-axis location, divided by the total height of the fitted curve H. In this case, there are 16 experimental data points, so Deviation = (|d1|+|d2|+...+|d16|)/(16H). For the plot above, we get Asymmetry=0.0333, which means the degree of asymmetry is small. We also get Deviation=0.0467, which means the degree of deviation is also small.

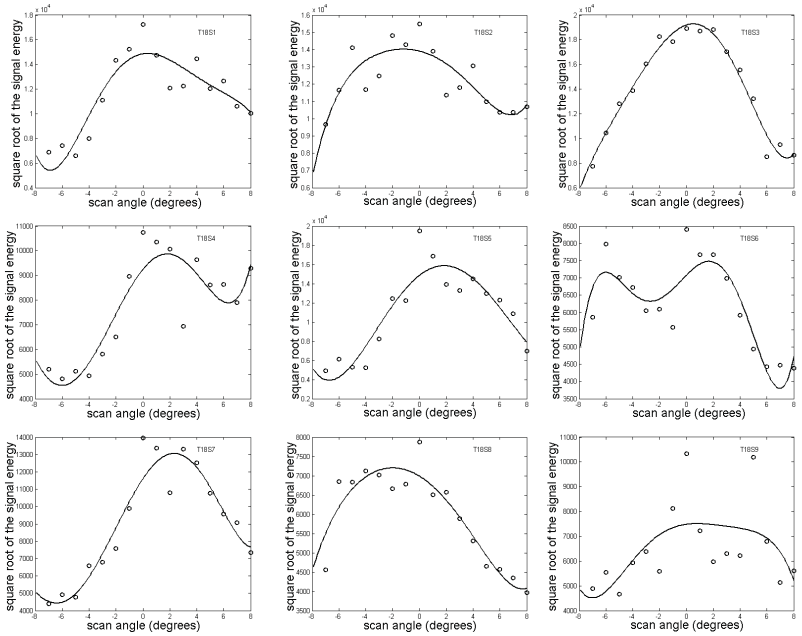


Fig. 5. Square root of signal energy plots of tree T18 when the sensor is (S1) 75cm (S2) 100cm (S3) 125cm (S4) 150cm (S5) 175cm (S6) 200cm (S7) 225cm (S8) 250cm (S9) 275cm from the tree.

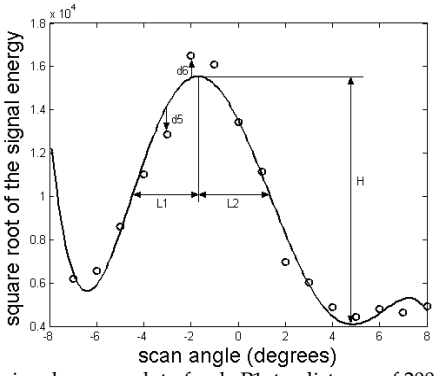


Fig. 6. Square root of signal energy plot of pole P1at a distance of 200cm.

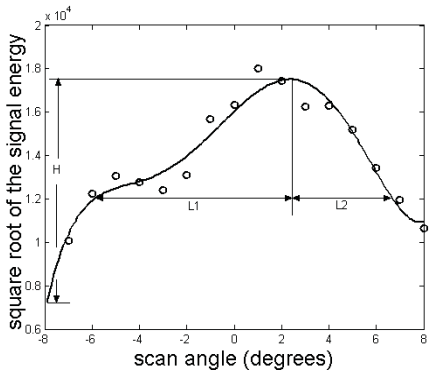


Fig. 7. Square root of signal energy plot of tree T14 at a distance of 100cm.

For the square root of energy plot of tree T14 in Fig. 7, its Asymmetry will be bigger than the more bell-shaped plots. Trees T1-T4, T8, T11-T13 have rough surfaces (tree group No.1) while trees T5-T7, T9-T10 have smooth surfaces (tree group No.2). The pole group contains poles P4-P8. Each tree has two sweeps of scans while each pole has four sweeps of scans. We plot the Asymmetry-Deviation phase plane in Fig. 8. Circles are for the pole group while stars indicate tree group No.1 and dots indicate tree group No.2. We find circles representing the poles are usually within [0,0.2] on the Asymmetry axis. Stars representing the rough surface trees (tree group No.1) are spread widely in Asymmetry from 0 to 1. Dots representing the smooth surface trees (tree group No.2) are also within [0,0.2] on the Asymmetry axis. Hence, we conclude that two scans per tree may be good enough to tell a rough tree from a pole, but not to distinguish a smooth tree from a pole.

We next acquired a series of nine or more scans from different locations relative to each object, constructed the square root of signal energy plots from the data, extracted the Asymmetry and Deviation features from each sweep of square root of signal energy plots and then plotted them in the phase plane. If all of the data points for an object are located within a small Asymmetry region, we say it's a smooth circular pole. If some of the results are located in the small Asymmetry region and some are located in the large Asymmetry region, we can say it's a tree. If all the dots are located in the large Asymmetry region, we say it's a tree with rough surface.

Our purpose is to classify the unknown cylindrical objects by the relative location of their feature vectors in a phase plane, with a well-defined boundary to segment the tree group from the pole group. First, for the series of points of one object in the Asymmetry-Deviation scatter plot, we calculate the average point of the series of points and find the average squared Euclidean distance from the points to this average point. We then calculate the Average Squared Euclidean Distance from the points to the average point and call it Average Asymmetry. We combine these two features into a new feature vector and plot it into an Average Asymmetry-Average Squared Euclidean Distance phase plane. We then get a single point for each tree or pole, as shown in Fig. 9. Stars indicate the trees and circles indicate poles. We find that the pole group clusters in the small area near the origin (0,0) while the tree group is spread widely but away from the origin. Hence, in the Average Asymmetry-Average Squared Euclidean Distance phase plane, if an object's feature vector

is located in the small area near the origin, which is within [0,0.1] in Average Asymmetry and within [0,0.02] in Average Squared Euclidean Distance, we can say it's a pole. If it is located in the area away from the origin, which is beyond the set area, we can say it's a tree.

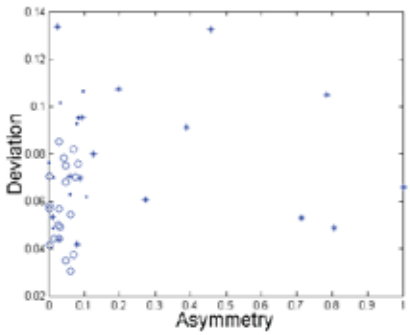


Fig. 8. Asymmetry-Deviation phase plane of the pole group and two tree groups. Circles indidate poles, dots indicate smaller smooth bark trees, and stars indicate the larger rough bark trees.

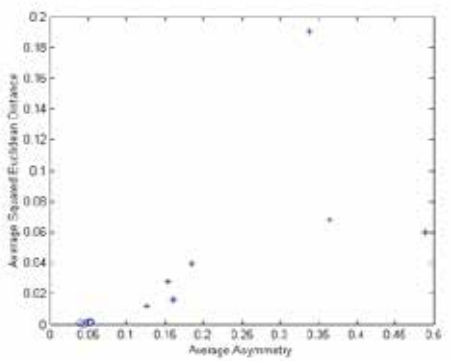


Fig. 9. Average Asymmetry-Average Squared Euclidean Distance phase plane of trees T14-T20 and poles P1-P3, P9 and P10.

3. Distinguishing Walls, Fences & Hedges with Deformable Templates

In this section we present an algorithm to distinguish several kinds of brick walls, picket fences and hedges based on the analysis of backscattered sonar echoes. The echo data are acquired by our mobile robot with a 50kHz sonar computer-controlled scanning system packaged as its sensor head (Figs. 1 and 2). For several locations along a wall, fence or hedge, fans of backscatter sonar echoes are acquired and digitized as the sonar transducer is swept over horizontal arcs. Backscatter is then plotted vs. scan angle, with a series of N-peak deformable templates fit to this data for each scan. The number of peaks in the best-fitting N-peak template indicates the presence and location of retro-reflectors, and allows automatic categorization of the various fences, hedges and brick walls.

In general, one sweep over an extended object such as a brick wall, hedge or picket fence is not sufficient to identify it (Gao, 2005). As a robot is moving along such an object, however, it is natural to assume that several scans can be taken from different locations. For objects such as picket fences, for example, there will be a natural periodicity determined by post spacing. Brick walls with architectural features (buttresses) will similarly have a well-defined periodicity that will show up in the sonar backscatter data. Defining one spatial unit for each object in this way, five scans with equal distances typically cover a spatial unit. Fig. 10 shows typical backscatter plots for a picket fence scanned from inside (the side with the posts). Each data point was obtained by low-pass filtering, rectifying, and integrating over the window corresponding to the echo from the object to calculate the square root of the signal energy as a measure of backscatter. Each step represents 1° of scan angle with zero degrees perpendicular to the fence. Note that plot (a) has a strong central peak, where the robot is lined up with a square post that reflects strongly for normal incidence. There is some backscatter at the oblique angles of incidence because the relatively broad sonar beam (spot size typically 20 to 30 cm diameter) interacts with the pickets (4.5 cm in width, 9 cm on center) and scatters from their corners and edges. The shape of this single-peak curve is thus a characteristic response for a picket fence centered on a post.

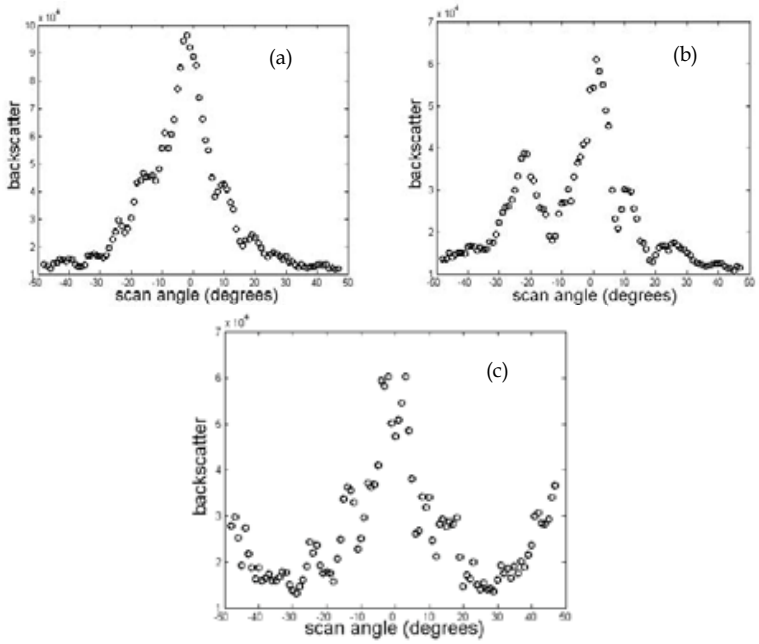


Fig. 10. Backscatter plots for a picket fence scanned from the inside, with (a) the robot centered on a post, (b) at 25% of the way along a fence section so that at zero degrees the backscatter is from the pickets, but at a scan angle of about –22.5 degrees the retroreflector made by the post and the adjacent pickets causes a secondary peak. (c) at the middle of the fence section, such that the retroreflectors made by each post show up at the extreme scan angles.

Plot (b) in Fig. 10 shows not only a central peak but also a smaller side peak. The central peak is from the pickets while the side peak is from the right angle made by the side surface of the post (13 x 13 cm) and the adjacent pickets, which together form a retro-reflector. The backscatter echoes from a retro-reflector are strong for a wide range of the angle of incidence. Consequently, a side peak shows up when the transducer is facing a retroreflector, and the strength and spacing of corresponding side peaks carries information about features of extended objects. Note that a picket fence scanned from the outside will be much less likely to display such side peaks because the posts will tend to be hidden by the pickets. Plot (c) in Fig. 10 also displays a significant central peak. However, its shape is a little different from the first and second plots. Here when the scan angle is far from the central angle the backscatter increases, which indicates a retro-reflector, i.e. the corner made by the side surface of a post is at both extreme edges of the scan.

Fig. 11 shows two typical backscatter plots for a metal fence with brick pillars. The brick pillars are 41 cm square and the metal pickets are 2 cm in diameter spaced 11 cm on center, with the robot scanning from 100cm away. Plot (a) has a significant central peak because the robot is facing the square brick pillar. The other has no apparent peaks because the robot is facing the metal fence between the pillars. The round metal pickets have no flat surfaces and no retro-reflectors are formed by the brick pillars. The chaotic nature of the backscatter is due to the broad beam of the sonar interacting with multiple cylindrical scatterers, which are each comparable in size to the sonar wavelength. In this "Mie-scattering" regime the amount of constructive or destructive interference from the multiple scatterers changes for each scan angle. Also, note that the overall level of the backscatter for the bottom plot is more than a factor of two smaller than when the sonar beam hits the brick pillar squarely.

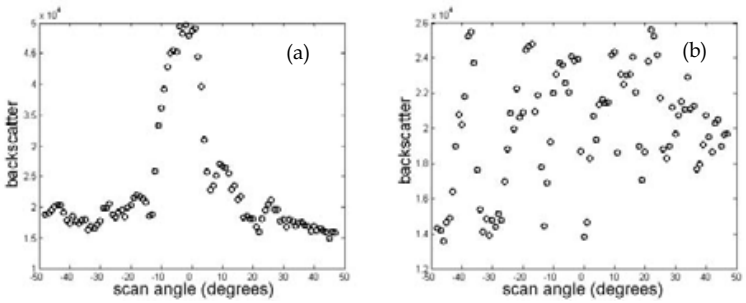


Fig. 11. Backscatter plots of a unit of the metal fence with brick pillar with the robot facing (a) brick pillar and (b) the metal fencing, scanned at a distance of 100cm.

Fig. 12 shows typical backscatter plots for brick walls. Plot (a) is for a flat section of brick wall, and looks similar to the scan centered on the large brick pillar in Fig. 11. Plot (b) is for a section of brick wall with a thick buttress at the extreme right edge of the scan. Because the buttress extends out 10 cm from the plane of the wall, it makes a large retroreflector which scatters back strongly at about 50 degrees in the plot. Note that this size of this side-peak depends strongly on how far the buttress extends out from the wall. We've also scanned walls with regularly-spaced buttresses that extend out only 2.5 cm (Gao, 2005) and found that they behave similarly to the thick-buttress walls, but with correspondingly smaller side peaks.

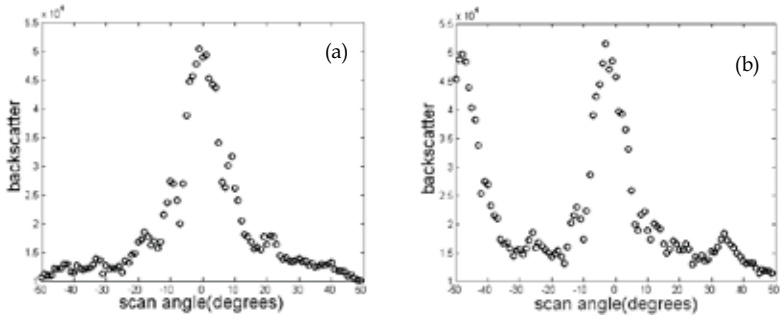


Fig. 12 Backscatter plots of a unit of brick wall with thick buttress with the robot at a distance of 100cm facing (a) flat section of wall and (b) section including retroreflecting buttress at extreme left scan angle.

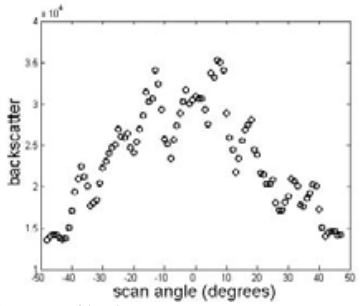


Fig. 13 Backscatter plot of a unit of hedge.

Fig. 13 shows a typical backscatter plot for a trimmed hedge. Note that although the level of the backscatter is smaller than for the picket fence and brick wall, the peak is also much broader. As expected the foliage scatters the sonar beam back over a larger range of angles.

Backscatter data of this type was recorded for a total of seven distinct objects: the wood picket fence described above from inside (side with posts), that wood picket fence from outside (no posts), the metal fence with brick pillars described above, a flat brick wall, a trimmed hedge, and brick walls with thin (2.5 cm) and thick (10 cm) buttresses, respectively (Gao, 2005; Gao & Hinders, 2006). For those objects with spatial periodicity formed by posts or buttresses, 5 scans were taken over such a unit. The left- and right-most scans were centered on the post or buttress, and then three scans were taken evenly spaced in between. For typical objects scanned from 100 cm away with +/- 50 degrees scan angle the middle scans just see the retroreflectors at the extreme scan angles, while the scans 25% and 75% along the unit length only have a single side peak from the nearest retro-reflector. For those objects without such spatial periodicity a similar unit length was chosen for each with five evenly spaced scans taken as above. Analyzing the backscatter plots constructed from this data, we concluded that the different objects each have a distinct sequence of backscatter plots, and that it should be possible to automatically distinguish such objects based on characteristic features in these backscatter plots. We have implemented a deformable

template matching scheme to use this backscattering behaviour to differentiate the seven types of objects.

A deformable template is a simple mathematically defined shape that can be fit to the data of interest without losing its general characteristics (Gao, 2005). For example, for a one-peak deformable template, its peak location may change when fitting to different data, but it always preserves its one peak shape characteristic. For each backscatter plot we next create a series of deformable N-peak templates (N=1, 2, 3… Nmax) and then quantify how well the templates fit for each N. Obviously a 2-peak template (N=2) will fit best to a backscatter plot with two well-defined peaks. After consideration of a large number of backscatter vs. angle plots of the types in the previous figures, we have defined a general sequence of deformable templates in the following manner.

For one-peak templates we fit quintic functions to each of the two sides of the peak, located at x_p, each passing through the peak as well as the first and last data points, respectively. Hence, the left part of the one-peak template is defined by the function $y = c_1(x - x_L)^5 + B(x_L)$ which passes through the peak giving $c_1 = \dfrac{B(x_p) - B(x_L)}{(x_p - x_L)^5}$.

Here *B(x)* is the value of the backscatter at angle *x*. Therefore, the one-peak template function defined over the range from $x = x_L$ to $x = x_p$ is

$$y = \frac{B(x_p) - B(x_L)}{(x_p - x_L)^5}(x - x_L)^5 + B(x_L)\,. \tag{1a}$$

The right part of the one peak template is defined similarly over the range between $x = x_p$ to $x = x_R$, i.e. the function $y = c_2(x - x_R)^5 + B(x_R)$, with c_2 given as $c_2 = \dfrac{B(x_p) - B(x_R)}{(x_p - x_R)^5}$.

Therefore, the one-peak template function of the right part is

$$y = \frac{B(x_p) - B(x_R)}{(x_p - x_R)^5}(x - x_R)^5 + B(x_R)\,. \tag{1b}$$

For the double-peak template, the two selected peaks x_{p1} and x_{p2} as well as the location of the valley x_v between the two backscatter peaks separate the double-peak template into four regions with $x_{p1}<x_v<x_{p2}$. The double peak template is thus comprised of four parts, defined as second-order functions between the peaks and quintic functions outbound of the two peaks.

$$y = \frac{B(x_{p1}) - B(x_L)}{(x_{p1} - x_L)^5}(x - x_L)^5 + B(x_L) \quad x_L \le x \le x_{p1}$$

$$y = \frac{B(x_{p1}) - B(x_v)}{(x_{p1} - x_v)^2}(x - x_v)^2 + B(x_v) \quad x_{p1} \le x \le x_v$$

$$y = \frac{B(x_{p2}) - B(x_v)}{(x_{p2} - x_v)^2}(x - x_v)^2 + B(x_v) \quad x_v \le x \le x_{p2}$$

$$y = \frac{B(x_{p2}) - B(x_R)}{(x_{p2} - x_R)^5}(x - x_R)^5 + B(x_R) \quad x_{p2} \le x \le x_R$$

In the two middle regions, shapes of quadratic functions are more similar to the backscatter plots. Therefore, quadratic functions are chosen to form the template instead of quintic functions. Fig. 14 shows a typical backscatter plot for a picket fence as well as the corresponding single- and double-peak templates. The three, four, five, …, -peak template building follows the same procedure, with quadratic functions between the peaks and quintic functions outboard of the first and last peaks.

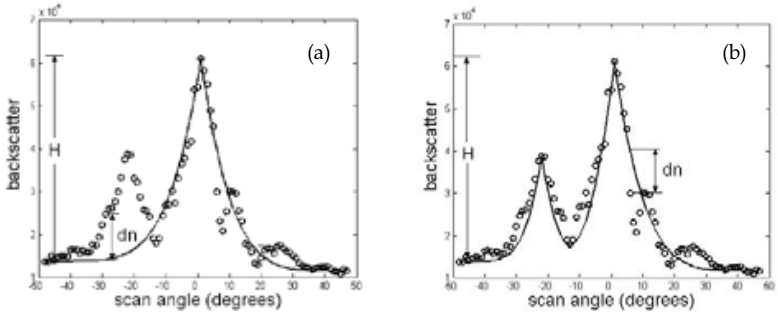


Fig. 14. Backscatter plots of the second scan of a picket fence and its (a) one-peak template (b) two-peak template

In order to characterize quantitatively how well the N-peak templates each fit a given backscatter plot, we calculate the sum of the distances from the backscatter data to the template at the same scan angle normalized by the total height H of the backscatter plot and the number of scan angles. For each backscatter plot, this quantitative measure of goodness of fit (Deviation) to the template is calculated automatically for N=1 to N=9 depending upon how many distinct peaks are identified by our successive enveloping and peak-picking algorithm (Gao, 2005). We can then calculate Deviation vs. N and fit a 4th order polynomial to each. Where the deviation is smallest indicates the N-peak template which fits best. We do this on the polynomial fit rather than on the discrete data points in order to automate the process, i.e. we differentiate the Deviation vs. N curve and look for zero crossings by setting a threshold as the derivative approaches zero from the negative side. This corresponds to the deviation decreasing with increasing N and approaching the minimum deviation, i.e. the best-fit N-peak template.

Because the fit is a continuous curve we can consider non-integer N, i.e. the derivative value of the 4th order polynomial fitting when the template value is N+0.5. This describes how the 4th order polynomial fitting changes from N-peak template fitting to (N+1)-peak template fitting. If it is positive or a small negative value, it means that in going from the N-peak template to the (N+1)-peak template, the fitting does not improve much and the N-peak template is taken to be better than the (N+1)-peak template. Accordingly, we first set a threshold value and calculate these slopes at both integer and half-integer values of N. The threshold value is set to be -0.01 based on experience with data sets of this type, although this threshold could be considered as an adjustable parameter. We then check the value of the slopes in order. The N-peak-template is chosen to be the best-fit template when the slope at (N+0.5) is bigger than the threshold value of -0.01 for the first time.

We also set some auxiliary rules to better to pick the right number of peaks. The first rule helps the algorithm to key on retroreflectors and ignore unimportant scattering centers: if

the height ratio of a particular peak to the highest peak is less than 0.2, it is not counted as a peak. Most peaks with a height ratio less than 0.2 are caused by small scattering centers related to the rough surface of the objects, not by a retro-reflector of interest. The second rule is related to the large size of the sonar beam: if the horizontal difference of two peaks is less than 15 degrees, we merge them into one peak. Most of the double peaks with angular separation less than 15 degrees are actually caused by the same major reflector interacting with the relatively broad sonar beam. Two 5-dimensional feature vectors for each object are next formed. The first is formed from the numbers of the best fitting templates, i.e. the best N for each of the five scans of each object. The second is formed from the corresponding Deviation for each of those five scans. For example, for a picket fence scanned from inside, the two 5-dimensional feature vectors are N=[1,2,3,2,1] and D=[0.0520, 0.0543, 0.0782, 0.0686, 0.0631]. For a flat brick wall, they are N=[1,1,1,1,1] and D=[0.0549, 0.0704, 0.0752, 0.0998, 0.0673].

The next step is to determine whether an unknown object can be classified based on these two 5-dimensional feature vectors. Feature vectors with higher dimensions (>3) are difficult to display visually, but we can easily deal with them in a hyper plane. The Euclidean distance of a feature vector in the hyper plane from an unknown object to the feature vector of a known object is thus calculated and used to determine if the unknown object is similar to any of the objects we already know.

For both 5-dimensional feature vectors of an unknown object, we first calculate their Euclidean distances to the corresponding template feature vectors of a picket fence. $\Delta N1 = |N_{unknown} - N_{picketfence}|$ is the Euclidean distance between the N vector of the unknown object $N_{unknown}$ to the N vector of the picket fence $N_{picketfence}$. Similarly, $\Delta D1 = | N_{unknown} - N_{picketfence} |$ is the Euclidean distance between the D vectors of the unknown object $N_{unknown}$ and the picket fence $N_{picketfence}$. We then calculate these distances to the corresponding feature vectors of a flat brick wall $\Delta N2$, $\Delta D2$, their distances to the two feature vectors of a hedge $\Delta N3$, $\Delta D3$ and so on.

The unknown object is then classified as belonging to the kinds of objects whose two feature vectors are nearest to it, which means both ΔN and ΔD are small. Fig. 15 is an array of bar charts showing these Euclidean distances of two feature vectors of a unknown objects to the two feature vectors of seven objects we already know. The horizontal axis shows different objects numbered according to"1" for picket fence scanned from the inside "2" for a flat brick wall"3" for a trimmed hedge "4" for a brick wall with thin buttress "5" for a brick wall with thick buttress "6" for a metal fence with brick pillar and"7" for the picket fence scanned from the outside. The vertical axis shows the Euclidean distances of feature vectors of an unknown object to the 7 objects respectively. For each, the height of black bar and grey bar at object No.1 represent $\Delta N1$ and $10\Delta D1$ respectively while the height of black bar and grey bar at object No.2 represent $\Delta N2$ and $10\Delta D2$ respectively, and so on. In the first chart both the black bar and grey bar are the shortest when comparing to the N, D vectors of a picket fence scanned from inside. Therefore, we conclude that this unknown object is a picket fence scanned from inside, which it is. Note that the D values have been scaled by a factor of ten to make the bar charts more readable. The second bar chart in Fig. 15 has both the black bar and grey bar the shortest when comparing to N, D vectors of object No.1—picket fence scanned from inside, which is what it is. The third bar chart in Fig. 15 has both the black bar and grey bar shortest when comparing to N, D vectors of object No.2 —flat brick wall and object No.4 —brick wall with thin buttress. That means the most probable kinds of the

unknown object are flat brick wall or brick wall with thin buttress. Actually it is a flat brick wall.

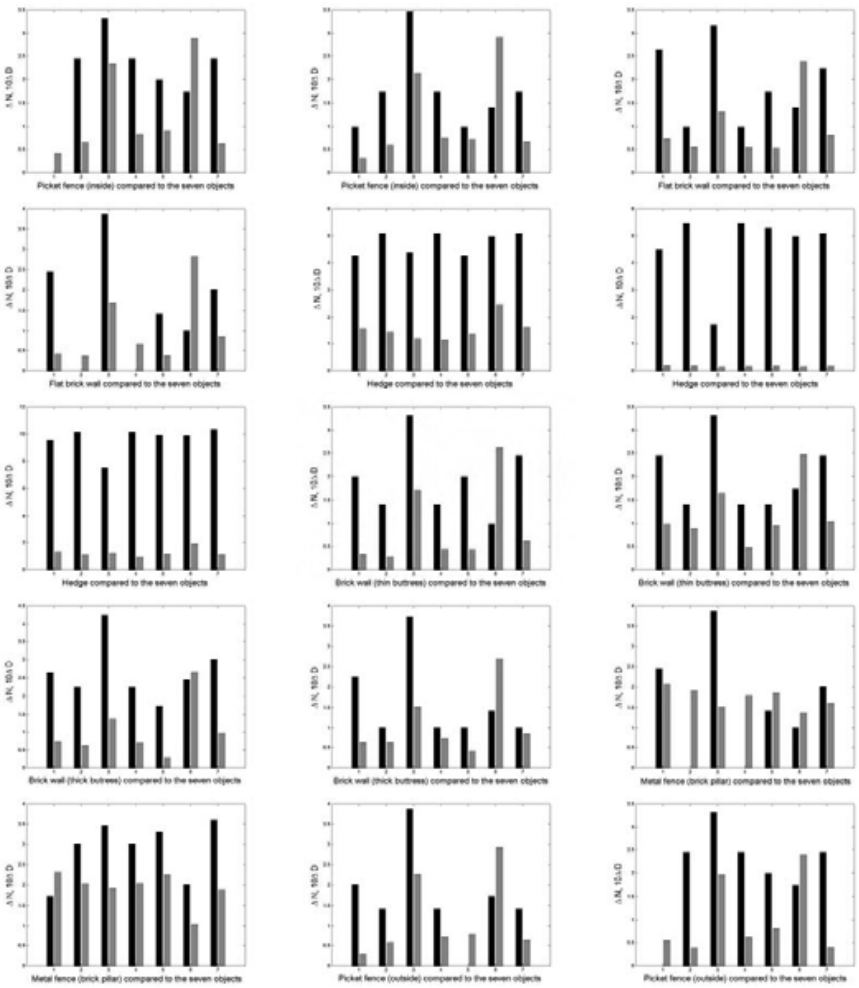


Fig. 15. ΔN (black bar) and $10\Delta D$ (gray bar) for fifteen objects compared to the seven known objects: *1* picket fence from inside, *2* flat brick wall, *3* hedge, *4* brick wall with thin buttress, *5* brick wall with thick buttress, *6* metal fence with brick pillar, *7* picket fence from outside.

Table 1 displays the results of automatically categorizing two additional scans of each of these seven objects. In the table, the + symbols indicate the correct choices and the x

symbols indicate the few incorrect choices. Note that in some cases the two feature vector spaces did not agree on the choice, and so two choices are indicated. Data sets 1A and 1B are both picket fences scanned from inside. They are correctly categorized as object No.1. Data sets 2A and 2B are from flat brick walls. They are categorized as either object No.2 (flat brick wall) or object No.4 (brick wall with thin buttress) which are rather similar objects. Data sets 3A and 3B are from hedges and are correctly categorized as object No.3. Data sets 4A and 4B are from brick walls with thin buttress. 4A is categorized as object No.2 (flat brick wall) or object No.4 (brick wall with thin buttress). Data sets 5A and 5B are from brick walls with thick buttress. Both are correctly categorized as object No.5. Data sets 6A and 6B are from metal fences with brick pillars. 6B is properly categorized as object No.6. 6B is categorized as either object No.6 (metal fence with brick pillar) or as object No.2 (flat brick wall). Data sets 7A and 7B are from picket fences scanned from outside, i.e. the side without the posts. 7A is mistaken as object No.5 (brick wall with thick buttress) while 7B is mistaken as object No.1 (picket fence scanned from inside). Of the fourteen new data sets, eight are correctly categorized via agreement with both feature vectors, four are correctly categorized by one of the two feature vector, and two are incorrectly categorized. Both of the incorrectly categorized data sets are from picket fence scanned from outside, presumably due to the lack of any significant retro-reflectors, but with an otherwise complicated backscattering behavior.

	1A	*2A*	*3A*	*4A*	*5A*	*6A*	*7A*	*1B*	*2B*	*3B*	*4B*	*5B*	*6B*	*7B*
1	+							+						X
2		+		X		X			+					
3			+							+				
4		X		+					X		+			
5					+		X					+		
6						+							+	
7														

Table 1. Results categorizing 2 additional data sets for each object.

4. Thermal Infrared Imaging as a Mobile Robot Sensor

In the previous sections we have used 50 kHz features in the ultrasound backscattering to distinguish common objects. Here we discuss the use of thermal infrared imaging as a

complementary technique. Note that both ultrasound and infrared are independent of lighting conditions, and so are appropriate for use both day and night. The technology necessary for infrared imaging has only recently become sufficiently portable, robust and inexpensive to imagine exploiting this full-field sensing modality for small mobile robots. We have mounted an infrared camera on one of our mobile robots and begun to systematically explore the behavior of the classes of outdoor objects discussed in the previous sections.

Our goal is simple algorithms that extract features from the infrared imagery in order to complement what can be done with the 50 kHz ultrasound. For this preliminary study, infrared imagery was captured on a variety of outdoor objects during a four-month period, at various times throughout the days and at various illumination/temperature conditions. The images were captured using a Raytheon *ControlIR* 2000B long-wave (7-14 micron) infrared thermal imaging video camera with a 50 mm focal length lens at a distance of 2.4 meters from the given objects. The analog signals with a 320X240 pixel resolution were converted to digital signals using a *GrabBeeIII* USB Video Grabber, all mounted on board a mobile robotic platform similar to Fig. 2. The resulting digital frames were processed offline in MATLAB. Table 1 below provides the times, visibility conditions, and ambient temperature during each of the nine sessions. During each session, the infrared images were captured on each object at three different viewing angles: normal incidence, 45 degrees from incidence, and 60 degrees from incidence. A total of 27 infrared images were captured on each object during the nine sessions.

Date	Time Span	Visibility	Temp. (°F)
8 Mar 06	0915-1050	Sunlight, Clear Skies	49.1
8 Mar 06	1443-1606	Sunlight, Clear Skies	55.0
8 Mar 06	1847-1945	No Sunlight, Clear Skies	49.2
10 Mar 06	1855-1950	No Sunlight, Clear Skies	63.7
17 Mar 06	0531-0612	No Sunlight-Sunrise, Slight Overcast	46.1
30 May 06	1603-1700	Sunlight, Clear Skies	87.8
30 May 06	2050-2145	No Sunlight, Partly Cloudy	79.6
2 Jun 06	0422-0513	No Sunlight, Clear Skies	74.2
6 Jun 06	1012-1112	Sunlight, Partly Cloudy	68.8

Table 2. Visibility conditions and temperatures for the nine sessions of capturing infrared images of the nine stationary objects.

The infrared images were segmented to remove the image background, with three center segments and three periphery segments prepared for each. A Retinex algorithm (Rahman, 2002) was used to enhance the details in the image, and a highpass Gaussian filter (Gonzalez et al., 2004) was applied to attenuate the lower frequencies and sharpen the image. By attenuating the lower frequencies that are common to most natural objects, the remaining higher frequencies help to distinguish one object from another. Since the discrete Fourier transform used to produce the spectrum assumes the frequency pattern of the image is periodic, a high-frequency drop-off occurs at the edges of the image. These "edge effects" result in unwanted intense horizontal and vertical artifacts in the spectrum, which are suppressed via the edgetaper function in MATLAB. The final preprocessing step is to apply a median filter that denoises the image without reducing the previously established sharpness of the image.

Fig. 16 Cedar Tree visible (left) and Infrared (right) Images.

Fig. 16 shows the visible and infrared images of a center segment of the cedar tree captured at 1025 hours on 8 March 2006. The details in the resulting preprocessed image are enhanced and sharpened due to Retinex, highpass Gaussian filter, and median filter. We next 2D Fouier transform the preprocessed image and take the absolute value to obtain the spectrum, which is then transformed to polar coordinates with angle measured in a clockwise direction from the polar axis and increasing along the columns in the spectrum's polar matrix. The linear radius (i.e., frequencies) in polar coordinates increases down the rows of the polar matrix. Fig. 17 display the spectrum and polar spectrum of the same center segment of the cedar tree.

Fig. 17 Frequency Spectrum (left) and Polar Spectrum (right) of cedar tree center segment.

Sparsity provides a measure of how well defined the edge directions are on an object (Luo & Boutell, 2005) useful for distinguishing between "manmade" and natural objects in visible imagery. Four object features generated in our research were designed in a similar manner. First, the total energy of the frequencies along the spectral radius was computed for angles from 45 to 224 degrees. This range of angle values ensures that the algorithm captures all possible directions of the frequencies on the object in the scene. A histogram with the angle values along the abscissa and total energy of the frequencies on the ordinate is smoothed using a moving average filter. The values along the ordinate are scaled to obtain frequency energy values ranging from 0 to 1 since we are only interested in how well the edges are defined about the direction of the maximum frequency energy, not the value of the frequency energy. The resulting histogram is plotted as a curve with peaks representing directions of maximum frequency energy. The full width at 80% of the maximum (FW(0.80)M) value on the curve is used to indicate the amount of variation in frequency energy about a given direction. Four features are generated from the resulting histogram defined by the terms: sparsity and direction. The sparsity value provides a measure of how well defined the edge directions are on an object. The value for sparsity is the ratio of the global maximum scaled frequency energy to the FW(0.80)M along a given interval in the histogram. Thus, an object with well defined edges along one given direction will display a curve in the histogram with a global maximum and small FW(0.80)M, resulting in a larger sparsity value compared to an object with edges that vary in direction. To compute the feature values, the intervals from 45 to 134 degrees and from 135 to 224 degrees were created along the abscissa of the histogram to optimally partition the absolute vertical and horizontal components

in the spectrum. The sparsity value along with its direction are computed for each of the partitioned intervals. A value of zero is provided for both the sparsity and direction if there is no significant frequency energy present in the given interval to compute the FW(0.80)M.
By comparing the directions (in radians) of the maximum scaled frequency energy along each interval, four features are generated: Sparsity about Maximum Frequency Energy (1.89 for tree vs. 2.80 for bricks), Direction of Maximum Frequency Energy (3.16 for tree vs. 1.57 for bricks), Sparsity about Minimum Frequency Energy (0.00 for tree vs. 1.16 for bricks), Direction of Minimum Frequency Energy (0.00 for tree vs. 3.14 for bricks). Fig. 19 below compares the scaled frequency energy histograms for the cedar tree and brick wall (Fig. 18), respectively.

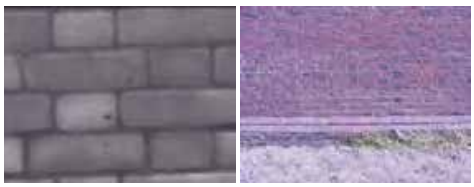


Fig. 18. Brick Wall Infrared (left) and Visible (right) Images.

As we can see in the histogram plot of the cedar tree (Fig. 19, left) the edges are more well defined in the horizontal direction, as expected. Furthermore, the vertical direction presents no significant frequency energy. On the other hand, the results for the brick wall (Fig. 19, right) imply edge directions that are more well defined in the vertical direction. The brick wall results in a sparsity value and direction associated with minimum frequency energy. Consequently, these particular results would lead to features that could allow us to distinguish the cedar tree from the brick wall.
Curvature provides a measure to distinguish cylindrical shaped objects from flat objects (Sakai & Finkel, 1995) since the ratio of the average peak frequency between the periphery and the center of an object in an image is strongly correlated with the degree of surface curvature. Increasing texture compression in an image yields higher frequency peaks in the spectrum. Consequently, for a cylindrically shaped object, we should see more texture compression and corresponding higher frequency peaks in the spectrum of the object's periphery compared to the object's center.

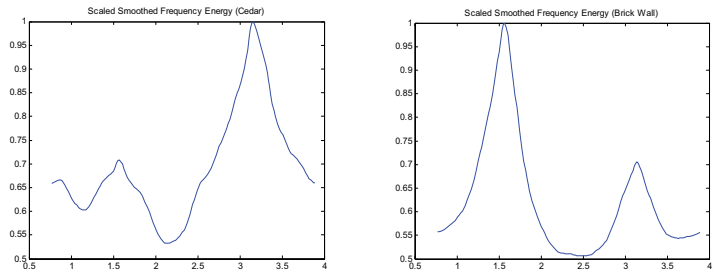


Fig. 19. Cedar (left) and Brick Wall (right) histogram plots.

To compute the curvature feature value for a given object, we first segment 80x80 pixel regions at the periphery and center of an object's infrared image. The average peak frequency in the

horizontal direction is computed for both the periphery and center using the frequency spectrum. Since higher frequencies are the primary contributors in determining curvature, we only consider frequency peaks at frequency index values from 70 to 100. The curvature feature value is computed as the ratio of the average horizontal peak frequency in the periphery to that of the center. Fig. 20 compares the spectra along the horizontal of both the center and periphery segments for the infrared image of a cedar tree and a brick wall, respectively.

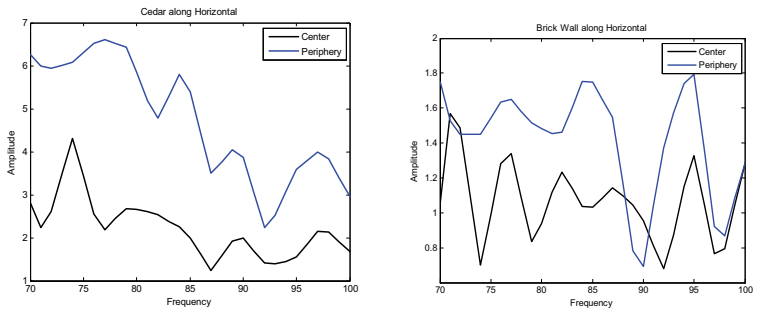


Fig. 20 Cedar (left) and Brick Wall (right) Center vs. Periphery Frequency Energy Spectrum along Horizontal. The computed curvature value for the cedar tree is 2.14, while the computed curvature for the brick wall is 1.33.

As we can see in the left plot of Fig. 20 above, the periphery of the cedar tree's infrared image has more energy at the higher frequencies compared to the center, suggesting that the object has curvature away from the observer. As we can see in the right plot of Fig. 20 above, there is not a significant difference between the energy in the periphery and center of the brick wall's infrared image, suggesting that the object does not have curvature.

5. Summary and Future Work

We have developed a set of automatic algorithms that use sonar backscattering data to distinguish extended objects in the campus environment by taking a sequence of scans of each object, plotting the corresponding backscatter vs. scan angle, extracting abstract feature vectors and then categorizing them in various phase spaces. We have chosen to perform the analysis with multiple scans per object as a balance between data processing requirements and robustness of the results. Although our current robotic scanner is parked for each scan and then moves to the next scan location before scanning again, it is not difficult to envision a similar mobile robotic platform that scans continuously while moving. It could then take ten or even a hundred scans while approaching a tree or while moving along a unit of a fence, for example. Based on our experience with such scans, however, we would typically expect only the characteristic variations in backscattering behavior described above. Hence, we would envision scans taken continuously as the robot moves towards or along an object, and once the dominant features are identified, the necessary backscatter plots could be processed in the manner described in the previous sections, with the rest of the data safely purged from memory.
Our reason for performing this level of detailed processing is a scenario where an autonomous robot is trying to identify particular landmark objects, presumably under low-

light or otherwise visually obscured conditions where fences, hedges and brick walls can be visually similar. Alternatively, we envision a mobile robot with limited on board processing capability such that the visual image stream must be deliberately degraded by either reducing the number of pixels or the bits per pixel in order to have a sufficient video frame rate. In either case the extended objects considered here might appear very similar in the visual image stream. Hence, our interest is in the situation where the robot knows the obstacle is there and has already done some preliminary classification of it, but now needs a more refined answer. It could need to distinguish a fence or wall from a hedge since it could plow through hedge but would be damaged by a wrought iron fence or a brick wall. It may know it is next to a picket fence, but cannot tell whether it's on the inside or outside of the fence. Perhaps it has been given instructions to "turn left at the brick wall" and the "go beyond the big tree" but doesn't have an accurate enough map of the campus or more likely the landmark it was told to navigate via does not show up on its on-board map.

We have now added thermal infrared imaging to our mobile robots, and have begun the systematic process of identifying exploitable features. After preprocessing, feature vectors are formed to give unique representations of the signal data produced by a given object. These features are chosen to have minimal variation with changes in the viewing angle and/or distance between the object and sensor, temperature, and visibility. Fusion of the two sensor outputs then happens according to the Bayesian scheme diagrammed in Fig. 21 below, which is the focus of our ongoing work.

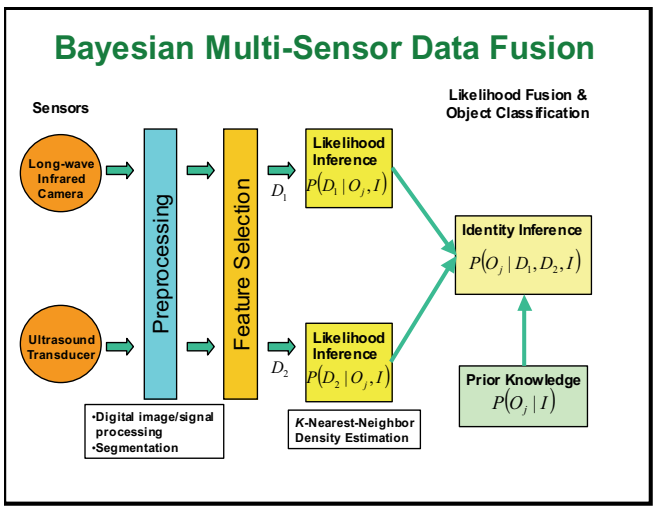


Fig. 21. Bayesian multi-sensor data fusion architecture using ultrasound and infrared sensors.

6. References

Au, W. (1993). *The Sonar of Dolphins*, Springer-Verlag, New York.

Barshan, B. & Kuc, R. (1992). A bat-like sonar system for obstacle localization, IEEE transactions on systems, *Man, and Cybernetics,* Vol.22, No. 4, July/August 1992, pp. 636-646.

Chou, T. & Wykes, C. (1999). An integrated ultrasonic system for detection, recognition and measurement, *Measurement,* Vol. 26, No. 3, October 1999, pp. 179-190.

Crowley, J. (1985). Navigation for an intelligent mobile robot, *IEEE Journal of Robotics and Automation,* vol. RA-1, No. 1, March 1985, pp. 31-41.

Dror, I.; Zagaeski, M. & Moss, C. (1995). Three-dimensional target recognition via sonar: a neural network model, *Neural Networks,* Vol. 8, No. 1, pp. 149-160, 1995.

Gao, W. (2005). *Sonar Sensor Interpretation for Ectogenous Robots,* College of William and Mary, Department of Applied Science Doctoral Dissertation, April 2005.

Gao, W. & Hinders, M.K. (2005). Mobile Robot Sonar Interpretation Algorithm for Distinguishing Trees from Poles, *Robotics and Autonomous Systems,* Vol. 53, pp. 89-98.

Gao, W. & Hinders, M.K. (2006). Mobile Robot Sonar Deformable Template Algorithm for Distinguishing Fences, Hedges and Low Walls, *Int. Journal of Robotics Research,* Vol. 25, No. 2, pp. 135-146.

Gonzalez, R.C. (2004) Gonzalez, R. E. Woods & S. L. Eddins, *Digital Image Processing using MALAB,* Pearson Education, Inc.

Griffin, D. *Listening in the Dark,* Yale University Press, New Haven, CT.

Harper, N. & McKerrow, P. (2001). Recognizing plants with ultrasonic sensing for mobile robot navigation, Robotics and Autonomous Systems, Vol.34, 2001, pp.71-82.

Jeon, H. & Kim, B. (2001). Feature-based probabilistic map building using time and amplitude information of sonar in indoor environments, *Robotica,* Vol. 19, 2001, pp. 423-437.

Kay, L. (2000). Auditory perception of objects by blind persons, using a bioacoustic high resolution air sonar, *Journal of the Acoustical Society of America* Vol. 107, No. 6, June 2000, pp. 3266-3275.

Kleeman, L. & Kuc, R. (1995). Mobile robot sonar for target localization and classification, *The International Journal of Robotics Research,* Vol. 14, No. 4, August 1995, pp. 295-318.

Leonard, J. & Durrant-Whyte, H. (1992). *Directed sonar sensing for mobile robot navigation,* Kluwer Academic Publishers, New York, 1992.

Lou & Boutell (2005). J. Luo and M. Boutell, Natural scene classification using overcomplete ICA, *Pattern Recognition,* Vol. 38, (2005) 1507-1519.

Maxim, H. (1912). Preventing Collisions at Sea. A Mechanical Application of the Bat's Sixth Sense. *Scientific American,* 27 July 1912, pp. 80-81.

McKerrow, P. & Harper, N. (1999). Recognizing leafy plants with in-air sonar, *Sensor Review,* Vol. 19, No. 3, 1999, pp. 202-206.

Rahman, Z. et al., (2002). *Multi-sensor fusion and enhancement using the Retinex image enhancement algorithm*, Proceedings of SPIE 4736 (2002) 36-44.

Ratner, D. & McKerrow, P. (2003). Navigating an outdoor robot along continuous landmarks with ultrasonic sensing, *Robotics and Autonomous Systems*, Vol. 45, 2003 pp. 73-82.

Rosenfeld, A. & Kak, A. (1982). *Digital Picture Processing*, 2nd Edition, Academic Press, Orlando, FL.

Sakai, K. & Finkel, L.H. (1995). Characterization of the spatial-frequency spectrum in the perception of shape from texture, *Journal of the Optical Society of America,* Vol. 12, No. 6, June 1995, 1208-1224.

Theodoridis, S. & Koutroumbas, K. (1998). *Pattern Recognition,* Academic Press, New York.

Tou, J. (1968). Feature extraction in pattern recognition, *Pattern Recognition,* Vol. 1, No. 1, July 1968, pp. 3-11.

5

Obstacle Detection Based on Fusion Between Stereovision and 2D Laser Scanner

Raphaël Labayrade, Dominique Gruyer, Cyril Royere,
Mathias Perrollaz, Didier Aubert
LIVIC (INRETS-LCPC)
France

1. Introduction

Obstacle detection is an essential task for mobile robots. This subject has been investigated for many years by researchers and a lot of obstacle detection systems have been proposed so far. Yet designing an accurate and totally robust and reliable system remains a challenging task, above all in outdoor environments. The DARPA Grand Challenge (Darpa, 2005) proposed efficient systems based on sensors redundancy, but these systems are expensive since they include a large set of sensors and computers: one can not consider to implement such systems on low cost robots. Thus, a new challenge is to reduce the number of sensors used while maintaining a high level of performances. Then, many applications will become possible, such as Advance Driving Assistance Systems (ADAS) in the context of Intelligent Transportation Systems (ITS).

Thus, the purpose of this chapter is to present new techniques and tools to design an accurate, robust and reliable obstacle detection system in outdoor environments based on a minimal number of sensors. So far, experiments and assessments of already developed systems show that using a single sensor is not enough to meet the requirements: at least two complementary sensors are needed. In this chapter a stereovision sensor and a 2D laser scanner are considered.

In Section 2, the ITS background under which the proposed approaches have been developed is introduced. The remaining of the chapter is dedicated to technical aspects. Section 3 deals with the stereovision framework: it is based on a new technique (the so-called "v-disparity" approach) that efficiently tackles most of the problems usually met when using stereovision-based algorithms for detecting obstacles. This technique makes few assumptions about the environment and allows a generic detection of any kind of obstacles; it is robust against adverse lightning and meteorological conditions and presents a low sensitivity towards false matches. Target generation and characterization are detailed. Section 4 focus on the laser scanner raw data processing performed to generate targets from lasers points and estimate their positions, sizes and orientations. Once targets have been generated, a multi-objects association algorithm is needed to estimate the dynamic state of the objects and to monitor appearance and disappearance of tracks. Section 5 intends to present such an algorithm based on the Dempster-Shaffer belief theory. Section 6 is about fusion between stereovision and laser scanner. Different possible fusion schemes are introduced and discussed. Section 7 is dedicated to experimental results. Eventually, section 8 deals with trends and future research.

2. Intelligent Transportation Systems Background

In the context of Intelligent Transportation Systems and Advanced Driving Assistance Systems (ADAS), onboard obstacle detection is a critical task. It must be performed in real time, robustly and accurately, without any false alarm and with a very low (ideally nil) detection failure rate. First, obstacles must be detected and positioned in space; additional information such as height, width and depth can be interesting in order to classify obstacles (pedestrian, car, truck, motorbike, etc.) and predict their dynamic evolution. Many applications aimed at improving road safety could be designed on the basis of such a reliable perception system: Adaptative Cruise Control (ACC), Stop'n'Go, Emergency braking, Collision Mitigation. Various operating modes can be introduced for any of these applications, from the *instrumented mode* that only informs the driver of the presence and position of obstacles, to the *regulated mode* that take control of the vehicle through activators (brake, throttle, steering wheel). The *warning mode* is an intermediate interesting mode that warn the driver of an hazard and is intended to alert the driver in advance to start a manoeuver before the accident occurs.

Various sensors can be used to perform obstacle detection. 2D laser scanner (Mendes 2004) provides centimetric positioning but some false alarms can occur because of the dynamic pitching of the vehicle (from time to time, the laser plane collides with the ground surface and then laser points should not be considered to belong to an obstacle). Moreover, width and depth (when the side of the object is visible) of obstacles can be estimated but height cannot. Stereovision can also be used for obstacle detection (Bertozzi, 1998 ; Koller, 1994 ; Franke, 2000 ; Williamson, 1998). Using stereovision, height and width of obstacles can be evaluated. The pitch value can also be estimated. However, positioning and width evaluation are less precise than the ones provided by laser scanner.

Fusion algorithms have been proposed to detect obstacles using various sensors at the same time (Gavrila, 2001 ; Mobus, 2004 ; Steux, 2002). The remaining of the chapter presents tools designed to perform fusion between 2D laser scanner and stereovision that takes into account their complementary features.

3. Stereovision Framework

3.1 The "v-disparity" framework

This section deals with the stereovision framework. Firstly a modeling of the stereo sensor, of the ground and of the obstacles is presented. Secondly details about a possible implementation are given.

Modeling of the stereo sensor: The two image planes of the stereo sensor are supposed to belong to the same plane and are at the same height above the ground (see Fig. 1). This camera geometry means that the epipolar lines are parallel. The parameters shown on Fig. 1 are:

- θ s the angle between the optical axis of the cameras and the horizontal,
- h is the height of the cameras above the ground,
- b is the distance between the cameras (i.e. the stereoscopic base).

(R_a) is the absolute coordinate system, and O_a lies on the ground. In the camera coordinate system (R_{ci}) (i equals *l (left)* or *r (right)*), the position of a point in the image plane is given by its coordinates (u_i,v_i). The image coordinates of the projection of the optical center will be denoted by (u_0,v_0), assumed to be at the center of the image. The intrinsic parameters of the camera are f (the focal length of the lens), t_u and t_v (the size of pixels in u and v). We also use $a_u=f/t_u$ and $a_v=f/t_v$. With the cameras in current use we can make the following approximation: $\alpha_u \approx \alpha_v = \alpha$.

Using the pin-hole camera model, a projection on the image plane of a point P(X,Y,Z) in (R_a) is expressed by:

$$\begin{cases} u = \alpha \dfrac{X}{Z} + u_0 \\ v = \alpha \dfrac{Y}{Z} + v_0 \end{cases} \quad (1)$$

On the basis of Fig. 1, the transformation from the absolute coordinate system to the right camera coordinate system is achieved by the combination of a vector translation ($\vec{t} = -h\vec{Y}$ and $\vec{b} = (b/2)\vec{X}$) and a rotation around $\vec{X}$, by an angle of $-\theta$. The combination of a vector translation ($\vec{t} = -h\vec{Y}$ and $\vec{b} = -(b/2)\vec{X}$) and a rotation around $\vec{X}$, by an angle of $-\theta$ is the transformation from the absolute coordinate system to the left camera coordinate system.

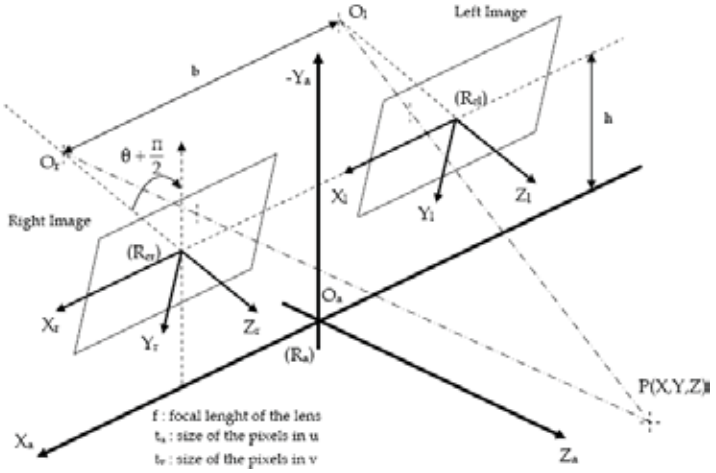


Fig. 1. The stereoscopic sensor and used coordinate systems.

Since the epipolar lines are parallel, the ordinate of the projection of the point P on the left or right image is $v_r = v_l = v$, where:

$$v = \frac{\left[v_0 \sin\theta + \alpha\cos\theta\right](Y+h) + \left[v_0\cos\theta - \alpha\sin\theta\right]Z}{(Y+h)\sin\theta + Z\cos\theta} \quad (2)$$

Moreover, the disparity Δ of the point P is:

$$\Delta = u_l - u_r = \frac{\alpha b}{(Y+h)\sin\theta + Z\cos\theta} \quad (3)$$

Modeling of the ground: In what follows the ground is modeled as a plane with equation: $Z=aY+d$. If the ground is horizontal, the plane to consider is the plane with equation $Y=0$.
Modeling of the obstacles: In what follows any obstacle is characterized by a vertical plane with equation $Z = d$.
Thus, all planes of interest (ground and obstacles) can be characterized by a single equation: $Z = aY+d$.

The image of planes of interest in the "v-disparity" image: From (2) and (3), the plane with the equation $Z = aY+d$ in (R_a) is projected along the straight line of equation (1) in the "v-disparity" image:

$$\Delta_M = \frac{b}{ah-d}(v-v_0)(a\cos\theta+\sin\theta)+\frac{b}{ah-d}\alpha(a\sin\theta+\cos\theta) \tag{4}$$

N.B.: when $a = 0$ in equation (1), the equation for the projection of the vertical plane with the equation $Z = d$ is obtained:

$$\Delta_M = \frac{b}{d}(v_0-v)\sin\theta+\frac{b}{d}\alpha\cos\theta \tag{5}$$

When $a\to\infty$, the equation of the projection of the horizontal plane with the equation $Y = 0$ is obtained:

$$\Delta_M = \frac{b}{h}(v-v_0)\cos\theta+\frac{b}{h}\alpha\sin\theta \tag{6}$$

Thus, planes of interest are all projected as straight lines in the "v-disparity" image.
The "v-disparity" framework can be generalized to extract planes presenting roll with respect to the stereoscopic sensor. This extension allows to extract any plane in the scene. More details are given in (Labayrade, 2003 a).

3.2 Exemple of implementation

"v-disparity" image construction: A disparity map is supposed to have been computed from the stereo image pair (see Fig. 2 left). This disparity map is computed taking into account the epipolar geometry; for instance the primitives used can be horizontal local maxima of the gradient; matching can be local and based on normalized correlation around the local maxima (in order to obtain additional robustness with respect to global illumination changes).
The "v-disparity" image is line by line the histogram of the occurring disparities (see Fig. 2 right). In what follows it will be denoted as $\boldsymbol{I_{v\Delta}}$.
Case of a flat-earth ground geometry: robust determination of the plane of the ground: Since the obstacles are defined as objects located above the ground surface, the corresponding surface must be estimated before performing obstacle detection.

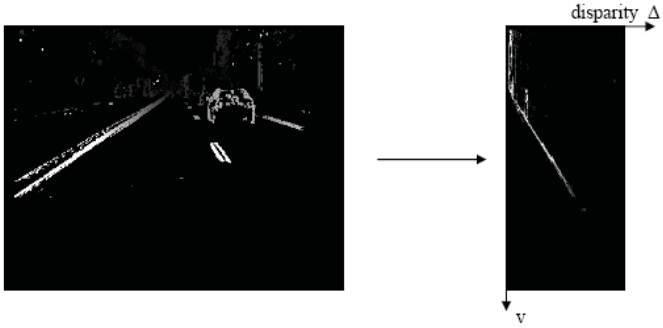


Fig. 2. Construction of the grey level "v-disparity" image from the disparity map. All the pixels from the disparity map are accumulated along scanning lines.

When the ground is planar, with for instance the following mean parameter values of the stereo sensor:

- θ = 8.5°,
- h = 1.4 m,
- b = 1 m,

the plane of the ground is projected in $I_{v\Delta}$ as a straight line with mean slope 0.70. The longitudinal profile of the ground is therefore a straight line in $I_{v\Delta}$. Robust detection of this straight line can be achieved by applying a robust 2D processing to $I_{v\Delta}$. The Hough transform can be used for example.

Case of a non flat-earth ground geometry: The ground is modeled as a succession of parts of planes. As a matter of fact, its projection in $I_{v\Delta}$ is a piecewise linear curve. Computing the longitudinal profile of the ground is then a question of extracting a piecewise linear curve in $I_{v\Delta}$. Any robust 2D processing can be used. For instance it is still possible to use the Hough Transform. The k highest Hough Transform values are retained (k can be taken equal to 5) and correspond to k straight lines in $I_{v\Delta}$. The piecewise linear curve researched is either the upper (when approaching a downhill gradient) or the lower (when approaching a uphill gradient) envelope of the family of the k straight lines generated. To choose between these two envelope, the following process ca be performed. $I_{v\Delta}$ is investigated along both curves extracted and a score is computed for each: for each pixel on the curve, the corresponding grey level in $I_{v\Delta}$ is accumulated. The curve is chosen with respect to the best score obtained. Fig. 3 shows how this curve is extracted. From left to right the following images are presented: an image of the stereo pair corresponding to a non flat ground geometry when approaching an uphill gradient; the corresponding $I_{v\Delta}$ image; the associated Hough Transform image (the white rectangle show the research area of the k highest values); the set of the k straight lines generated; the computed envelopes, and the resulting ground profile extracted.

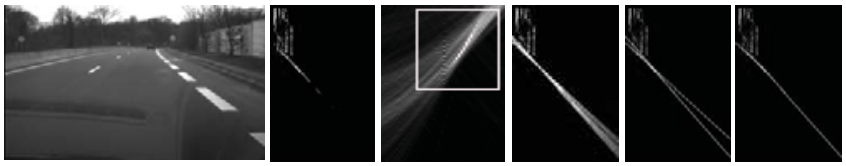


Fig. 3. Extracting the longitudinal profile of the ground in the case of a non planar geometry (see in text for details).

Evaluation of the obstacle position and height: With the mean parameter values of the stereo sensor given above for example, the plane of an obstacle is projected in $I_{v\Delta}$ as a straight line nearly vertical above the previously extracted ground surface. Thus, the extraction of vertical straight lines in $I_{v\Delta}$ is equivalent to the detection of obstacles. In this purpose, an histogram that accumulates all the grey values of the pixels for each column of the $I_{v\Delta}$ image can be built; then maxima in this histogram are looked for. It is then possible to compute the ordinate of the contact point between the obstacle and the ground surface (intersection between the ground profile and the obstacle line in the "v-disparity" image, see Fig. 4). The distance D between the vehicle and the obstacle is then given by:

$$D = \frac{b(\alpha \cos\theta - (v_r - v_0)\sin\theta)}{\Delta} \tag{7}$$

where v_r is the ordinate of the ground-obstacle contact line in the image.
The height of the obstacle is given by the height of the straight line segment in the "v-disparity" image (see Fig. 4). The lateral position and left and right border of the obstacle can be estimated by similar processing in the "u-disparity" image (The "u-disparity" image is column by column the histogram of the occurring disparities). Thus, a target detected by stereovision is characterized by its (X,Z) coordinates, its height and its width.
Moreover, a dynamic estimation of the sensor pitch θ can be obtained from the horizon line, at each fame processed:

$$\theta = \arctan\left(\frac{v_0 - v_{hor}}{\alpha}\right) \tag{8}$$

where v_{hor} is the ordinate of the horizon line. Since the horizon line belongs to the ground surface and is located at infinite distance (which corresponds to nil disparity), v_{hor} is the ordinate of the point located on the ground profile for a nil disparity (see Fig. 4).

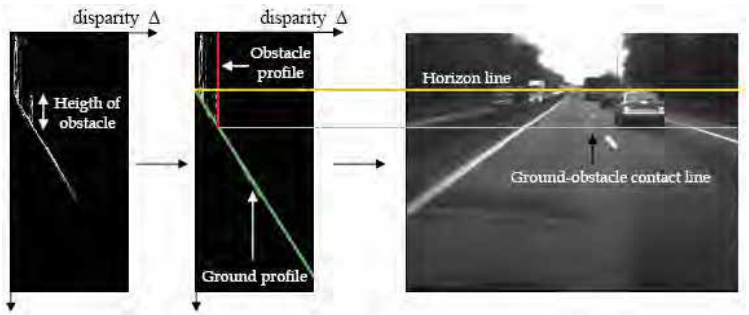


Fig. 4. Extracting obstacles and deducing obstacle-ground contact line and horizon line.

Practical good properties of the algorithm: It should be noticed that the algorithm is able to detect any kind of obstacles. Furthermore, all the information in the disparity map is exploited and the accumulation performed increases the density of the alignments in $I_{v\Delta}$. Any matching error that occur when the disparity map is computed causes few problems as the probability that the points involved will generate coincidental alignments in $I_{v\Delta}$ is low. As a matter of fact, the algorithm is able to perform accurate detection even in the event of a lot of noise or matching errors, and when there is only a few correct matches or a few amount of correct data in the images: in particular in night condition when the majority of the pixels are very dark. Eventually, the algorithm works whatever the process used for computing the disparity map (see (Scharstein, 2001)) or for processing the "v-disparity" image. Eventually, as detailed in (Labayrade, 2003 b), it is possible in a two-stages process to improve the disparity map and remove a lot of false matches.

4. Laser Scanner Raw Data Processing

The 2D laser scanner provides a set of laser impacts on the scanned plane: each laser point is characterized by an incidence angle and a distance which corresponds to the distance of the nearest object in this direction (see Fig. 6). From these data, a set of clusters must be built, each cluster corresponding to an object in the observed scene.

Initially, the first laser impact defines the first cluster. For all other laser points, the goal is to know if they are a membership of the existent cluster or if they belong to a new cluster. In the literature, a great set of distance functions can be found for this purpose. The chosen distance $D_{i,j}$ must comply with the following criteria

Firstly, this function $D_{i,j}$ must give a result scaled between 0 and 1. The value 0 indicates that the measurement i is a member of the cluster j,

Secondly, the result must be above 1 if the measurement is out of the cluster j,

Finally, this distance must have the properties of the distance functions.

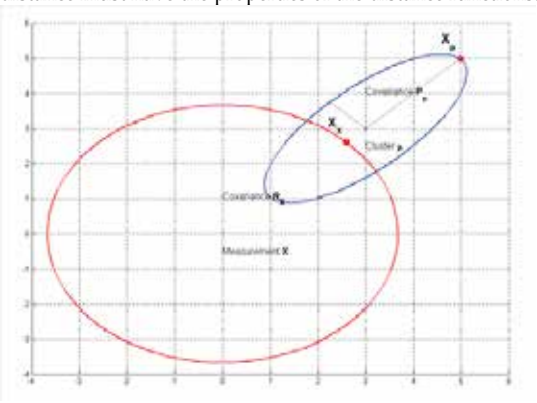


Fig. 5. Clustering of a measurement.

The distance function must also use both cluster and measurement covariance matrices. Basically, the chosen function computes an inner distance with a normalisation part build from the sum of the outer distances of a cluster and a measurement. Only the outer distance uses the covariance matrix:

$$D_{i,j} = \frac{\sqrt{(X-\mu)(X-\mu)'}}{\sqrt{(X_\mu-\mu)}+\sqrt{(X_X-X)}} \tag{9}$$

In the normalisation part, the point X_μ represent the border point of a cluster i (centre μ). This point is localised on the straight line between the cluster i (centre μ) and the measurement j (centre X). The same border measurement is used with the cluster. The computation of X_μ and X_X is made with the covariance matrices R_x and P_μ. P_μ and R_x are respectively the cluster covariance matrix and the measurement covariance matrix. The measurement covariance matrix is given from its polar covariance representation (Blackman 1999) with ρ_0 the distance and θ_0 the angle:

$$R_x = \begin{bmatrix} \sigma^2_{x_0} & \sigma^2_{x_0,y_0} \\ \sigma^2_{x_0,y_0} & \sigma^2_{y_0} \end{bmatrix} \tag{10}$$

where, using a first order expansion:

$$\begin{aligned} \sigma^2_{x_0} &= \sigma^2_{\rho_0}\cos^2\theta_0 + \sigma^2_{\theta_0}\rho_0{}^2\sin^2\theta_0 \\ \sigma^2_{y_0} &= \sigma^2_{\rho_0}\sin^2\theta_0 + \sigma^2_{\theta_0}\rho_0{}^2\cos^2\theta_0 \end{aligned} \tag{11}$$

$$\sigma^2_{x_0 y_0} = \frac{1}{2}\sin 2\theta_0 \left[\sigma^2_{\rho_0} - \sigma^2_{\theta_0}\rho_0{}^2\right]$$

$\sigma^2_{\rho_0}$ and $\sigma^2_{\theta_0}$ are the variances in both distance and angle of each measurement provided by the laser scanner. From this covariance matrix, the eigenvalues σ and the eigenvectors V are extracted. A set of equations for both ellipsoid cluster and measurement modelling and line between the cluster centre μ and the laser measurement X is then deduced:

$$\begin{aligned} x &= V_{11}\sqrt{\sigma_1{}^2}\cos\Psi + V_{12}\sqrt{\sigma_2{}^2}\sin\Psi \\ y &= V_{21}\sqrt{\sigma_1{}^2}\cos\Psi + V_{22}\sqrt{\sigma_2{}^2}\sin\Psi \\ y &= ax + b \end{aligned} \tag{12}$$

The solution of this set of equations gives:

$$\Psi = \arctan\left(\frac{-\sqrt{\sigma_1{}^2}\left[V_{2,1} - aV_{1,1}\right]}{\sqrt{\sigma_2{}^2}\left[V_{2,2} - aV_{1,2}\right]}\right)$$

$$\Psi \in \left[\frac{-\pi}{2}, \frac{\pi}{2}\right] \quad \text{with} \tag{13}$$

From (13), two solutions are possible:

$$\begin{aligned} X_\mu &= P_\mu\sqrt{\sigma^2}\begin{bmatrix}\cos\Psi \\ \sin\Psi\end{bmatrix} \\ X_\mu &= P_\mu\sqrt{\sigma^2}\begin{bmatrix}\cos\Psi + \pi \\ \sin\Psi + \pi\end{bmatrix} \end{aligned} \tag{14}$$

and

Then equation (9) is used with X_μ to know if a laser point belongs to a cluster.

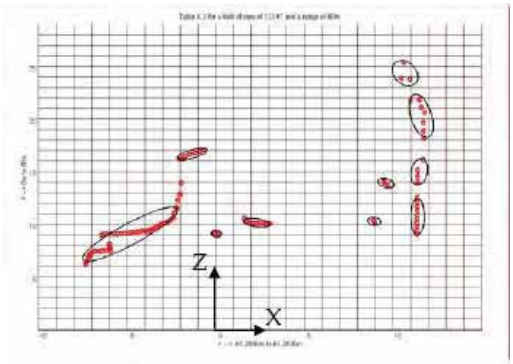


Fig. 6. Example of a result of autonomous clustering (a laser point is symbolized by a little circle, and a cluster is symbolized by a black ellipse).

Fig. 5 gives a visual interpretation of the used distance for the clustering process. Fig. 6 gives an example of a result of autonomous clustering from laser scanner data. Each cluster is characterized by its position, its orientation, and its size along the two axes.

5. Multi-Objects Association

Once targets have been generated from stereovision or from laser scanner, a multi-objects association algorithm is needed to estimate the dynamic state of the obstacles and to monitor appearance and disappearance of tracks. The position of previously perceived objects is predicted at the current time using Kalman Filtering. These predicted objects are already known objects and will be denoted in what follows by Y_j. Perceived objects at the current time will be denoted by X_i. The proposed multi-objects association algorithm is based on the belief theory introduced by Shafer (Shafer, 1976).

5.1 Generalities

In a general framework, the problem consist in identifying an object designated by a generic variable X among a set of hypotheses Yi. One of these hypotheses is supposed to be the solution. The current problem consists in associating perceived objects X_i to known objects Y_j. Belief theory allows to assess the veracity of P_i propositions representing the matching of the different objects.

A magnitude allowing the characterization of a proposition must be defined. This magnitude is the basic belief assignment (mass $m_\Theta(\ )$) defined on [0,1]. This mass is very close to the probabilistic mass with the difference that it is not only shared on single elements but on all elements of the definition referential $2^\Theta = \{ A/A \subseteq \Theta\} = \{\varnothing, Y_1, Y_2, ..., Y_n, Y_1 \cup Y_2, \ldots, \Theta\}$. This referential is built through the frame of discernment $\Theta = \{Y_1, Y_2, \cdots, Y_n\}$, which regroups all admissible hypotheses, that in addition must be exclusive. ($Y_i \cap Y_j = \varnothing$, $\forall$ i ≠ j). This distribution is a function of the knowledge about the source to model. The whole mass obtained is called "*basic belief assignment*". The sum of these masses is equal to 1 and the mass corresponding to the impossible case $m(\varnothing)$ must be equal to 0.

5.2. Generalized combination and multi-objects association

In order to succeed in generalizing the Dempster combination rule and thus reducing its combinatorial complexity, the reference frame of definition is limited with the constraint that a perceived object can be connected with one and only one known object.

For example, for a detected object, in order to associate among three known objects, frame of discernment is:

$$\Theta = \{Y_1, Y_2, Y_3, *\}$$
$$\text{where } Y_i \text{ means that "X and } Y_i \text{ are supposed to be the same object"}$$

In order to be sure that the frame of discernment is really exhaustive, a last hypothesis noted "*" is added. This one can be interpreted as an association of a perceived object with any of the known objects. In fact each Y_j represents a local view of the world and the "*" represents the rest of the world. In this context, "*" means well: "*an object is associated with nothing in the local knowledge set*".

The total ignorance is represented by the hypothesis Θ which is the disjunction of all the hypotheses of the frame of discernment. The conflict is given by the hypothesis $\varnothing$

which corresponds to the empty set (since the hypotheses are exclusive, their intersections is empty).

A distribution of masses made up of the masses is obtained:

$m_{i,j}(Y_j)$: mass associated with the proposition « X_i and Y_j. are supposed to be the same object »,

$m_{i,j}(\overline{Y_j})$: mass associated with the proposition « X_i and Y_j. are not supposed to be the same object »,

$m_{i,j}(\Theta_{i,j})$: mass representing ignorance,

$m_{i,\cdot}(*)$: mass representing the reject: X_i is in relation with nothing.

In this mass distribution, the first index i denotes the processed perceived objects and the second index j the known objects (predictions). If one index is replaced by a dot, then the mass is applied to all perceived or known objects according to the location of this dot.

Moreover, if an iterative combination is used, the mass $m_{i,\cdot}(*)$ is not part of the initial mass set and appears only after the first combination. It replaces the conjunction of the combined masses $m_{i,j}(\overline{Y_j})$. By observing the behaviour of the iterative combination with n mass sets, a general behaviour can be seen which enables to express the final mass set according to the initial mass sets. This enables to compute directly the final masses without any recurrent stage. For the construction of these combination rules, the work and a first formalism given in (Rombaut, 1998) is used. The use of an initial mass set generator using the strong hypothesis: "*an object can not be in the same time associated and not associated to another object*" allows to obtain new rules. These rules firstly reduce the influence of the conflict (the combination of two identical mass sets will not produce a conflict) and secondly the complexity of the combination. The rules become:

$$m_{i,\cdot}(Y_i) = 0 \quad if\ H_1 \tag{15}$$

$$m_{i,\cdot}(Y_j) = K_{i,\cdot} \cdot m_{i,j}(Y_j) \cdot E_1 \quad if\ H_2 \tag{16}$$

with

$$\begin{cases} E_1 = \prod_{\substack{k=1\cdots n \\ k\neq j}} \left(1 - m_{i,k}(Y_k)\right) \\ H_1 \Leftrightarrow \exists j, m_{i,j}(Y_j) = 0 \\ H_2 \Leftrightarrow \forall j, m_{i,j}(Y_j) \neq 0 \end{cases} \tag{17}$$

$$m_{i,\cdot}(*) = 0 \quad if\ H_1 \tag{18}$$

$$m_{i,\cdot}(*) = K_{i,\cdot} \prod_{j=1\cdots n} m_{i,j}(\overline{Y_j}) \quad if\ H_2 \tag{19}$$

with

$$\begin{cases} H_1 \Leftrightarrow \exists j, m_{i,j}(Y_j) \neq 0 \\ H_2 \Leftrightarrow \forall j, m_{i,j}(Y_j) = 0 \end{cases} \tag{20}$$

$$m_{i,\cdot}(\theta) = K_{i,\cdot} \cdot E_1 \cdot E_2 \quad if\ H_1 \tag{21}$$

$$m_{i,\cdot}(\theta) = K_{i,\cdot} \cdot E_1 \quad if\ H_2 \tag{22}$$

$$m_{i,\cdot}(\theta) = K_{i,\cdot} \cdot (E_3 - E_4) \quad if\ H_3 \tag{23}$$

$$\text{with} \quad \begin{cases} E_1 = \prod_{l=1\cdots n} m_{i,l}(\theta_{i,l}) \\ E_2 = \prod_{\substack{j=1\cdots n \\ j\neq l}} \left(m_{i,j}(\theta_{i,j}) - m_{i,j}(\overline{Y}_j)\right) \\ E_3 = \prod_{j=1\cdots n} \left(m_{i,j}(\theta_{i,j}) - m_{i,j}(\overline{Y}_j)\right) \\ E_4 = \prod_{j=1\cdots n} m_{i,j}(\overline{Y}_j) \\ H_1 \Leftrightarrow \exists l \in [1\cdots n], \quad m_{i,l}(Y_l) \neq 0 \\ H_2 \Leftrightarrow \forall j \in [1\cdots n], \quad m_{i,j}(Y_j) \neq 0 \\ H_3 \Leftrightarrow \forall j \in [1\cdots n], \quad m_{i,j}(Y_j) = 0 \end{cases} \tag{24}$$

$$K_{i,\bullet} = \frac{1}{1 - m_{i,\bullet}(\varnothing)} = \frac{1}{E_1 + E_2} \tag{25}$$

$$\text{with} \quad \begin{cases} E_1 = \prod_{j=1\cdots n} \left(1 - m_{i,j}(Y_j)\right) \\ E_2 = \sum_{j=1\cdots n} m_{i,j}(Y_j) \cdot \prod_{\substack{k=1\cdots n \\ k\neq j}} \left(1 - m_{i,k}(Y_k)\right) \end{cases} \tag{26}$$

From each mass set, two matrices $\mathbf{M}^{cr}_{i,\bullet}$ and $\mathbf{M}^{cr}_{\bullet,j}$ are built which give the belief that a perceived object is associated with a known object and conversely. The sum of the elements of each column is equal to 1 because of the re-normalization.
The resulting frames of discernment are:

$$\Theta_{\bullet,j} = \{Y_{1,j}, Y_{2,j}, \cdots, Y_{n,j}, Y_{*,j}\}$$

$$\text{and} \quad \Theta_{i,\bullet} = \{X_{i,1}, X_{i,2}, \cdots, X_{i,m}, X_{i,*}\}$$

The first index represents the perceived object and the second index the known object. The index "*" is the notion of "emptiness" or more explicitly "*nothing*". With this hypothesis, it can be deduced if an object has appeared or disappeared.
The following stage consists in establishing the best decision on association using these two matrices obtained previously. Since a referential of definition built with singleton hypotheses is used, except for Θ and *, the use of credibilistic measure will not add any useful information. This redistribution will simply reinforce the fact that a perceived object is really in relation with a known object. This is why the maximum of belief on each column of the two belief matrices is used as the decision criterion:

$$d(Y_{i,\bullet}) = \underset{j}{Max}[M^{Cr}_{i,j}] \tag{27}$$

This rule answers the question "*which is the known object Y_j in relation with the perceived object X_i*"? The same rule is available for the known objects:

$$d(X_{\bullet,j}) = \underset{i}{Max}[M^{Cr}_{i,j}] \tag{28}$$

Unfortunately, a problem appears when the decision obtained from a matrix is ambiguous (this ambiguity quantifies the duality and the uncertainty of a relation) or when the decisions between the two belief matrices are in conflict (this conflict represents antagonism between two relations resulting each one from a different belief matrix). Both problems of

conflicts and ambiguities are solved by using an assignment algorithm known under the name of the Hungarian algorithm (Kuhn, 1955 ; Ahuja, 1993). This algorithm has the advantage of ensuring that the decision taken is not " *good*" but "*the best*". By the "*best*", we mean that if a known object has some defective or poor sensor to perceive it, then it is unlikely to know what this object corresponds to, and therefore ensuring that the association is good is a difficult task. But among all the available possibilities, we must certify that the decision is the "*best*" of all possible decisions.
Once the multi-objects association has been performed, the Kalman filter associated to each object is updated using the new position of the object, and so the dynamic state of each object is estimated.

6. Fusion

So far, the chapter has described the way in which the two sensors (stereovision and 2D laser scanner) are independently used to perform obstacle detection. Tables 1 and 2 remind the advantages and drawbacks of each sensor.

	Detection range	**Obstacle position accuracy**	**Frequency**	**False alarms occurrence**
Stereovision	Short to medium range (up to 50 m).	Decreases when the obstacle distance increases.	Video frame rate.	When the disparity map is of poor quality.
Laser scanner	Medium to long range (up to 120 m).	Usually a few cm. Independent to the obstacle distance.	Usually higher than the stereovision.	When the laser plane collides with the ground surface.

Table 1. Features of the stereovision and 2D laser scanner sensors.

	Detection failure occurrence	**Ground geometry**	**Width, height, depth, orientation**
Stereovision	Adverse lighting conditions, very low obstacles (<30 cm).	Provide ground geometry, including roll, pitch, longitudinal profile.	Provide width and height
Laser Scanner	When the laser plane passes above obstacle.	Cannot provide ground geometry.	Provide orientation, width and depth (when the side of the obstacle is visible)

Table 2. Features of the stereovision and 2D laser scanner sensors (continued).

From Tables 1 and 2, some remarks can be made. Laser scanner and stereovision are complementary sensors: laser scanner is more accurate but a lot of false alarms can occur when the laser plane collides with the ground (see Fig. 7); stereovision is less accurate but can distinguish the ground from an obstacle, because it can provide a 3D modelling of the scene. The question is then to know how the data provided by stereovision and laser scanner can be combined and/or fused together in order to obtain the best results.

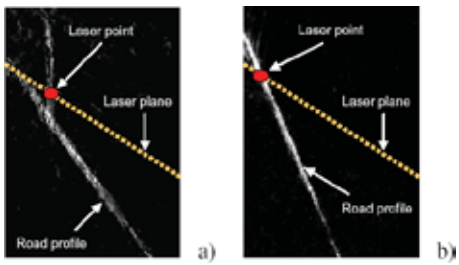


Fig. 7. "v-*disparity*" view of the laser scanner plane. a) An obstacle is detected. b) The ground is viewed as an obstacle, due to sensor pitch.

In this section we discuss several possible cooperative fusion schemes.

6.1 Laser scanner raw data filtering and clustering

The idea is here to use the geometric description of the ground provided by stereovision in order to filter the laser raw data or clustered objects which could be the result of the collision of the laser plane with the ground surface.

Two possibilities are available :

. Strategy 1: firstly, remove laser points that could be the result of the collision of the laser plane with the ground surface from the laser raw data; secondly, cluster laser points from the filtered raw data (see Fig. 8),

. Strategy 2: firstly, cluster impacts from the laser raw data; secondly, remove clustered objects that collide partially or totally the ground surface (see Fig. 9).

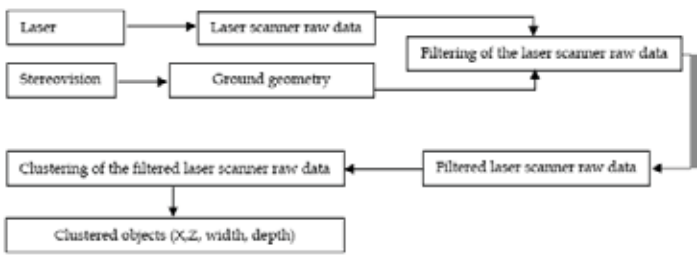


Fig. 8. Laser scanner raw data filtering and clustering. Strategy 1.

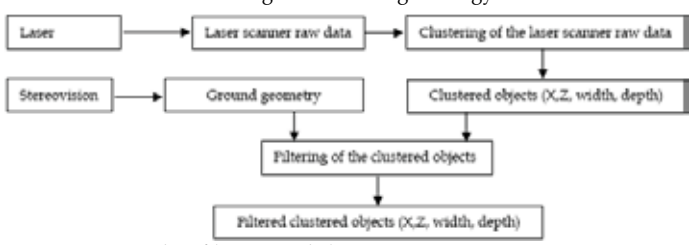


Fig. 9. Laser scanner raw data filtering and clustering. Strategy 2.

6.2 Simple redundant fusion

At this step, filtered objects from laser scanner and stereovision are available. The idea of the first fusion strategy is very simple. It consists in introducing redundancy by matching the set of obstacles detected by stereovision with the set of obstacles detected by laser scanner. If an obstacle detected by laser scanner is located at the same position than an obstacle detected by stereovision, this obstacle is supposed to be real, otherwise it is removed from the set of obstacles (see Fig. 10). However this scheme provide no information about the dynamic states (velocities, etc.) of the obstacles.

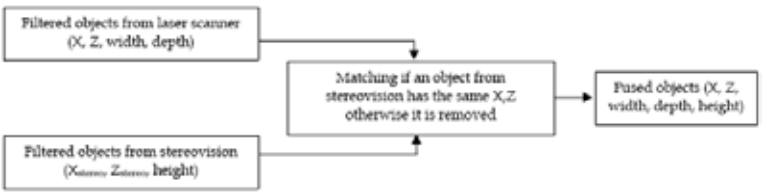


Fig. 10. Simple redundant fusion.

6.3 Fusion with global association

More complex strategies consist in introducing global association, using the algorithm presented in section 5. The idea consists in: a) performing multi-obstacles tracking and association for each sensor in order to obtain multi-tracks for each sensor; b) performing multi-track association between the tracks from the stereovision and the tracks from the laser scanner; c) fusing the tracks together in order to increase their certainty. Fig. 11 presents a fusion scheme including tracking and association for both stereovision and laser scanner sensor, and global fusion.

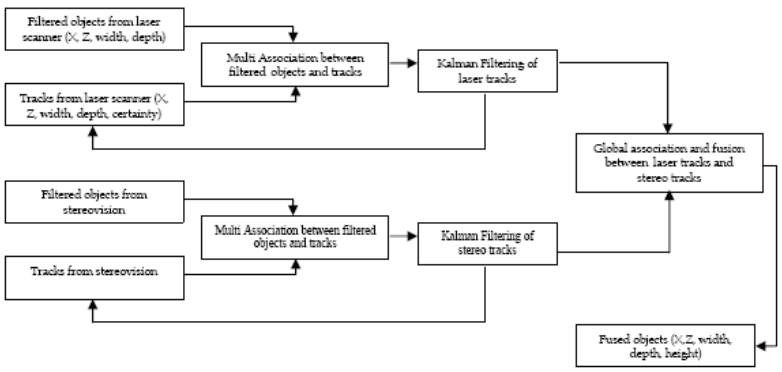


Fig. 11. Fusion with tracking and global association.

From our experiments, it seems that the tracking is difficult to perform for the stereovision tracks when the obstacles are beyond 15 meters, because of the unaccuracy of the positioning provided by the stereovision and resulting in noisy speed used in the linear

Kalman filter. Another strategy is then to perform multi-obstacles tracking and association for the single laser scanner, and then to check whether an obstacle has been detected by the stereovision at the tracked positions. If so, the certainty about the track is increased. Fig. 12 presents the corresponding fusion scheme.
Fig. 13 shows another scheme which consists in using the stereovision only to confirm the existence of an obstacle tracked by the laser scanner: the stereovision detection is performed only at the positions corresponding to objects detected by the laser scanner, in order to save computational time (indeed the stereovision will only be performed in the part of the image corresponding to the position of obstacles detected by laser scanner). Then the existence of an obstacle is confirmed if the stereovision detects an obstacle at the corresponding position. This scheme presents the advantage to work with complex ground geometry since this geometry can be estimated locally around the position of the tracked laser objects.
For each fusion scheme, the resulting positioning of each obstacle is the centimetric positioning provided by laser scanner. The estimated velocity is estimated through a linear Kalman filter applied on laser clustered data. Orientation, width and depth come from laser scanner, and height comes from stereovision.

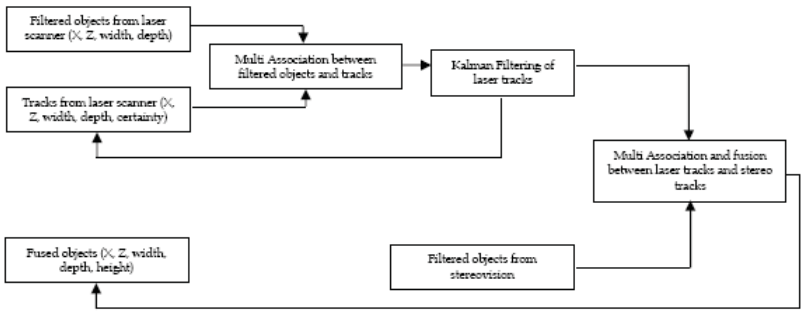


Fig. 12. Fusion with tracking of laser objects.

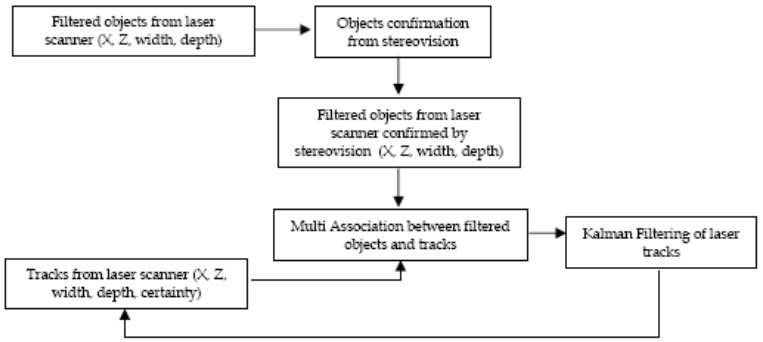


Fig. 13. Fusion with laser scanner tracking and confirmation by stereovision.

6.5 Stereovision-based obstacle confirmation criteria

To confirm the existence of an obstacle in a region of interest given by the projection of a laser-tracked object onto the image, three approaches can be used.

Number of obstacle-pixels: The first approach consists in classifying the pixels of the region of interest. A local ground profile is first extracted using the "v-disparity" iamge. Afterwards, the (u_r,Δ,v) coordinates of each pixel are analyzed to determine whether it belongs to the ground surface. If not, the pixel is classified as an obstacle-pixel. At the end of this process, every pixel in the region of interest has been classified as ground or obstacle. The number of obstacle-pixels gives a confidence on the existence of an object over the ground surface. Therefore, an obstacle is confirmed if the confidence is above a threshold.

The obstacle-pixels criterion has the advantage to avoid any assumption on the obstacles to detect. Moreover, this method gives a confidence, in an intuitive way. However, as it considers each pixel individually, it can be strongly influenced by errors in the disparity map.

Prevailing alignment orientation: Assuming that the obstacles are seen as vertical planes by the stereoscopic sensor, an other confirmation criterion can be defined (Fig. 4 and 7 a). The prevailing alignment of pixels in the local "v-disparity" image is extracted using the Hough transform. The confirmation of the track depends on the orientation of this alignment: a quite vertical alignment corresponds to an obstacle. Other alignments correspond to the ground surface. The Prevailing Alignment criterion relies on a global approach in the region of interest (alignment seeking). This makes it more robust with respect to the errors in the disparity map.

Laser points altitude: Many false detections are due to the intersection of the laser plane with the ground (see Fig. 4). The knowledge of the longitudinal ground geometry allows to deal with such errors. Therefore, the local profile of the ground is estimated through "v-disparity" framework. The altitude of the laser points is then compared to the altitude of the local ground surface. An obstacle is confirmed if this altitude is high enough.

7. Experimental Results

7.1 Experimental protocol

The algorithm has been implemented on one of the experimental vehicle of LIVIC to assess their behaviour in real conditions. The stereoscopic sensor is composed of two Sony™ 8500C cameras featuring Computar™ Auto Iris 8.5 mm focal length. Quarter PAL 8 bits gray-scale images are grabbed every 40 ms. The baseline is b = 1 m, the height h = 1.4 m and the pitch θ = 5°. The laser sensor is a Sick™ scanner which measures 201 points every 26 ms, with a scanning angular field of view of 100 °. It is positioned horizontally 40 cm over the ground surface. The whole algorithm runs at video frame rate on a dual Intel Xeon™ 1.8 GHz personal computer.

7.2 Results

The main objective is to obtain a correct detection rate and almost no false detections. Several aspects must be highlighted: the global performances (rates of non detections and false detections), the robustness of the criteria with respect to errors in the local disparity map, and the ability to work with various types of obstacles.

False detections: To assess the false detection rate, the test vehicle has been driven on a very bumpy and dent parking area to obtain a large number of false detections due to the

intersection of the laser plane with the ground surface. The results are reported in Table 3 (7032 images have been processed).
False detections are globally correctly invalidated using the obstacle-pixels and prevailing alignment criteria. The laser points altitude criterion provides more false

	Laser scanner	Number of obstacle-pixels	Prevailing alignment orientation	Laser points altitude
False Detections	781	3	10	167

Table 3. Number of false detections with the different criteria.

detections than expected, because of its high sensibility to the calibration errors between stereovision and laser scanner. Indeed, a slight error in the positioning of the scanner relative to the cameras can lead to a serious error in laser points projection, especially at long ranges. The other criteria are not dramatically affected by this issue. Most of the remaining false detection occur when the local ground surface is uniform, without any texture allowing to match pixels. So they can be removed using simple heuristics as: no obstacle can be confirmed without enough information in the region of interest. It hardly affects the detection rate, and the false detection rate of obstacle-pixels criterion almost falls to zero.
The main source of errors for the prevailing alignment algorithm comes from cases where the ground surface has non relevant texture, but where the region of interest contains a small part of a nearby object (wall, vehicle, . . .).
Detection failure: The rate of correct laser detections that have been confirmed by the different criteria has been assessed. To check, at the same time, that it can indifferently deal with various kinds of obstacles, this test has been realized with two different obstacles: a vehicle followed by the instrumented vehicle (1268 images processed), and a pedestrian crossing the road at various distances (1780 images processed). The confirmation rate of each criterion (number of obstacles detected by the laser / number of obstacles confirmed) for these two scenarios is reported in Table 4. The three criteria can successfully confirm most of the detections with both kinds of obstacles.

	Number of obstacle-pixels	Prevailing alignment orientation	Laser points altitude
Car	97.4 %	98.5 %	95.2 %
Pedestrian	91.9 %	94.9 %	97.8 %

Table 4. Rate of correct detection successfully confirmed.

Conclusion of the comparison: None of the presented obstacle confirmation criteria really outperforms the others. The obstacle-pixels is based on an intuitive approach and can deal with any types of obstacles. But it is seriously influenced by the quality of the disparity map. The more global feature of the prevailing alignment criterion makes it more robust to this kind of errors.
The laser points altitude is not sufficiently reliable to be exploited alone. Thus an efficient architecture for the application consists in using the laser points altitude to invalidate some false laser targets before the tracking step. Then the tracked obstacles are confirmed using obstacle-pixels criterion.
Performances of the perception system embedded in a collision-mitigation system: a collision mitigation system has been designed on the basis of the fusion scheme described above. This collision mitigation system can be divided into three sub-systems and a decision

unit that interconnects these sub-systems. The first sub-system is the very obstacle detection system, implementing the number of obstacle-pixels criteria for confirmation; the second sub-system is a warning area generation system that predict the path the vehicle will follow and that uses an odometer and an inertial sensor. The decision unit checks whether an obstacle is located in the warning area, and whether its Time To Collision (i.e. distance / relative speed) is under 1 second; if so, a warning message is sent to the third sub-system. The third sub-system is an automatic braking system, based on an additional brake circuit activated when a warning message is received.
The detection rate has been tested on test tracks on the basis of different driving scenarios, including cross roads and suddenly appearing obstacles. The detection rate is 98.9 %.
Then, to assess the false alarm rate, this collision mitigation system has been tested in real driving conditions, on different road types: freeway, highways, rural roads and downtown. All these tests took place on the French road network around Paris. The automatic braking system was turned off and only the warning messages were checked. In normal driving situations, an automatic system should never be launched. Each time an emergency braking would have been launched is thus considered as a false alarm. The tests have been carried out under various meteorological situations: sunny, cloudy, rainy, and under various traffic situations: low traffic to dense traffic.
403 km have been ridden on freeways. The velocity was up to 36 m / s. No false alarm was observed during these tests. Fig. 14 (a) and (b) presents some typical freeway situations under which the system has been tested. 78 km have been ridden on highways and 116 km on rural roads. The velocity was up to 25 m / s. No false alarm was observed during these tests. Fig. 14 (c) (d) presents some typical highway situations, and Fig. 14 (e) (f) some rural road situations under which the system has been tested. The downtown tests are certainly the most challenging tests since the context is the more complex. 140 km have been ridden in downtown and in urban areas. The velocity was up to 14 m/s. A false alarm was observed twice. The first one is due to a matching error during association, and the second one is due to a false target detected by stereovision on a uphill gradient portion. Fig. 15 presents some typical urban situations under which the system has been tested.
For the 737 km ridden, two false alarms were observed. The false alarm rate is thus 2.7 false alarms for 1000 km. No false alarm was observed either on freeways or on highways and rural roads. The two remaining false alarms were observed in downtown. Thus, the false alarm rate in downtown is thus 1.4 false alarm for 100 km. These results are quite promising, even if the false alarm rate must be reduced by a factor of about 1000 before the system can be envisaged to be put in the hands of common driver.

Fig. 14. Typical images of freeway and rural road situations. (a) truck following on a freeway, dense traffic - (b) freeway with low traffic - (c)(d) peri-urban highway - (e)(f) rural road with tight uphill gradient.

8. Trends and Future Research

Experiments of the proposed system give the feeling that an accurate and totally robust and reliable obstacle detection system can be designed on the basis of the techniques described in this chapter. Some tuning of the different modules are required to still improve the performances: for instance, combination of various confirmation criteria should allow to avoid any false alarm. Yet, some issues still need to be tackled, such as the auto-calibration of the set of sensors. Moreover, laser scanner remain a quite.

Fig. 15. Typical images of urban situations. (a) pedestrian crossing - (b) road works - (c) car driving out of parking lot - (d) car and bus traffic - (e) narrow road and tight curve - (f) tight curve, non flat road - (g) dense traffic - (h) road with high roll - (i) narrow paved road, tight curve.

Expensive device. Designing a medium range cheap obstacle detection system featuring high performances is still a challenge for the next years but should be possible. The key could be to use only the stereovision sensor and to implement various competitive stereovision algorithms designed to confirm each other. On a global view, a first algorithm could generate a set of targets that would be tracked along time and confirmed by the other algorithms. The confirmation criteria presented above could be used for this purpose. To reach acceptable accuracy, sub-pixel analysis should be used. Auto-calibration techniques are also required, above all for long baseline stereo sensors. Since stereovision algorithms require massive computations, real-time performance could be achieved only at the cost of a dedicated powerful chipset. Once designed, such a chipset should be not expansive to produce. Thus, a breakthrough in the field of robotics is foreseeable and would result in many applications that can not be considered nowadays because of the dissuasive cost of state-of-the-art obstacle detection systems.

9. References

Ahuja R. K., Magnanti T. L. &, Orlin J. B. (1993), *Network Flows, theory, algorithms, and applications* , Editions Prentice-Hall.

Bertozzi M. & Broggi A. (1998). *GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection*, IEEE Transaction on image processing, Vol. 7, N°1.

Blackman S. and Popoli R. (1999). *Modern Tracking Systems*, Artech.

Franke U. & Joos A. (2000). *Real-time stereovision for urban traffic scene understanding*, IEEE Intelligent Vehicle Symposium, Dearborn, USA.

Gavrila M., Kunert M.& Lages U. (2001), *A Multi-Sensor Approach for the Protection of Vulnerable Traffic Participants: the PROTECTOR Project,* IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary.

Kuhn H. W. (1955), *The Hungarian method for assignment problem*, Nav. Res. Quart., 2.

Koller D., Luong T. & Malik J. (1994). *Binocular stereopsis and lane marker flow for vehicle navigation: lateral and longitudinal control*, Technical Report UCB/CSD 94-804, University of California at Berkeley, Computer Science Division.

Labayrade R. & Aubert D. (2003 a), *A Single Framework for Vehicle Roll, Pitch, Yaw Estimation and Obstacles Detection by Stereovision*, IEEE Intelligent Vehicles Symposium 2003, Columbus, USA.

Labayrade R. & Aubert D. (2003 b). *In-Vehicle Obstacle Detection and Characterization by Stereovision*, IEEE In-Vehicle Cognitive Computer Vision Systems, Graz, Austria.

Mendes A., Conde Bento L. & Nunes U. (2004), *Multi-target Detection and Tracking with a Laserscanner, IEEE Intelligent Vehicles Symposium 2004*, pp 796-801, Parma, Italy.

Mobus R. & Kolbe U. (2004), *Multi-Target Multi-Object Tracking, Sensor Fusion of Radar and Infrared,* IEEE Intelligent Vehicles Symposium 2004, pp 732-737, Parma, Italy.

Rombaut M. (1998). *Decision in Multi-obstacle Matching Process using Theory of Belief,* AVCS'98, Amiens, France.

Scharstein D., Szeliski R. & Zabih R. (2001). *A taxonomy and evaluation of dense two-frame stereo correspondence algorithm*s, IEEE Workshop on Stereo and Multi-Baseline Vision, Kauai, HI.

Shafer G. (1976). *A mathematical theory of evidence,* Princeton University Press.

Steux B. (2002), *Fade: A Vehicle Detection and Tracking System Featuring Monocular Color Vision and Radar Data Fusion,* IEEE Intelligent Vehicles Symposium 2002, Versailles, France.

Williamson T-A. (1998). *A high-performance stereo vision system for obstacle detection*, PhD, Carnegie Mellon University.

6

Optical Three-axis Tactile Sensor

Ohka, M.
Graduate School of Information Science, Nagoya University
Japan

1. Introduction

The three-axis tactile sensor has attracted the greatest anticipation for improving manipulation because a robot must detect the distribution not only of normal force but also of tangential force applied to its finger surfaces (Ohka, M. et al., 1994). Material and stability recognition capabilities are advantages of a robotic hand equipped with the three-axis tactile sensor (Takeuchi, S. et al., 1994). In peg-in-hole, a robot can compensate for its lack of degrees of freedom by optimum grasping force, allowing an object to move between two fingers using measured shearing force occurring on the finger surfaces (Brovac, B. at al., 1996). Also, a micro-robot would be required to remove any object attached to the inside a blood-vessel or pipe wall (Guo, S. et al., 1996; Mineta, T. et al., 2001; Yoshida, K. et al., 2002). It therefore becomes necessary to measure not only the normal force but also the shearing force.

Principle of the three-axis tactile sensor is described in this chapter. The authors have produced three kinds of three-axis tactile sensor: one columnar and four conical feelers type, none columnar feeler type for micro robots and a hemispherical type for humanoid robotic hands. Finally, a tactile information processing is presented to apply it to robotic object-recognition. The information processing method is based on a mathematical model formulated according to human tactile sensation.

2. Principle of Three-axis Tactile Sensor

2.1 Optical tactile sensor

Tactile sensors have been developed using measurements of strain produced in sensing materials that are detected using physical quantities such as electric resistance and capacity, magnetic intensity, voltage and light intensity (Nicholls, H. R., 1992). The optical tactile sensor shown in Fig. 1, which is one of these sensors, comprises an optical waveguide plate, which is made of transparent acrylic and is illuminated along its edge by a light source (Mott, D. H. et al., 1984; Tanie, K. et al., 1986; Nicholls, H. R., 1990; Maekawa, H. et al., 1992). The light directed into the plate remains within it due to the total internal reflection generated, since the plate is surrounded by air having a lower refractive index than the plate. A rubber sheet featuring an array of conical feelers is placed on the plate to keep the array surface in contact with the plate. If an object contacts the back of the rubber sheet, resulting in contact pressure, the feelers collapse, and at the points where these feelers collapse, light is diffusely reflected out of the reverse surface of the plate because the rubber has a higher refractive index than the plate. The distribution of contact pressure is calculated

from the bright areas viewed from the reverse surface of the plate.
The sensitivity of the optical tactile sensor can be adjusted by texture morphology and hardness of the sheet. The texture can be easily made fine with a mold suited for micro-machining because the texture is controlled by adjusting the process of pouring the rubber into the mold. This process enables the production of a micro-tactile sensor with high density and sensitivity by using the abovementioned principle of the optical tactile sensor. However, this method can detect only distributed pressure applied vertically to the sensing surface and needs a new idea to sense the shearing force. In this chapter, the original optical tactile sensor is called a uni-axial optical tactile sensor.
If we produce molds with complex structures to make rubber sheets comprising two types of feeler arrays attached to opposite sides of the rubber sheet, it will be possible to improve the uni-axial tactile sensor for use in three-axis tactile sensors (Ohka, M. et al., 1995, 1996, 2004). One of these types is a sparse array of columnar feelers that make contact with the object to be recognized; the other is a dense array of conical feelers that maintain contact with the waveguide plate. Because each columnar feeler is arranged on several conical feelers so that it presses against conical feelers under the action of an applied force, three components of the force vector are identified by distribution of the conical feelers' contact-areas.
Besides of the abovementioned three-axis tactile sensor comprised of two kinds of feelers, there is another design for ease of miniaturization. In the three-axis tactile sensor, the optical uni-axial tactile sensor is adopted as the sensor hardware and three-axis force is determined by image data processing of conical feeler's contact-areas to detect three-axis force (Ohka, M. et al., 1999, 2005a). In the algorithm, an array of conical feelers is adopted as the texture of the rubber sheet. If combined normal and shearing forces are applied to the sensing surface, the conical feelers make contact with the acrylic board and are subjected to compressive and shearing deformation. The gray-scale value of the image of contact area is distributed as a bell shape, and since it is proportional to pressure caused on the contact area, it is integrated over the contact area to calculate the normal force. Lateral strain in the rubber sheet is caused by the horizontal component of the applied force and it makes the contact area with the conical feelers move horizontally. The horizontal displacement of the contact area is proportional to the horizontal component of the applied force, and is calculated as a centroid of the gray-scale value. Since the horizontal movement of the centroid has two degrees of freedom, both horizontal movement and contact area are used to detect the three components of the applied force.

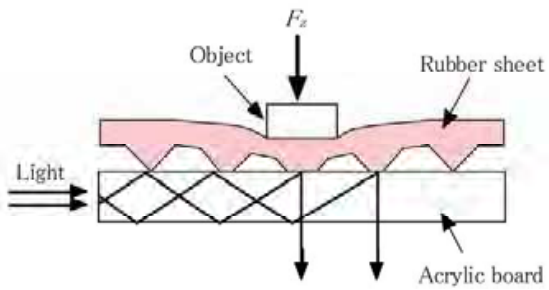


Fig. 1. Principle of an optical uni-axis tactile sensor.

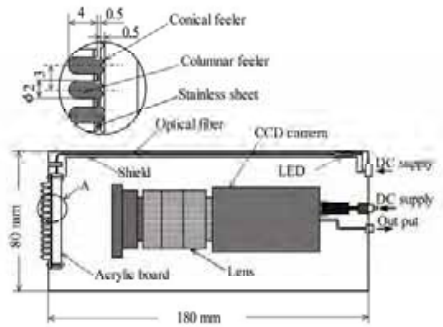
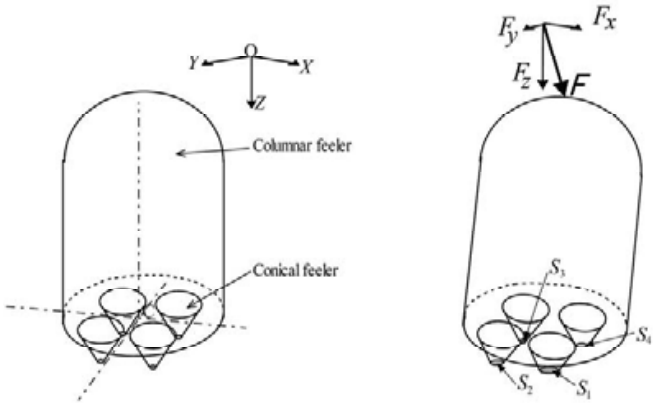


Fig.2. One columnar and four conical feeler type three-axis tactile sensor.



a) Initial, no force applied b) After force has been applied

Fig. 3.Three-axis force detection mechanism.

2.2 One Columnar and Four Conical Feelers Type

The schematic view shown in Fig. 2 demonstrates the structure of the tactile sensor equipped with sensing elements having one columnar and four conical feelers (Ohka, M. et al., 1995, 1996, 2004). This sensor consists of a rubber sheet, an acrylic plate, a CCD camera (Cony Electronics Co., CN602) and a light source. Two arrays of columnar feelers and conical feelers are attached to the detecting surface and the reverse surface of the sensor, respectively. The conical feelers and columnar feelers are made of silicon rubber (Shin-Etsu Silicon Co., KE1404 and KE119, respectively). Their Young's moduli are 0.62 and 3.1 MPa, respectively.

The sensing element of this tactile sensor comprises one columnar feeler and four conical feelers as shown in Fig. 3(a). The conical feelers and columnar feeler are made of silicon rubber. Four conical feelers are arranged at the bottom of each columnar feeler. If F_x, F_y and F_z are applied to press against these four conical feelers, the vertices of the conical feelers

collapse as shown in Fig. 3 (b). The F_x , F_y and F_z were proportional to the x -directional area-difference, A_x the A_y -directional area-difference, A_y and the area- sum, A_z respectively. The parameters A_x, A_y and A_z are defined below.

$$A_x=S_1 - S_2 - S_3 + S_4 \tag{1}$$

$$A_y = S_1 + S_2 - S_3 - S_4 \tag{2}$$

$$A_z=S_1 + S_2 + S_3 + S_4 \tag{3}$$

Under combined force, the conical feelers are compressed by the vertical component of the applied force and each cone height shrinks. Consequently, the moment of inertia of the arm length decreases while increasing the vertical force. Therefore, the relationship between the area-difference and the horizontal force should be modified according to the area-sum:

$$\begin{aligned} F_x &= (\alpha_{h0} - \alpha_h A_z)A_x \\ F_y &= (\alpha_{h0} - \alpha_h A_z)A_y \\ F_z &= \alpha_v A_z \end{aligned} \tag{4}$$

Fig. 4 .Robot equipped with the three-axis tactile sensor.

where, F_x, F_y and F_z are components of three-axis force applied to the sensing-element's tip. α_{h0} , α_h and α_v are constants determined by calibration tests.

The three-axis tactile sensor was mounted on a manipulator with five degrees of freedom as shown in Fig. 4, and the robot rubbed a brass plate with the tactile sensor to evaluate the tactile sensor. The robotic manipulator brushed against the brass plate with step-height δ = 0.1 mm to obtain the experimental results shown in Fig. 5. Figures 5(a), (b) and (c) show variations in Fz , Fx and the friction coefficient, μ , respectively. The abscissa of each figure is the horizontal displacement of the robotic manipulator. As shown in these figures, Fz and Fx jump at the step-height position. Although these parameters are convenient for presenting the step-height, the variation in Fz is better than that in Fx because it does not has a concave portion, which does not exist on the brass surface. Therefore Fz is adopted as the parameter to represent step-height.

It is noted that variation in the friction coefficient, μ , is almost flat while the robot was rubbing the tactile sensor on the brass plate at the step-height. This indicates that the tactile sensor can detect the distribution of the coefficient of friction because that coefficient should be uniform over the entire surface.

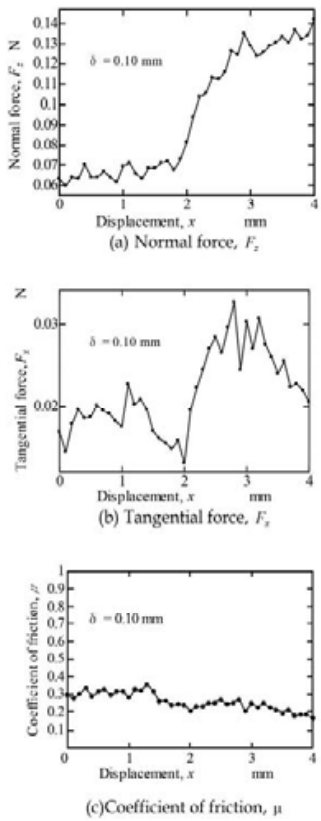


Fig. 5 Experimental results obtained from surface scanning.

2.3 None Columnar Feeler Type Three-axis Tactile Sensor for Micro Robots

In order to miniaturize the three-axis tactile sensor, the optical uni-axial tactile sensor is adopted as the sensor hardware because of simplicity and three-axis force is determined by image data processing of conical feeler's contact-areas to detect three-axis force (Ohka, M. et al., 1999, 2005a). The three-axis force detection principle of this sensor is shown in Fig. 6. To provide a definition for the force direction, a Cartesian coordinate frame is added to the figure. If the base of the conical feeler accepts three-axis force, it contacts the acrylic board, which accepts both compressive and shearing deformation. Because the light scatters on the contact area, the gray-scale value of the contact image acquired by the CCD camera distributes as a bell shape, in which the gray-scale intensity is highest at the centroid and decreases with increasing distance from the centroid.

It is found that the gray-scale $g(x, y)$ of the contact image is proportional to the contact pressure $p(x, y)$ caused by the contact between the conical feeler and the acrylic board, That is,

$$P(x, y) = Cg(x, y), \tag{5}$$

where C and $g(x, y)$ are the conversion factor and the gray-scale distribution, respectively.
If S is designated as the contact area of the acrylic board and the conical feeler, the vertical force, F_z is obtained by integrating the pressure over the contact area as follows:

$$F_z = \int_S p(x, y)dS . \tag{6}$$

If Eq. (5) is substituted for Eq. (6),

$$F_z = \int_S Cg(x, y)dS = CG , \tag{7}$$

where the integration of $g(x, y)$ over the contact area is denoted as G.
Next, to formulate horizontal components of the force vector F_x and F_y, x- and y- coordinates of the centroid of gray-scale value, (X_G, Y_G) are calculated by

$$X_G = \frac{\int_S g(x, y)xdS}{\int_S g(x, y)dS} , \tag{8}$$

and

$$Y_G = \frac{\int_S g(x, y)ydS}{\int_S g(x, y)dS} . \tag{9}$$

In the integrations, the integration area S can be enlarged as long as it does not invade adjacent contact areas, because $g(x, y)$ occupies almost no space outside contact area. Since the shearing force induces axial strain in the silicon rubber sheet, the contact area of the conical feeler moves in the horizontal direction. The x- and y-components of the movement are denoted as u_x and u_y, respectively. They are variations in the abovementioned X_G and Y_G:

$$u_x = X_G^{(t)} - X_G^{(0)}, \tag{10}$$

$$u_y = Y_G^{(t)} - Y_G^{(0)}, \tag{11}$$

where the superscripts (t) and (0)represent current and initial steps, respectively.
If friction between the silicon rubber and the acrylic board is ignored, x- and y-directional forces, F_x and F_y are calculated as follows:

$$F_x = K_x u_x , \tag{12}$$

$$F_y = K_y u_y , \tag{13}$$

where K_x and K_y are x- and y-directional spring constants of the rubber sheet, respectively.
Here we examine the relationship between the gray-scale value of the contact image and contact pressure on the contact area to validate the sensing principle for normal force. In the investigation FEM software (ABAQUS/Standard, Hibbitt, Karlsson & Sorensen, Inc.) was used and contact analysis between the conical feeler and the acrylic board was performed. Figure 7(a) shows a mesh model of the conical feeler generated on the basis of the obtained morphologic data; actually, the conical feeler does not have a perfect conical shape, as shown in Fig. 7(a). The radius and height of the conical feeler are 150 and 100 μ m, respectively.

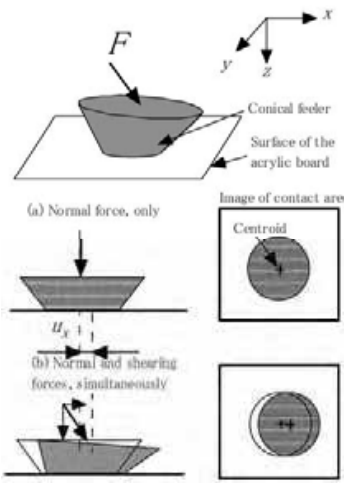


Fig. 6. Principle of a none columnar type three-axis tactile sensor.

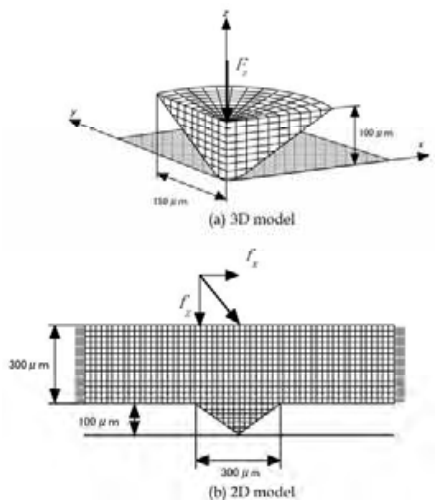


Fig. 7. Models for FEM analysis.

The Young's modulus of the silicon rubber sheet was presumed to be 0.476 Mpa. The Poisson's ratio was assumed to be 0.499 because incompressibility of rubber, which is assumed in mechanical analysis for rubber, holds for the value of Poisson's ratio. Only one quarter of the conical feeler was analyzed because the conical feeler is assumed to be symmetric with respect to the *z*-axis. Normal displacements on cutting planes of *x*-*z* and *y*-*z* were constrained to satisfy the symmetrical deformation, and the acrylic board was

modeled as a rigid element with full constraint. The three-dimensional (3-D) model was used for a precise simulation in which a normal force was applied to the top surface of the conical feeler. In the previous explanation about the principle of shearing force detection, we derived Eqs. (12) and (13) while ignoring the friction between the conical feeler and the acrylic board. In this section, we analyze the conical feeler's movement while taking into account the friction to modify Eqs. (12) and (13). Figure 7(b) shows a 2-D model with which we examine the deformation mechanism and the conical feeler movement under the combined loading of normal and shearing forces.

In the 2-D model, the same height and radius values for the conical feeler are adopted as those of the previous 3-D model. The thickness of the rubber sheet is 300 μ m and both sides of the rubber sheet are constrained. Young's modulus and Poisson's ratio are also adopted at the same values as those of the previous 3-D model. The acrylic board was modeled as a rigid element with full constraint as well. The coefficient of friction between the conical feeler and the acrylic board is assumed to be 1.0 because this is a common value for the coefficient of friction between rubber and metal. The critical shearing force, τ_{max} , which means the limitation value for no slippage occurring, is presumed to be 0.098 Mpa.

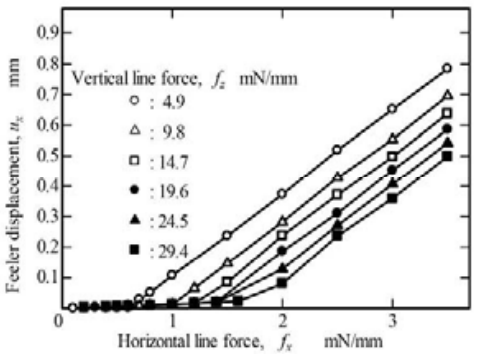


Fig. 8. Relationship between horizontal feeler movement and horizontal line force.

Combined loadings of normal and shearing forces were applied to the upper surface of the rubber sheet. The conical feeler's movement, u_x, was calculated with Eq. (10) while maintaining the vertical component of line force f_z, a constant value, and increasing the horizontal component of line force f_x, where the components of line forces f_y and f_z are x- and z-directional force components per depth length, respectively. Since the conical feeler's movement is calculated as movement of the gray-scale's centroid in the later experiments, in this section it is calculated as the movement of the distributed pressure's centroid.

Figure 8 shows the relationships that exist between the movement of the centroid of the distributed pressure, u_x, and the horizontal component of the line force, f_x. As shown in that figure, there are bi-linear relationships where the inclination is small in the range of the low-horizontal line force and becomes large in the range of the high-horizontal line force, exceeding a threshold. This threshold depends on the vertical line force and increases with increasing vertical line force, because the bi-linear relationship moves to the right with an increase in the vertical line force.

The abovementioned bi-linear relationship can be explained with the Coulomb friction law

and elastic deformation of the conical feeler accepting both normal and shearing forces. That is, the conical feeler accepts shearing deformation while contacting the acrylic board when shearing stress arising between the acrylic board and conical feeler does not exceed a resolved shearing stress. At this stage of deformation, since the contact area changes from a circular to a pear shape, the centroid of distributed pressure moves in accordance with this change in contact shape. The inclination of the relationship between u_x and f_x is small in the range of a low loading level due to the tiny displacement occurring in the abovementioned deformation stage. In the subsequent stage, when the shearing stress exceeds the resolved shearing stress τ_{max}, then according to the increase of the lateral force, the friction state switches over from static to dynamic and the conical feeler moves markedly due to slippage occurring between the conical feeler and the acrylic board. The inclination of u_x -f_x, therefore, increases more in the range of a high shearing force level than in the range of a low shearing force.

Taking into account the abovementioned deformation mechanism, we attempt to modify Eqs. (12) and (13). First, we express the displacement of centroid movement at the beginning of slippage as u_{x1}. If $u_x = u_{x1}$is adopted as the threshold, the relationship between u_x and F_x is expressed as the following two linear lines:

$$F_x = \beta_x u_x \quad (u_x < u_{x1}), \tag{14}$$

$$F_x = K_x(u_x - u_{x1}) + \beta_x u_{x1} \quad (u_x \geq u_{x1}), \tag{15}$$

where β_x is the tangential directional spring constant of the conical feeler.

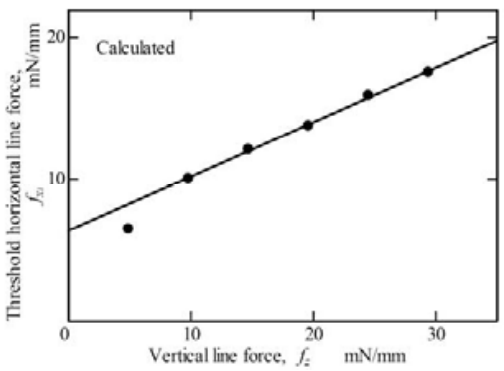


Fig. 9. Relationship between threshold of horizontal line force and vertical line force.

Second, the relationship between the horizontal line force at bending point f_{x1} and the vertical line force, f_z, is shown in Fig. 9. As is evident from this figure, f_{x1} versus f_z is almost linear in the region covering f_x =10 mN/mm. In the present paper, we assume the obtained relationship approximates a solid linear line in Fig. 9. If we denote horizontal force corresponding to u_{x1} as F_{x1}, F_{x1} is expressed as following equation:

$$F_{x1} = \alpha_x F_z + \gamma_x, \tag{16}$$

where a_x and γ_x are constants identified from F_z versus F_{x1}.

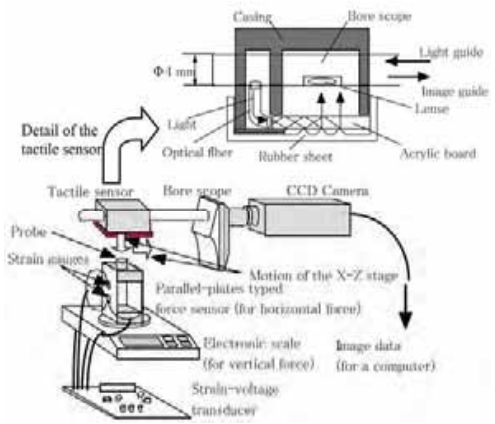


Fig. 10. A micro three-axis tactile sensor system.

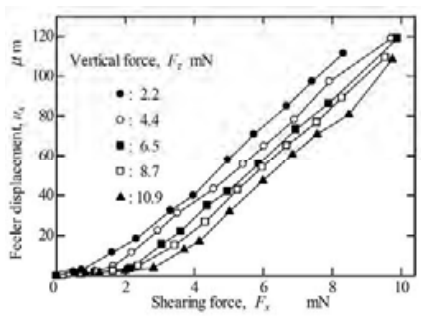


Fig. 11. Relationship between horizontal displacement of the conical feeler and shearing force.

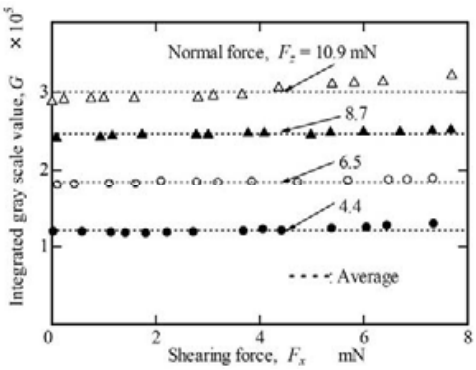


Fig. 12. Variation in integrated gray scale value under applying shearing force.

An experimental tactile sensing system is shown in Fig. 10. Light emitted from a light source in the bore scope is introduced into one side of the acrylic board, functioning as an optical waveguide. Distributed light spots caused by contact with conical feelers on the rubber sheet and the acrylic board are detected by the CCD camera through the bore scope and are accumulated into the frame memory board built into the computer. The normal force applied to the sensing surface of the tactile sensor is measured by an electric scale (resolution: 0.1 mN) and is sent to the computer through an RS232C interface. The shearing force is measured by a load cell created through our own work. The load cell consists of a pair of parallel flat springs with four strain gauges plastered to their surfaces and was calibrated with the electric scale. Two-dimensional force is applied to the sensing surface of the tactile sensor with the adjustment of a precision feed screw of the X-Z stage.

In order to evaluate Eqs. (14) to (16), after applying the initial normal force onto the sensing surface, F_x was increased in a stepwise manner while maintaining a constant normal force. Upon each increase in force, the centroid of gray-scale values within the aforementioned sub-window was calculated and the displacement of the centroid from the initial position was called u_x. In Fig. 11, the ordinate and abscissa represent the horizontal force, F_x, and the centroid displacement, u_x, respectively. As is evident from Fig. 11, the low- and high-load regions exhibit different sensitivity coefficients. This is a similar inclination to the simulated results discussed in Fig. 8.

Finally, we show variation in G under a stepwise increase of F_x and constant F_z in Fig. 12 to determine whether the relationship between G and F_z is not influenced by a variation in F_x. In fact, Fig. 12 indicates that G maintains a constant value even if F_x increases. Figure 13 shows a comparison between relationships of G -F_z with shearing force and without shearing force. In Fig. 13 the solid circles represent the relationship with the shearing force obtained from Fig. 12, and it is clear that both of the relationships almost coincide in Fig. 13. Since the magnitude of the shearing force has no influence on the sensitivity characteristic in the normal direction, it is possible to identify the shearing force and normal force independently.

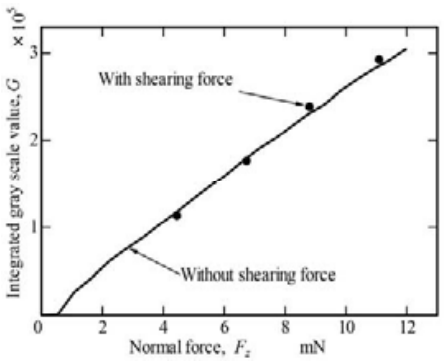


Fig. 13. Relationship between normal force and integrated gray scale value.

2.4 Hemispherical Three-axis Tactile Sensor for Robotic Fingers

On the basis of the aforementioned two examples of three-axis tactile sensors, a hemispherical tactile sensor was developed for general-purpose use. The hemispherical

tactile sensor is mounted on the fingertips of a multi-fingered hand (Ohka, M. et al., 2006). Figure 14 shows a schematic view of the present tactile processing system to explain the sensing principle. In this tactile sensor, the optical waveguide dome is used instead of the waveguide plate, which is used in the previously described tactile sensors. The light emitted from the light source is directed into the optical waveguide dome.

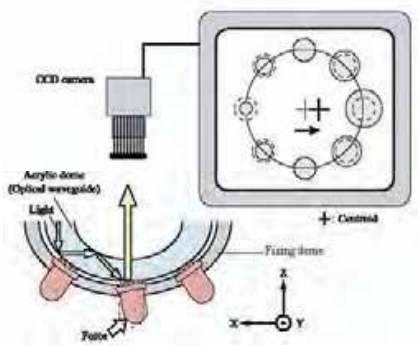


Fig. 14. Principle of the hemisherical tactile sensor system.

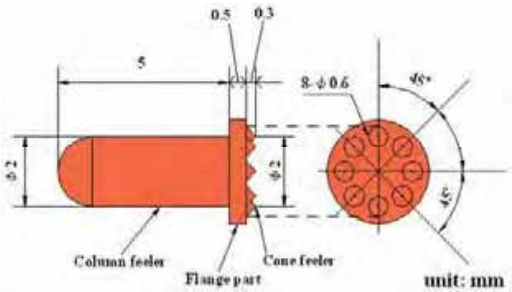


Fig. 15. Sensing element of eight feeler type.

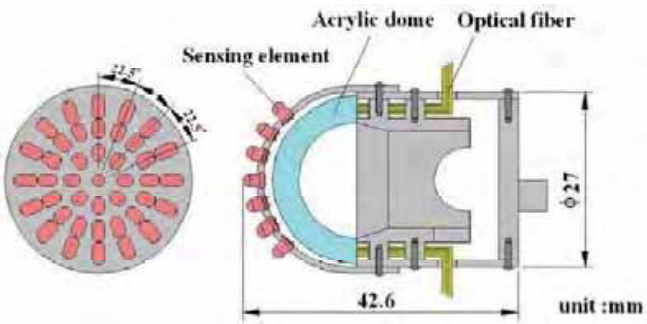


Fig. 16. Fingertip including the three-axis tactile sensor.

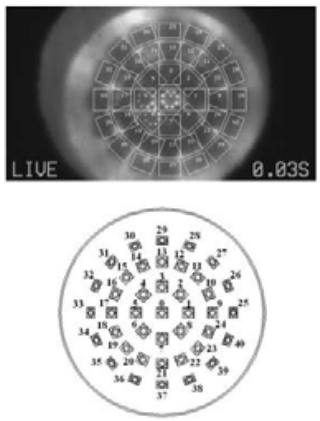


Fig. 17. Address of sensing elements

The sensing element presented in this paper comprises a columnar feeler and eight conical feelers as shown in Fig. 15. The sensing elements are made of silicone rubber, as shown in Fig. 14, and are designed to maintain contact with the conical feelers and the acrylic dome and to make the columnar feelers touch an object. Each columnar feeler features a flange to fit the flange into a counter-bore portion in the fixing dome to protect the columnar feeler from horizontal displacement caused by shearing force.

When three components of force, the vectors F_x, F_y and F_z, are applied to the tip of the columnar feeler, contact between the acrylic board and the conical feelers is measured as a distribution of gray-scale values, which are transmitted to the computer. Since the contact mechanism between the acrylic dome and conical feelers is difference from the case of flat acrylic board, relationships between the shearing force and centroid displacement and between the normal force and integrated gray scale value cannot be approximated with linear functions as shown in Eqs. (7), (12) and (13). The F_x F_y and F_z values are calculated using the integrated gray-scale value G and horizontal displacement of the centroid of the gray-scale distribution $\boldsymbol{u}=u_x\boldsymbol{i}+u_y\boldsymbol{j}$ as follows:

$$\begin{aligned} F_x &= f(u_x), \\ F_y &= f(u_y), \\ F_z &= g(G), \end{aligned} \tag{17}$$

where $\boldsymbol{i}$ and $\boldsymbol{j}$ are orthogonal base vectors of the x- and y-axes of a Cartesian coordinate, respectively; $f(x)$ and $g(x)$ are approximate none-linear curves estimated in calibration experiments.

We are currently designing a multi-fingered robotic hand for general-purpose use in robotics. The robotic hand includes links, fingertips equipped with the three-axis tactile sensor, and micro-actuators (YR-KA01-A000, Yasukawa). Each micro-actuator consists of an AC servo-motor, a harmonic drive, and an incremental encoder, and is developed particularly for application to a multi-fingered hand. Since the tactile sensors should be fitted to the multi-fingered hand, we are developing a fingertip to include a hemispherical three-axis tactile sensor. That is, the fingertip and the three-axis tactile sensor are united as shown in Fig. 16.

The acrylic dome is illuminated along its edge by optical fibers connected to a light source. Image data consisting of bright spots caused by the feelers' collapse are retrieved by an optical fiber-scope connected to the CCD camera as shown in Fig. 17.

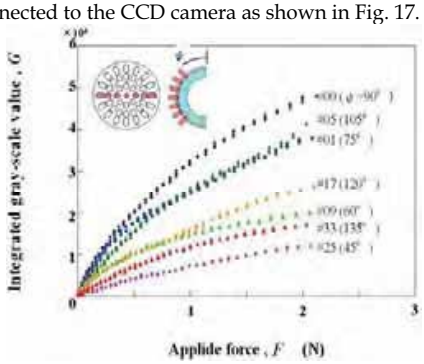


Fig. 18. Relationship between applied force and gray-scale value.

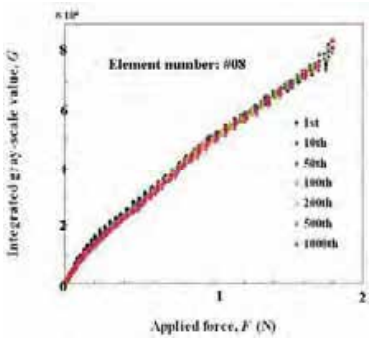


Fig. 19. Repeatability of relationship between integrated gray-scale value and applied force.

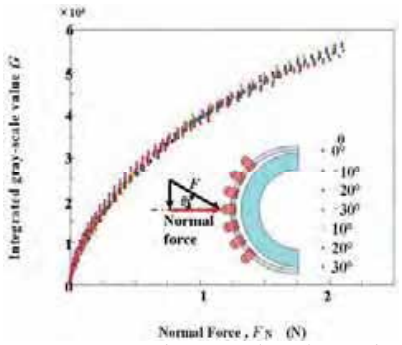


Fig. 20. Relationship between integrated gray-scale value and applied normal force at several inclinations.

To evaluate the sensing characteristics of sensing elements distributed on the hemispherical dome, we need to measure the variation within the integrated gray-scale values generated by the sensor elements. Figure 18 shows examples of variation in the integrated gray-scale value caused by increases in the normal force for sensors #00, #01, #05, #09, #17, #25 and #33. As the figure indicates, the gradient of the relationship between the integrated gray-scale value and applied force increases with an increase in φ; that is, the sensitivity depends upon the latitude on the hemisphere. Dome brightness is inhomogeneous because the edge of the dome is illuminated and light converges on the parietal region of the dome. The brightness is represented as a function of the latitude φ, and since the sensitivity is uniquely determined by the latitude, it is easy to modify the sensitivity according to φ.

The relationship between the integrated gray-scale value and applied force has high repeatability. Experimental results from 1,000 repetitions on #00 are superimposed in Fig. 19, which shows that all the curves coincide with each. The deviation among them is within 2%.

Normal force F_N and shearing force F_S applied to the sensing elements are calculated using the following formulas.

$$F_N = F \cos\theta \tag{18}$$

$$F_S = F \sin\theta \tag{19}$$

With Eq. (18) we obtained the variation in the integrated gray-scale values and applied normal force. Figure 20 displays the relationship for #00. Even if the inclination is varied from -30° to 30°, the relationship coincides within a deviation of 3.7%.

When force is applied to the tip of the sensing element located in the parietal region under several θ s, relationships between the displacement of the centroid and the shearing-force component calculated by Eq. (19) are obtained as shown in Fig. 21. Although the inclination of the applied force is varied in the range from 15° to 60°, the curves converge into a single one. Therefore, the applied shearing force is obtained independently from displacement of the centroid.

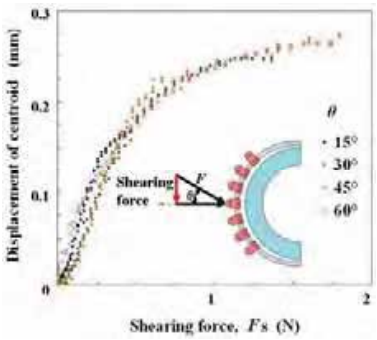


Fig. 21. Relationship between displacement of centroid and applied shearing force.

3. Human mimicking Tactile Sensing

3.1 Human Tactile Sensation

Human beings can recognize subtle roughness of surfaces by touching the surfaces with their fingers. Moreover, the surface sensing capability of human beings maintains a

relatively high precision outside the laboratory. If we can implement the mechanisms of human tactile sensation to robots, it will be possible to enhance the robustness of robotic recognition precision and also to apply the sensation to surface inspection outside the laboratory. Human tactile recognition is utilized as a model of robotic tactile regognition (Ohka, M. et al., 2005b). Human tactile recognition ability has been examined using psychophysical experiments and microneurography. Consequently, mechanoreceptors of skin are classified into four types according to response speed and receptive field size (Vallbo, Å. B. & Johansson R. S., 1984). In the present paper, we focus our discussion on FA I (First adapting type I unit) because FA I responds to surface roughness. In regard to remarks related to FA I obtained by the authors and other researchers, remarks used for the present formulation are summarized as follows:

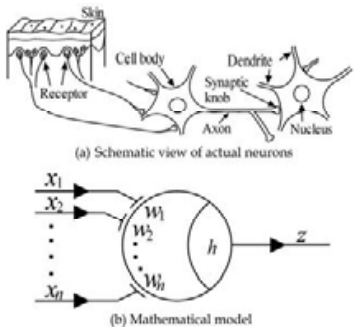


Fig. 22. Modeling of fast adaptive Type I mechanoreceptive unit.

(1) FA I responds to the first-order differential coefficient of mechanical stimulus varying with time (Moss-Salentijin, L., 1992; Miyaoka, T., 1994).
(2) Acquirable physical stimuli of FA I are surface roughness of several tens of microns in amplitude, and mechanical vibration of several microns in amplitude and several tens of Hz in frequency (Miyaoka, T., 1994).
(3) Human subjects feel moving fine step height more strongly at high scanning speeds than at low scanning speeds (Kawamura, T. et al., 1998)
(4) The mechanoreceptors related to FA I are Meissner's corpuscles(Moss-Salentijin, L., 1992; Miyaoka, T., 1994).

3.2 Neuron model

Neurophysiology studies have clarified that the mechanoreceptive units comprise a few mechanoreceptors accepting mechanical stimuli and a sensory nerve fiber transmitting sensory signals. In the present paper, a neuron processing the sensory signals is treated as an element of the unit in order to consider the unit as comprising mechanoreceptors, a sensory nerve fiber and a neuron in the brain. If we make a model of the tactile nerve system on the basis of neural network models, it is easy to incorporate the above-mentioned human tactile mechanism into robotics.

The McCulloch-Pitts model (McCulloch, W. & Pitts, W., 1943) is adopted here as the mechanoreceptive unit, while the afore-mentioned remarks on human tactile sensations are formulated to obtain expressions of the fine surface roughness recognition mechanism.

Figure 22 shows a neural network related to the tactile sensory system. When mechanical stimuli are applied to the surface of the skin, the mechanoreceptors accept the stimuli and emit a voltage signal. The signal is transmitted to a dendrite extending from a neuron through a synaptic connection. The arrival of the output signal from the mechanoreceptor effects a change in the membrane potential inside neuron. If several signals from mechanoreceptors arrive almost simultaneously at the neuron, these signals are superimposed in the neuron and summation of these signals change the membrane potential. This effect is called spatial summation and is modeled first.

The neuron accepts n-signals $x_1, x_2, \ldots, x_n$ emitted from n-mechanoreceptors distributed in the skin. The weight of the synaptic connection between i-th mechanoreceptor and the neuron is represented as w_i. Taking into account the spatial summation, the membrane potential, u is calculated as

$$u = \sum_{i=1}^{n} w_i x_i \,. \tag{20}$$

The mechanoreceptor seems to detect the time derivative of skin deformation according to Remark (1) in the previous section, where it is assumed that the mechanoreceptor detects the strain rate caused in the skin and that it emits signals proportional to the magnitude of the strain rate. Namely, the output of the i-th mechanoreceptor, x_i of Eq. (20) is calculated by the following expression,

$$x_i = a\left|\frac{d\varepsilon_i}{dt}\right|, \tag{21}$$

where ε_i is the compressive strain of the i-th mechanoreceptor and a is a coefficient.

When an output signal emitted from the mechanoreceptor arrives to the neuron, a change occurs in the membrane potential. If the next signal arrives at the neuron before the change attenuates and vanishes, the next signal is superimposed on the residual of the preceding signal. This effect is called *time summation* (Amari, T., 1978) and is formulated as convolution integral of $w_i(t - t')\,x(t')$ with respect to t' from the past to the present t if the weight of synaptic connection between the i-th mechanoreceptor and the neuron is represented as $w_i(t')$ at time t'. Consequently, by incorporating the time summation into Eq. (20), the membrane potential u is calculated as

$$u = \sum_{i=1}^{n} \int_{-\infty}^{t} w_i(t - t')x_i(t')dt' \,. \tag{22}$$

Influence of signal arrival on the membrane potential degreases with late of the signal arrival. This effect is expressed as degreasing the synaptic potential, $w_i(t)$. However, there are no available data on variation in the synaptic potential. In the present paper, it is assumed that $w_i(t)$ varies as square wave; namely it takes a constant value during 0 to τ sec, after which it takes 0.

$$w_i(t) = \begin{cases} 1, 0 \le t < \tau \\ 0, t < 0 \end{cases} \,. \tag{23}$$

It is known that neurons have the threshold effect where the neuron emits an output if the membrane potential, u expressed as Eq. (24), exceeds a threshold, h. The output is a pulse signal and the pulse density of the signal is proportional to the difference between membrane potential u and threshold h. The pulse density of the signal is expressed as z, while the threshold function,

φ (*q*) is designated to formulate the threshold effect. The pulse density, *z* is,

$$z = \phi(u - h) \tag{24}$$

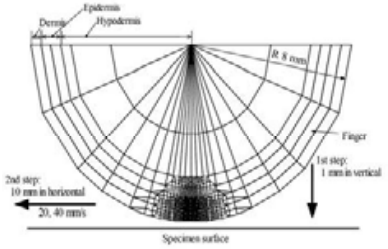
$$\phi(q) = \begin{cases} q, q \geq 0 \\ 0, q < 0 \end{cases}. \tag{25}$$

As mentioned above, data processing of the mechanoreceptive type FA I unit is formulated using a mathematical model for neuron-incorporated spatial and time summations. In the following sections, we confirm these expressions are by numerical simulation using FEM analysis of a human finger and experiments using an articulated robot installed in the present neural model.

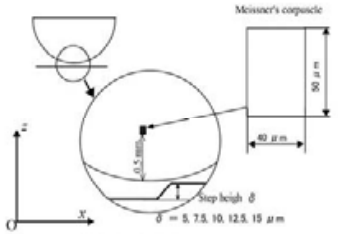
3.3 Simulation

As mentioned in Remark (4), the mechanoreceptor of FA I appears to be Meissner's corpuscle. In order to evaluate the present mathematical model derived in the preceding section, we performed a series of FEM analyses using a mesh model as shown in Fig. 23. In the present mesh model, a human finger is expressed as a half cylinder. Normal strain, ε_z arises at the existing potion of Meissner's corpuscle, calculated when the finger is slid along a flat surface having s fine step height. We selected δ=5 , 7.5 , 10 , 12.5 and 15 μ m as the step heights to compare experimental results obtained by psychophysical experiments.

It is possible that viscoelastic deformation of the skin causes the scanning speed effect described in Remark (3). In this paper, we adopt the first-order Prony series model (ABAQUS Theory manual, 1998) which is equivalent to the three-element solid, as the viscoelestic model to approximate the skin's viscoelastic behavior.



(a) Mesh model



(b) Detail of the contact portion

Fig. 23. Mesh model for contact analysis.

Human skin is composed of three layers: the epidermis, the dermis, and the hypodermis. Young's moduli of these three layers are assumed to be 0.14, 0.034 and 0.080 Mpa (Maeno, T. et al., 1998). On the other hand, the Poisson ratios of all layers are assumed to take same value of 0.45 because there are no reports concerned with it. Moreover, this value is reasonable if the skin has similar mechanical characteristics to rubber. Since there are no data on the ratio of the shearing modulus's initial value to its terminal value and the ratio between the bulk modulus' initial value and its terminal value for human skin, a common value of 0.5 for the three layers is assumed and a value of 12.9 msec (Oka, H. & Irie, T., 1993) is adopted as the time constant.

The present mesh model was compressed upon a flat rigid surface having a fine step height and slid over the surface. Then, we obtained the y-directional normal strain, ε_y in the Meissner's corpuscle, shown by a solid square in Fig. 23. The mesh element of Meissner's corpuscle is located 0.5 mm below the skin surface. The width and height of the element are 40 μ m and 50 μ m, respectively.

In the present loading history, the modeled finger was initially moved 1 mm in the negative perpendicular direction and compressed upon the flat surface. Subsequently, it was slid 10 mm in the horizontal direction. Any compressive deformation produced during the first step of the loading history should be diminished to allow evaluation of the stimulus of the fine step height caused by the scanning motion only. Therefore, after contact was established between the finger and the rigid flat surface, the finger was stabilized for 1 sec to diminish the effect of compressive deformation. Furthermore, we selected v = 20 mm/s and 40mm/s for the finger sliding speed to simplify comparison between simulated and experimental results of psychophysical experiments conducted in our previous works. We selected 0 for the coefficient of friction between the finger and the rigid surface.

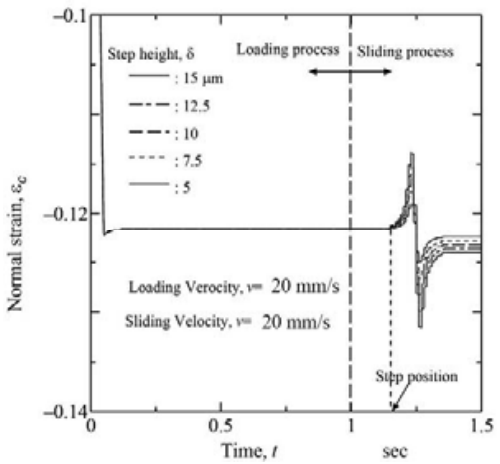


Fig. 24. Variation under compressive strain.

Next, we substituted the normal strain, ε_z obtained from the above-mentioned FEM analysis, into Eq. (21) by putting ε_z to ε_c. Subsequently, Eqs. (20)-(25) were calculated to obtain simulated signals emitted by FA I. Although the constants included in Eqs. (20)-(24), a , n , τ

and h should be determined by neurophysical experiments, we could not obtain such data. We assumed the values of these constants as follows. Here, a , the proportionality constant of relationship between output signal and stimulus magnitude, was presumed to be a = 1 Vsec. We were attempting to evaluate the simulation by normalizing outputs of the present model with the highest peak value among the outputs of different conditions. Since the plane strain condition was assumed in the present simulation, it was equivalent to a simulation of Meissner's corpuscles aligned in the depth direction of this sheet and having the same characteristics. To abbreviate the present analysis, the variance among mechanoreceptive units was ignored and n = 1 was presumed. Since the afore-mentioned dependence of speed on step height recognition seems closely related to temporal summation, we calculated several time constants within a range of τ= 10 ~300 msec. Following that, we selected the best τ that could best fit our experimental results. Since threshold h does not affect our simulated results, we summed h = 0 V.

Figure 24 shows the variation in normal strain of the position of Meissner's corpuscle as depicted in Fig. 23. Since the finger remains stational for 1 sec to erase the history of the initial compressive strain, the variation remains at an almost constant value following the transient variation occurring at the initial stage. Then, when the fine step height comes near the position of Meissner's corpuscle, two prominent spikes arise. The figure also indicates that the magnitude of the spike increases with an increase in step height.

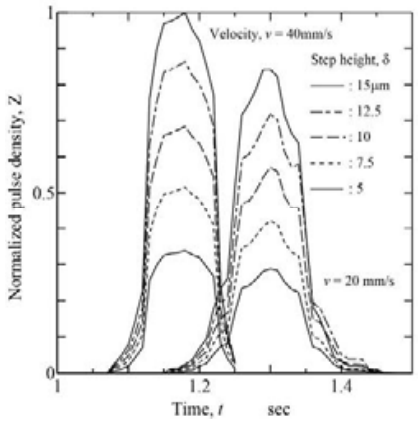


Fig. 25. Variation in normalized pulse density.

As mentioned in the previous section, we calculated several time constants within a range of τ= 10 ~300 msec. First, we will examine variation in normalized pulse density at τ= 300 msec. The strain rate calculated from the normal strain shown in Fig. 24 is substituted into the present mathematical model presented by Eqs. (20)-(25) to obtain the pulse density, z . Since we designated a= 1 as a value of the constant included in Eq. (2), the obtained pulse density z does not have any physical meaning. Hence, a comparison between calculated results under different conditions should be performed with a ratio. Here, the calculated pulse density is normalized as a peak of the calculated pulse density below v = 40 mm/s, and δ= 15 μ m is designated 1. In Fig. 25 the results are normalized according to the above-

mentioned procedure.
For both v = 20 mm/s and 40 mm/s, results show that normalized pulse density increases with the reach of the mechanoreceptor to fine step heights, and that their maximum values increase with an increase in step height, δ . In order to examine the influence of a finger's sliding speed and step height on pulse density, we obtained the maximum value for each simulation condition. Figure 26 illustrates the relationship between maximum pulse density and step height for v = 20 mm/s and 40mm/s.
The figure shows that the maximum pulse density is proportional to step height. If we compare pulse densities of different finger sliding speeds at the same step height to examine the influence of a finger's sliding speed on pulse density, we find the pulse density at a high finger speed is higher than at a low finger speed.
Next, to estimate a proper value of τ , we performed the same calculations (except for the value of τ) under the same calculation conditions as the calculation shown in Fig. 25.

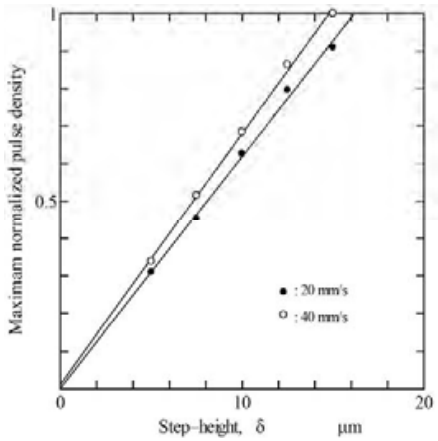


Fig. 26. Relationship between model output and step-height.

To obtain a conversion factor from the pulse density to the accepted step height, we obtained regression coefficients of calculated results for v = 20 and 40 mm/sec and adopted the mean value of the regression coefficient as the conversion factor. After employing this factor, the ordinate of Fig. 26 was transformed to an accepted step height, relationships between simulated step height and accepted step height were obtained, as shown in Fig. 27. The symbols in Fig. 27 show our experimental results (Ohka, M. Et al., 2004) obtained from a series of psychophysical experiments. This figure demonstrates that even if human subjects recognize the same step height, they feel that a given step height is higher at a high finger speed than at a low speed. Furthermore, on comparing calculated results with experimental results, we find that the calculated results coincide well with the experimental results below τ = 300 msec.

3.4 Application to robotics

The robotic manipulator shown in Fig. 4 rubbed a brass plate with the tactile sensor's sensing surface to obtain surface data of the brass plate. To enable the robotic manipulator

to traverse the brass plate correctly, it is possible to adjust the horizontal datum of the brass plate with three screws attached to it at intervals of 120˚. We prepared three brass plates having step heights of δ= 05.0 , 0.1 , 0.2 mm, and one brass plate having no step height (δ = 0 mm).

During the surface scanning test,

(1) we maintained contact between the tactile sensor's sensing surface and brass plate, and had the sensing surface press on the brass plate to apply an initial vertical force to the sensing element;
(2) the robotic manipulator traversed the brass plate in horizontal movement of 10 mm.

As a result, we obtained $\Delta U_z = U_z - U_{z0}$, which is the difference between the current vertical displacement and the initial vertical displacement, U_{z0}.

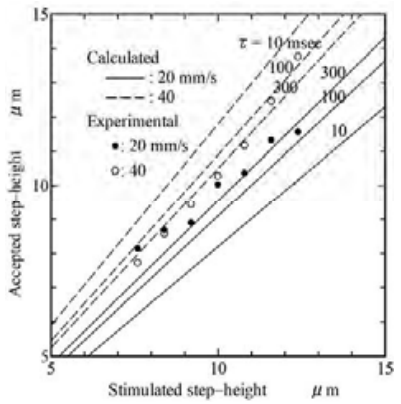


Fig. 27. Comparison of simulated results and experimental results.

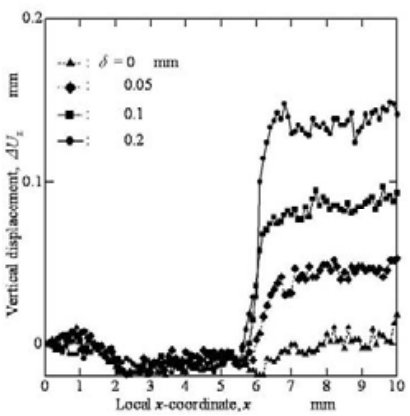


Fig. 28. Variation in vertical displacement.

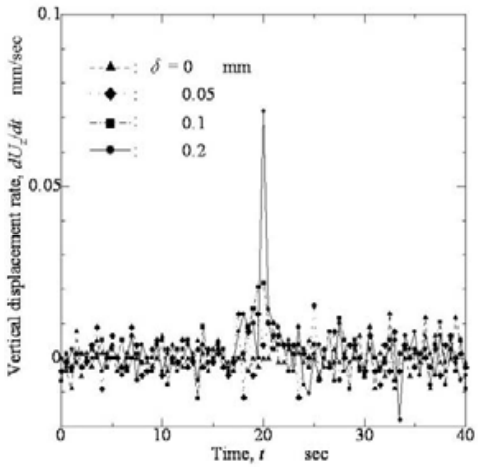


Fig. 29. Variation in rate of vertical displacement.

Figure 28 shows variation in the vertical displacement measured by the sensing element. The abscissa and ordinate of Fig. 28 are the local *x*-coordinates of the robotic manipulator's end-effecter and the difference between the current vertical displacement and initial vertical displacement, ΔU_z, respectively. The origin of the abscissa corresponds to the position where the robotic manipulator applied the initial vertical force to the tactile sensor. Figure 28 shows that ΔU_z jumps at the step-height position, and that the jump heights increase with increasing step height. Furthermore, according to Fig. 28, the experimental results vary according to the step: if the step-height magnitudes of the experimental results are examined, it is found that the ratio is about 1:2:5, a ratio that approximates the ratio of the step heights formed on the brass plates (1:2:4). Therefore, the sensor can detect step heights formed on the surface of an object. However, if we intend to measure the step height from variations in the vertical displacement ΔU_z , then the initial displacement U_{z0} should be constant because it is used as a datum for step-height measurement. In the case of robotics, the sensing surface of the tactile sensor often repeats touching and detaching from the object surface. Furthermore, there is no guarantee that the sensing surface faces parallel to the object surface; for step-height sensing, it is preferable that the step height is estimated from the current values.

As a candidate for the current physical quantity excluding the vertical displacement, U_z , we attempt to consider the time derivative of vertical displacement, $\dot{U}_z$. Figure 29 shows the variation in $\dot{U}_z$. The abscissa represents the time elapsed from begin of the scan just *z* after initial load is applied. In Fig. 29, $\dot{U}_z$ has a peak value corresponding to the position of step height. Since U_z is determined from the current value obtained from the measurement, it appears more suitable than ΔU_z for robotic real-time step-height recognition, though it

depends on the value of U_{z0}. However, since $\dot{U}_z$ contains many noise components, it is difficult to discriminate step heights of δ = 0.05 mm and 0.1 mm. Therefore, $\dot{U}_z$ holds no advantage for fine discrimination of step height.

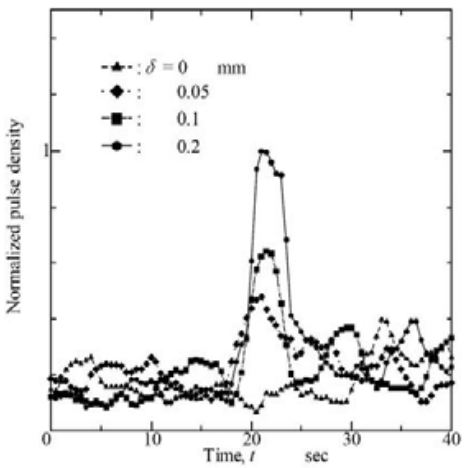


Fig. 30. Variation in pulse density

Next, the present model was incorporated into the robotic manipulator's surface-recognition system. The variation in the time derivative of vertical displacement of the present tactile sensor in Fig. 29, $\dot{U}_z$ is divided by a representative length of the tactile sensor to obtain the strain rate substituted into Eq. (21). The strain rate becomes an input signal of the present model and is used to derive pulse density, z with Eq. (24). In this calculation, we employed the following constants included in the model: a= 1 Vsec, n = 1, τ= 3 sec, h= 0 V.

In estimating the time constant, τ , we considered the difference of time consumption for data processing between human tactile sensation and robotic tactile recognition. Namely, since image-data processing is required to obtain tactile data in the present tactile sensor, sampling time is rather long at 0.5 sec. In contrast, FA I's band of tactile frequency is approximately several tens of Hz, and is one digit larger that the tactile sensor's band. Therefore, in calculating the present model, we used a value ten times larger than the τ= 300 msec in Fig. 27.

Figure 30 illustrates the output of the present model. In this figure , the ordinate shows a normalized pulse density with its maximum value at a step height of 0.2 mm, while variation in the normalized pulse density, Z shows a single peak value. Furthermore, it is easy to distinguish the difference between the cases of δ = 0.05 mm and 0.1 mm due to the noise-filtering effect of the present model. This discrimination was impossible in Fig. 29. As a result, we confirm that the present model is effective for robotic recognition of fine surface step heights in real time.

4. Conclusion

In this chapter, mechanism and evaluation of the optical three-axis tactile sensor are described to show sensing characteristics on three components of force vector applied to the sensing element. Then, recognition method for subtle convex portions of flat plate is presented to show effective of the present method using a series of experiments.
In future work, the author plans to mount the present three-axis tactile sensors on a micro and robotic fingers to perform verification experiments and will perform edge tracing of an object and object manipulation. In these experiments, the presented recognition method will be effective to determine subtle concave and convex portions located on any surfaces.

5. References

ABAQUS Theory Manual (Ver.5-7), (1998), pp.4.72.1-10.

Amari, T., *Mathematics of Neural Networks*, (1978) , Sangyo-Tosho, (in Japanese).

Borovac, B., Nagy, L., and Sabli, M., Contact Tasks Realization by sensing Contact Forces, *Theory and Practice of Robots and Manipulators* (*Proc. of 11th CISM-IFToNN Symposium)*, Springer Wien New York, (1996), pp. 381-388.

Guo, S, Fukuda, T., Kosuge, K., Arai, F., Oguro, K., and Negoro, M., Micro-Catheter Micro System with Active Guide Wire, *Proc. 1995 IEEE Int. Conf. On Robotics and Automation*, (1995), pp. 79-84.

Kawamura, T., Ohka, M., Miyaoka, T., and Mitsuya, Y., Human Tactile Sensation Capability to Discriminate Moving Fine Texture, *Proceedings of IEEE RO-MAN'98* (*IEEE International Workshop on Robot and Human Communication*), (1998), pp. 555-561.

Maekawa, H., Tanie, K., Komoriya, K., Kaneko, M., Horiguchi, C., and Sugawara, T., Development of Finger-shaped Tactile Sensor and Its Evaluation by Active Touch, *Proc. of the 1992 IEEE Int. Conf. on Robotics and Automation*, 1992, pp. 1327-1334.

Maeno, T., Kobayashi, K., and Yamazaki, N., Relationship Between the Structure of Human Finger Tissue and the Location of Tactile Receptors, *Bulletin of JSME International Journal*, Series C, Vol. 41, No. 1, (1998), pp. 94-100.

McCulloch, W. & Pitts, W., A Logical Calculus of the Ideas Imminent in Nervous Activity, *Bulletin of Mathematical Biophysics*, 7 (1943), 115-133.

Mineta, T., Mitsui, T., Watanabe, Y., Kobayashi, S., Haga, Y., and Esashi, M., Batch Fabricated Flat Meandering Shape Memory Alloy Actuator for Active Catheter, *Sensors and Actuators*, A 88, (2001), pp. 112-120.

Miyaoka, T., Module Mechanisms in Tactile Sensation, *Bulletin of Shizuoka Insititute of Science and Technology*, Vol. 3 (1994) , 15-28, (in Japanese)

Moss-Salentijin, L., The Human Tactile System, *Advanced Tactile Sensing for Robotics* (*edited by H. R. Nicholls*), (1992), 123-145.

Mott, D. H., Lee, M. H., and Nicholls, H. R., An Experimental Very High Resolution Tactile Sensor Array, *Proc. 4th Int. Conf. on Robot Vision and Sensory Control*, (1984), pp. 241-250.

Nicholls, H. R., Tactile Sensing Using an Optical Transduction Method, *Traditional and Non-traditional Robot Sensors* (*Edited by T. C. Henderson*), Springer-Verlag, (1990), pp. 83-99.

Nicholls, H. R., *Advanced Tactile Sensing for Robotics* (*H. R. Nicholls, eds.*), World Scientific Publishing, Singapore, (1992), pp. 13-47.

Oka, H. & Irie, T., Bio-mechanical properties for Soft Tissue Modeling, *Biomechanism*, Vol. 17-4 (1993), (in Japanese).

Ohka, M., Kobayashi, M., Shinokura, T., and Sagisawa, S. et al., Tactile Expert System Using a Parallel-fingered Hand Fitted with Three-axis Tactile Sensors, *JSME Int. J., Series C*, Vol. 37, No. 1, 1994, pp. 138-146.

Ohka, M., Mitsuya, Y., Takeuchi, S., Ishihara, H., and Kamekawa, O., A Three-axis Optical Tactile Sensor (FEM Contact Analyses and Sensing Experiments Using a Large-sized Tactile Sensor), *Proc. of the 1995 IEEE Int. Conf. on Robotics and Automation*, 1995, pp. 817-824.

Ohka, M., Mitsuya, Y., Hattori, K., and Higashioka, I., Data Conversion Capability of Optical Tactile Sensor Featuring an Array of Pyramidal Projections, *Proc. of 1996 IEEE/SICE/RSJ Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, (1996), pp.573-580.

Ohka, M., Higashioka, I., and Mitsuya, Y., A Micro Optical Three-axis Tactile Sensor (Validation of Sensing Principle Using Models), *Advances in Information Storage Systems*, Vol. 10, (Bhushan, B. & Ono, K., eds.),(World Scientific Publishing, Singapore, 1999), pp. 313-325.

Ohka, M., Mitsuya, Y., Matsunaga, Y., and Takeuchi, S., Sensing Characteristics of an Optical Three-axis Tactile Sensor Under Combined Loading, *Robotica*, (2004), Vol. 22, pp. 213-221.

Ohka, M., Mitsuya, Y., Higashioka, and I., Kabeshita, H., An Experimental Optical Three-axis Tactile Sensor for Micro-Robots, *Robotica*, vol.23-4, (2005), pp. 457-465.

Ohka, M., Kawamura, T., Itahashi, T., Miyaoka, T., and Mitsuya, Y., A Tactile Recognition System Mimicking Human Mechanism for Recognizing Surface Roughness, *JSME International Journa*l, Series C, Vol. 48, No.2, (2005), pp.278-285.

Ohka, M., Kobayashi, H., Takata, J., and Mitusya, Y., Sensing Precision of an Optical Three-axis Tactile Sensor for a Robotic Finger, *Proc. of 15th IEEE Int. Proceedings of IEEE RO-MAN'98* (*IEEE International Workshop on Robot and Human Communication*), (2006), (to be published).

Takeuchi, S., Ohka, M., and Mitsuya, Y., Tactile Recognition Using Fuzzy Production Rules and Fuzzy Relations for Processing Image Data from Three-dimensional Tactile Sensors Mounted on a Robot Hand, *Proc. of the Asian Control Conf.*, Vol. 3, 1994, pp. 631-634.

Tanie, K., Komoriya, K., Kaneko, M., Tachi, S., and Fujiwara, A., A High Resolution Tactile Sensor Array, *Robot Sensors Vol. 2*: *Tactile and Non-Vision, Kempston*, UK: IFS (Pubs), (1986), pp. 189-198.

Vallbo, Å. B. & Johansson, R. S., Properties of Cutaneous Mechanoreceptors in the Human Hand Related to Touch Sensation, *Human Neurobiology*, Vol. 3, (1984), pp. 3-14.

Yoshida, K, Kikuchi, M., Park, J.-H., and Yokota, S., Fabrication of Micro Electro-rheological Valves (ER Valves) by Micromachining and Experiments, *Sensors and Actuators*, A 95, (2002), pp. 227-233.

07

Vision Based Tactile Sensor Using Transparent Elastic Fingertip for Dexterous Handling

Goro Obinata, Ashish Dutta, Norinao Watanabe and Nobuhiko Moriyama
Nagoya University
Japan

1. Introduction

Humans have the ability to sense the weight and friction coefficient of the grasped object with their distributed tactile receptors. The ability makes it possible for humans to prevent from the slippage of manipulated object or collapsing the object. Such dexterous handlings are achieved by feeding back the signals from the receptors to muscle control system through neural networks. Therefore, it may be a key point for establishing dexterous handlings of robots when we try to mimic skilled human functions.

For tactile sensing of robots, several methods and sensors have been proposed by using electrical resistance, capacitance, electromagnetic component, piezoelectric component, ultrasonic component, optical component, and strain gauge (Shinoda, 2002), (Lee & Nicholls, 1999). There exist many problems of these sensors to be solved for establishing practical ones. For an example, the sensor which consists of elastic body and strain gauges requires a lot of gauges and the wiring. Moreover, the signal processing is not simple to obtain the values of the contact forces and the friction coefficients (Maeno et al, 1998). On the other hand, optical sensors have been introduced because wiring is not required in the contact part to the object (Ohka et al, 2004), (Ferrier & Brockett, 2000), (Kamiyama et al, 2003). The introduction of optical sensor makes the size small and the wiring simple. However, the sensing of friction coefficient is not considered in those papers. Piezoelectric sensors have a certain potential to solve the problems of size and wiring but there has not been a practical solution yet for measuring friction coefficient.

It is required for establishing dexterous handlings of robots to provide a purpose-built sensor for the measurement of friction coefficients between robot hand and the target surfaces. So as to avoid multiple usage of tactile sensors, we have proposed a new design of tactile sensors for multiple measuring of contact information including friction coefficient (Obinata et al, 2005).

2. Vision Based Tactile Sensor

We have proposed a vision-based sensor for multiple measuring of contact information including friction coefficient. The system consists of a CCD camera, LED lights, acrylic plate, and elastic body. The elastic body, which is made of transparent silicon rubber and has grid

pattern or dotted pattern on the spherical surface as shown in Fig.2, is to contact the object. The CCD camera is to take pictures of the spherical surface from the flat surface side of the elastic body. The experimental setup is shown in Fig.3. We can apply not only arbitrary normal and tangential forces but also moments around normal axis of contact with the sliding mechanisms.

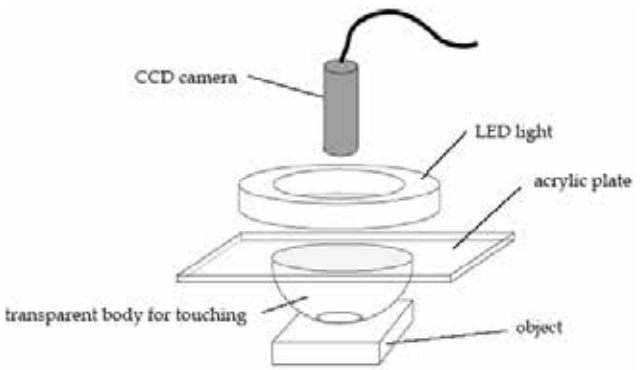


Fig. 1. Structure of vision-based tactile sensor.

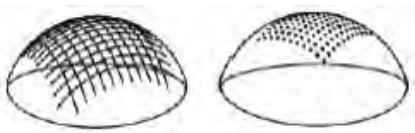


Fig. 2. Examples of shape and pattern on the surface of the elastic body.

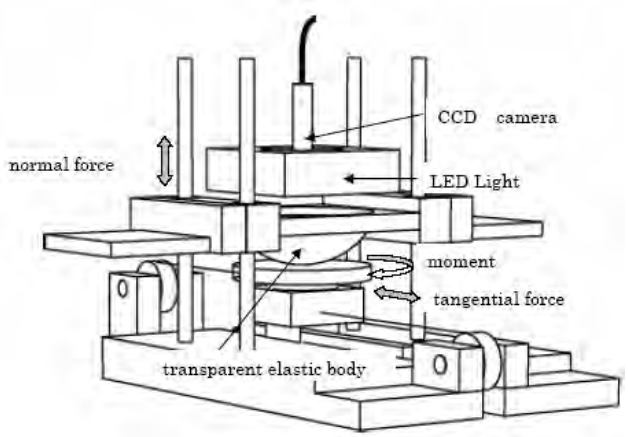


Fig. 3. Experimental setup for the sensor.

3. Measurement of Contact Force and Moment

3.1 Normal force

The relation between contact radius and normal contact force has been analyzed in cases of hemispherical surface (Xydas & Kao, 1999). The result provides the following relation:

$$a = cN^{\gamma} \tag{1}$$

where a is radius of contact area, c is a constant depending upon the material, γ is a constant depending upon the shape, N is normal force. This means that the normal force can be obtained from the contact area once the values of c and γ are determined. The picture of Fig.4 is an example when only a normal force is applied. We can estimate the contact area from the difference of brightness of each pixel; that is, we can obtain the estimation of the normal force. The dotted circle in the picture shows the estimated contact area with a certain threshold of the brightness. The experimental results are summarized as shown in Fig.5. The obtained values agree with the relation (1). The solid line in Fig.5 shows the curve with c = 4.35 and γ = 0.17. We can estimate normal contact forces based on this relation using the proposed sensor.

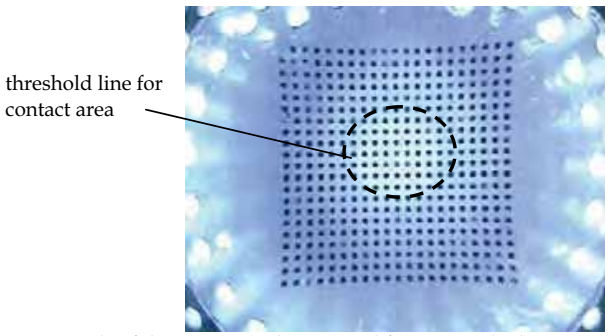


Fig. 4. Example of the picture (only a normal force is applied).

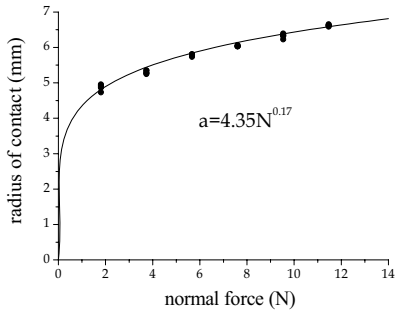


Fig. 5. Relation of radius of contact area to the applied normal force.

3.2 Tangential force

The picture in Fig.6 shows an example when normal and tangential forces are simultaneously applied. Central four dots on the surface of the elastic body were colored with red as shown in Fig.7, and are used as the origin and axes of the coordinate frame while identifying the displacements of all dots from the pre-contact positions. When the four dots are included in the contact area, the displacements of the four dots allow us to determine the direction of applied tangential force. We recognized that the displacements depend on the applied normal forces. On the other hand, the contact radius is independent on the tangential forces; thus, we estimate at first the normal force from the contact radius, and then estimate the tangential force with the estimated normal force. We found out the method for eliminating the dependency of

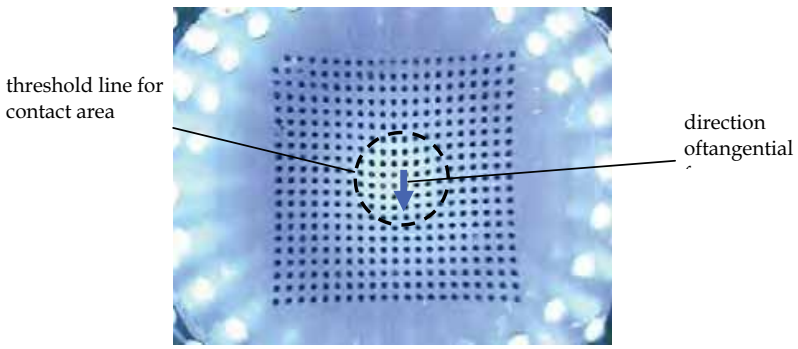


Fig. 6. Example of the picture.
(both normal and tangential forces are simultaneously applied).

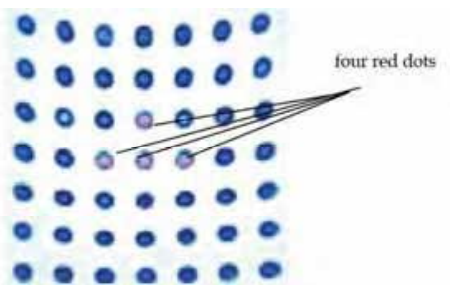


Fig. 7. Colored four dots on the surface of elastic body.

tangential force on the normal force by cut and try. Multiplying the ratio of contact radius to the measured displacements in tangential direction yields the normalized displacements. Then the relation between the normalized displacements and the tangential forces becomes one to one correspondence which is nearly expressed by one curve. The result of the normalization is summarized in Fig.8. Based on the relation, we can estimate the magnitudes of the applied tangential forces.

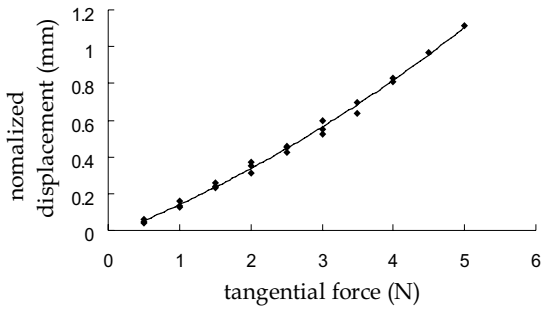


Fig. 8. Relation between normalized displacement and tangential force.

3.3 Moment

We can identify all dots on the surface of sensor in the coordinate frame defined by the colored dots. This fact allows us to obtain the vectors corresponding to the all dots which start from the positions of pre-contact phase and end at the positions of post-contact phase.

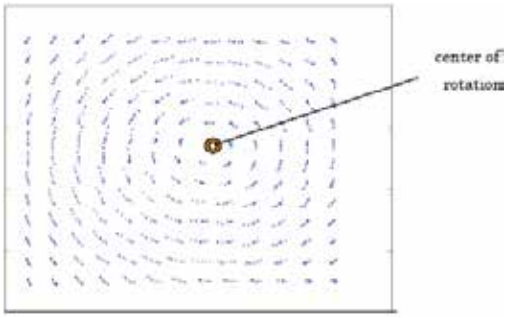


Fig. 9. Vectors for dots and center of rotation.

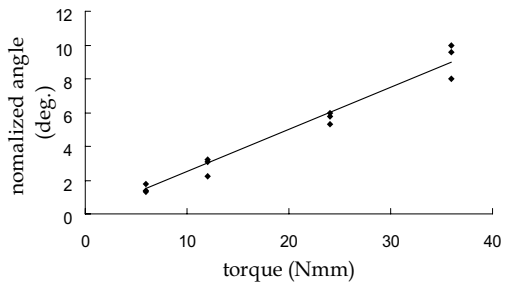


Fig.10. Relation between normalized rotation angle and applied moment.

We can estimate the center of rotation from the vectors when a moment is applied. Then, we can calculate the rotation angle from the position of the center and the vectors by using appropriate technique, for an example, least square method. The picture in Fig.9 shows how to estimate the center of rotation, for the example. We recognized that the estimated angles depend on the applied normal forces. We used the same method as the case of tangential force to eliminate the dependency; that is, the obtained angles were normalized by multiplying the ratio of contact radius. However, the result has a relatively large deviation and is summarized in Fig.10.

4. Estimation of Friction Coefficient

So as to prevent from the slippage of the manipulated object, we need to obtain the conditions of contact surface between the gripper and the object. The coefficient of static friction is important for handling the object without slipping. When the contact occurs between curved surface and flat surface, the pressure between the two surfaces distributes in the contact area. If the pressure of contact surface takes a lower value than the constant which is determined by both the surface conditions and the materials, the relative motion in tangential direction is possible in the area. This leads that the pressure distribution between the gripper and the object divides the contact area into two parts in general. In one part of contact surface, the relative motion in tangential

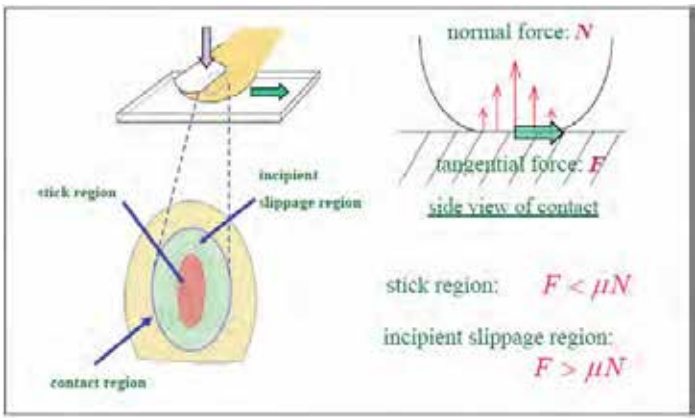


Fig.11 Schematic view of finger contact and definition of incipient slippage region.

direction is possible. We call the part as incipient slippage region. In the other part, the relative motion is impossible. This always occurs when human contact the object with fingertips. Several receptors in cutaneous sensory system of human distinguish these two areas. This means that human can sense the degree of slippage without macroscopic slippage occurring. The schematic view of finger contact and the definition of the two regions are illustrated in Fig.11. If we distinguish the two parts from the picture of CCD camera with our sensor, we can estimate the degree of slippage from the ratio of area of the incipient region to the total contact area. The ratio is defined by.

$$\rho = \frac{S_s}{S_c} = \frac{r_s^2}{r_c^2} \tag{2}$$

where S_s is stick area, S_c is contact area, r_s is radius of stick area, and r_c is radius of contact area. We call this ratio as stick ratio, and it relates to the friction coefficient. In the case of spherical and flat surfaces, the incipient slippage occurs in peripheral part of contact. So as to confirm the phenomenon experimentally with spherical and flat surfaces, we add a displacement sensor and a force sensor to the experimental setup of Fig.3. The force sensor measures directly the applied tangential force. The displacement sensor measures the movement of object. First, we identified the positions of all dots when only a normal force was applied. Next, we applied small additional force in tangential direction and increased the magnitude gradually. The dots in stick region moved in the direction, and the displacements were almost equivalent to that of the object. Note that macroscopic slippage did not occur at this moment while the surface in stick region moved with the object since the body of sensor is elastic. On the other hand, the dots in incipient slippage region moved a shorter distance in tangential direction because slippage occurred in the region. The relation between the initial distances of dots and the displacements for three cases of different tangential forces is shown in Fig.12. It should be noted that the radius of stick region decreased as the applied tangential force increased. When the radius reaches to zero, macroscopic slippage will occur. This leads to the possibility of estimating the degree of slippage from the displacements of central and peripheral dots. We propose a method for estimating stick ratio only from measurements of the sensor. The method uses the relative displacements of peripheral dots to the central dot. The radius of stick region can be determined by comparing the relative displacements with the threshold. In order to show the effectiveness of the proposed method, we carried out another experiments under different friction coefficients. We controlled the friction coefficient by using talcum powder on the contact surface and obtained the relation of the friction coefficients to the estimated stick ratios. The values of friction coefficient were determined with the ratio of the tangential force to the normal force at occurrence of the macroscopic slippage. We express the result with five lines in Fig.13. Each line corresponds to each friction coefficient. The lower stick ratio with the same displacement

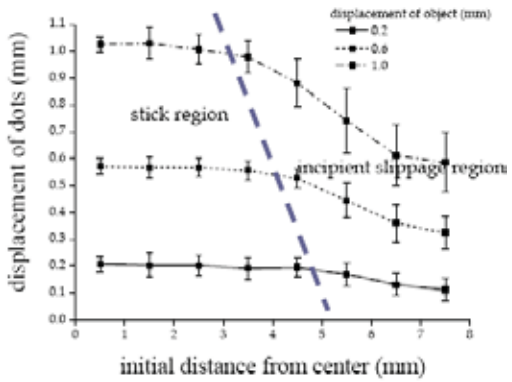


Fig. 12. Identifying incipient slippage region.

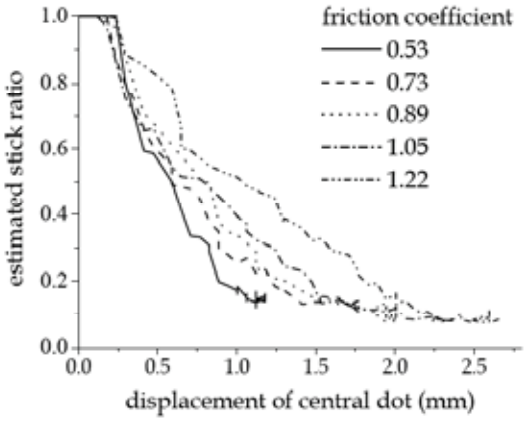


Fig.13. Estimated stick ratio in different friction coefficients.

of central dot means the lower friction coefficient. Although the proposed method cannot estimate smaller stick ratio under 0.1, we can use the estimated value to prevent from slippage because we can keep it over a certain value by feedback control of the contact force.

5. Control System for Robot Grippers Using Tactile Sensor

In this section, we describe the control system design for robot grippers to prevent from the slippage of the manipulated object. The feedback signal from the proposed tactile sensor is used to control the grip strength for stable handling of the object. The control system consists of the tactile sensor with image processing software, a voice coil motor, and a simple proportional controller with gain *K*, which is shown as the block diagram in Fig.14. The reference S_s^* is the set point for the stick area S_s. The controller amplifies the deviation $S_s^* - S_s$ by *K*, and transmit the calculated value to the voice coil motor. The voice coil motor generates the grip force under the control. The generated force is in proportion to the current in the voice coil motor. This feedback mechanism keeps the stick area around the set point. The experimental results of this control system are given by Fig.15. The tangential forces were applied manually; so, the curves in Fig.15 are not typical functions of time. The manipulated variables of control system correspond to curves in "current" of the second row which are in proportion to the generated normal forces. The current of one ampere is corresponding to the force of 3.5 N. The estimated stick ratios were normalized by the initial contact area and shows that a macroscopic slippage occurred while the control did not work. Moreover, it is shown in the figure that the control resulted in keeping the values of estimated stick ratio around the set point 0.5. This proves prevention from slippage by the control. It should be noted that the control system works only using signals from the tactile sensor.

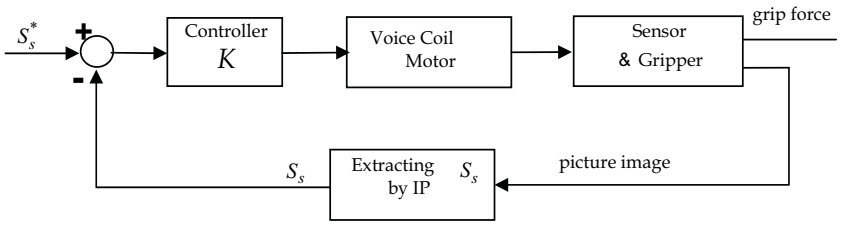


Fig.14. Block diagram of grip force control system.

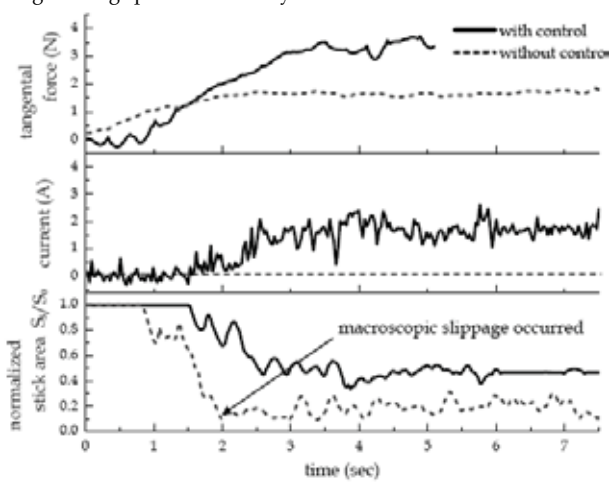


Fig. 15. Result of stick area control: Comparison between with and without control.

6. Comparison between the vision based sensor and a PVDF sensor

In this section a brief comparison is made between the proposed vision based tactile sensor and a fingertip shear force sensor made of poly-vinylidene fluoride (PVDF). Several earlier researchers have used thin piezo-electric films to sense the vibrations produced by object slip at the fingertip. However these sensors cannot measure the shear force that is essential for control of the fingertip normal force, and hence a new PVDF shear force sensor has been developed. The PVDF sensor works on the principle of directional properties of poled polymers. Poling is done with the application of a high DC voltage to orient the molecular dipoles in a particular direction. This causes the polymer to respond independently to forces in the three principle directions, as explained in (Furukawa, 1989). A small sensor was developed using poled PVDF of size 10 x 15 mm as shown in Fig. 16. The PVDF film is metallic coated on both sides on which thin wires are attached to collect the charge generated by the applied forces. The polymer film is placed in between two layers of rubber

(1 mm thick) and the whole composite assembly is placed on an aluminum 'finger' of thickness 5 mm. This prevents the polymer from bending. The three layers are bonded together by means of epoxy adhesive. Details of the polymer used are as follows:

material : PVDF, MONO
polarization class: 1-A
thickness : 25 micro-m
metallization : Ni, Cu both sides as electrode
dielectric constant : d31 ≠ d33 (where 3 refers to the poling direction)
E=3.5 Gpa

The sensor was placed on one finger (similar to Fig. 16) of a two-finger robot gripper and calibrated for different shear forces acting along the x-direction. Calibration result is as shown in Fig. 17. Several experiments were performed in which the sensor was used to grasp an object with a desired normal force and then a shear force was slowly applied to the object, to cause it to slip. Figure 18(a) shows that as the shear force increases it is recorded by the sensor. The shear force remains almost constant while slip occurs, until the object goes out of the grip. This same phenomenon is also shown in Fig. 18(b), where the shear force increase is recorded until slip occurs. Slip is recorded as a disturbance until the grip fails. From these figures it is not possible to accurately predict micro slip until macro slip takes place. Hence it is also not possible to take corrective control action to prevent macro slip. Other differences between the two sensors are, the signals obtained from PVDF sensor is noisy and requires signal processing and also measurement of a moment is not possible. The merit of the PVDF sensor appears to be its small size, weight and low cost. This comparison highlights the various advantages of the vision based tactile sensor over other types of piezo-sensors proposed so far.

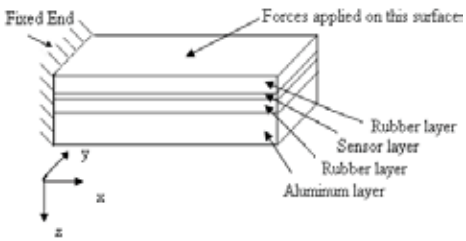


Fig. 16. Details of PVDF sensor attached to an aluminum finger.

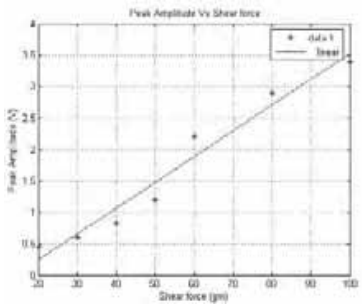
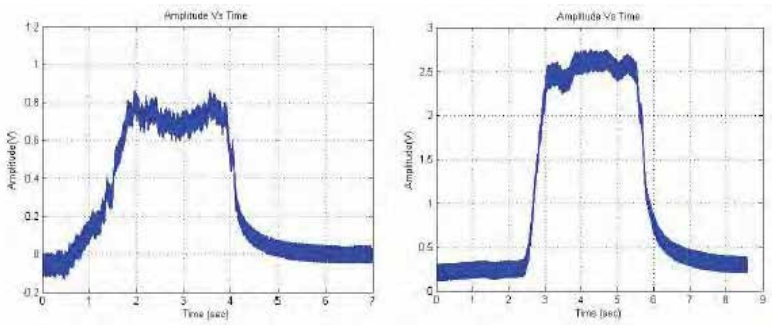


Fig. 17. Calibration of sensor output verses shear force on object.



(a) (b)

Fig. 18 (a) Shear force of 0.35 N applied to object (normal force 0.45 N),(b) shear force of 0.7 N applied to the object (normal force 0.9 N)

7. Conclusion

We have explained on a new tactile sensor for measuring multi-dimensional force and moment of contact. The structure of sensor is simple because the method is vision-based. The method to obtain the values of force and moment has been confirmed by several experiments. It is shown that the accurate estimation of contact force and moment is possible with the proposed sensor although there is a trade-off between the resolution and the computational time. There exist small interpositions between the tangential force, the moment and the normal force while measuring. Clear understanding the interposition between applied forces and moments will be required in further research. We defined the stick ratio as an index for indicating the degree of slippage. We have also proposed a new method to estimate the stick ratio for preventing from slippage of manipulated object. The exact relation of the defined stick ratio or the estimated stick ratio to the exact friction coefficient is an important problem to be solved. We demonstrated the control system for keeping the estimated stick ratio around a set point. Moreover, we have given a comparison with a piezoelectric sensor because it may be another candidate to cope with several practical requirements. The purpose-built integrated circuit for the image processing of this vision-based sensor may be required to achieve high speed control against disturbances in high frequency band. It is shown that the proposed sensor has the potential for dexterous handling like human.

8. References

Ferrier, N. J. & Brockett R. W. (2000). Reconstructing the Shape of a Deformable Membrane from Image Data, *The International Journal of Robotics Research*, Vol.19, No.9, pp.795-816, ISSN 0278-3649

Furukawa, T. (1989). Piezoelectricity and Pyroelectricity in Polymers, *IEEE Transactions on Electrical Insulation,* Vol. 24, No. 3, pp. 375-394, ISSN 0018-9367

Kamiyama, K., et. al (2003). Development of a vision-based tactile sensor, *Trans. of Society of Electrical Engineers of Japan*, Series E, Vol.123, No.1, pp16-22, ISSN 1341-8939 (in Japanese)

Lee, M. H. & Nicholls, H. R. (1999). Tactile sensing for mechatronics -a state of the art

survey-, *Mechatronics*, Vol.9, pp.1-31, ISSN 0957-4158

Maeno, T., et al (1998), Grip force control by detecting the internal strain distribution inside the elastic finger having curved surface, *Trans. of Japanese Society of Mechanical Engineers*, Vol.64, No.620, pp.1258-1265, ISSN 0025-6501 (in Japanese)

Obinata, G.; Kurashima, T. & Moriyama, N. (2005). Vision-based tactile sensor using transparent elastic fingertip for dexterous handling, *Proceeding of 36th International Symposium on Robotics,* TU41(CDROM), Tokyo Japan, Nov. 2005, International Federation of Robotics.

Ohka M.; Mitsuya, Y.; Matsunaga, Y. & Takeuchi, S. (2004). Sensing characteristics of an optical three-axis tactile sensor under combined loading, *Robotica*, Vol.22, pp. 213-221, ISSN 0263-5747

Shinoda, H. (2002). Contact sensing, a state of the art, *Trans. of Japanese Robotic Society*, Vol.20 , No.4 , pp385-388, ISSN 0289-1824 (in Japanese)

Xydas, N. & Kao, I. (1999). Contact mechanics and friction limit surfaces for soft fingers in robotics with experimental results, *International Journal of Robotics Research,* Vol.18, No. 8, p 941-950, ISSN 0278-3649

8

Accurate Color Classification and Segmentation for Mobile Robots

Raziel Alvarez, Erik Millán, Alejandro Aceves-López and Ricardo Swain-Oropeza
Tecnológico de Monterrey, Campus Estado de México
Mexico

1. Introduction

Visual perception systems are fundamental for robotic systems, as they represent an affordable interface to obtain information on different objects in the environment for a robot, and because they emulate the most commonly used sense in humans for world perception.
Many techniques can be used to identify an object within an image. Some of these techniques are color object identification, shape detection and pattern matching. Each one of these techniques has different advantages; however, color based techniques are usually preferred in real-time systems, as they require less computing power than other approaches.
Color object identification is composed by two phases: image segmentation, and object identification. The goal of the first phase is to identify all regions of the image that belong to the same object of interest. These regions are analyzed by the second phase in order to extract features of interest from these objects like geometry and relative distances and to infer the presence of a specific object.
Color image segmentation relies highly in the identification of a set of colors. Hence, color classification, which consists on identifying a pixel as a member of a color class, is essential for this process. In this chapter a technique for color image classification and its application for color segmentation will be explained in detail.
This chapter will start by presenting a set of general concepts on image processing, which will simplify the understanding of the rest of the chapter. Then, in Section 3, some of the existing approaches used for color image classification, as well as some of their advantages and drawbacks, will be described. Section 4 will describe an efficient technique for accurate color classification of images using implicit surfaces. In Section 5, it will be explained a color segmentation technique based on custom tolerance of color classification. Finally some applications will be presented in Section 6, and conclusions and future work will be discussed in Section 7.

2. Background on Image Processing

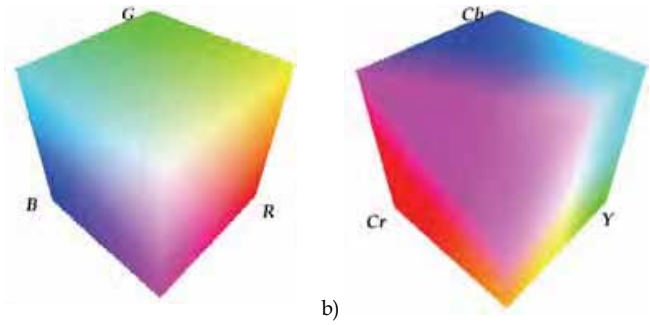
2.1 Images and pixels

An image is the graphic representation of something, usually from a 3D world, in a 2D space. That image can be defined as a function, $f(x, y)$, where x and y are spatial coordinates, and the value of f denotes the intensity in such coordinates. In particular, image processing is concerned with digital images, which contain a discrete number of x, y locations and of f values. Therefore, a

digital image is composed of a finite set of elements, each with a particular position and an associated value or set of values. Each one of these elements is referred as picture element, or *pixel*. Hence, a digital image can be considered as a two-dimensional array of pixels.
The number of values used to describe a pixel depends on how much information is used to encode the color for such elements of the image. In the case of grayscale images, a single value is used to describe the intensity of the pixel. In the case of color images, three values are usually required, f_1, f_2, f_3, indicating the intensity of different color components (i. e. primary colors) that combined will produce the color perceived by our eyes. These components will depend on the used color space.

2.2 Color spaces

A *color space*, also known as *color signal*, defines a set of attributes that uniquely identify a color. Color spaces are then important, as they set the distribution that colors present for different objects, which is fundamental in color classification and color segmentation. In addition, each color space provides with features that may be better suited for specific problems, such as varying lighting conditions or noisy environments.
Color spaces can be classified in linear and non-linear (Forsyth & Ponce, 2002). Linear color spaces are based on a set of primary colors, and describe colors in terms of a weighed sum of the intensities of these primary colors.
For instance, the RGB color space used in computer monitors describes colors in terms of three primary colors: red, green and blue. A linear combination of these components will produce all the colors that can be shown in such screens.
Another linear color space is YUV, or YCrCb. In this color space, the Y component express the brightness of the pixel, while the U and V components define their chromaticity, in terms of the amounts of blue and red, respectively. This color space is common in video cameras. Color distribution in both RGB and YCrCb color spaces is shown in Figure 1.
In contrast, non-linear color spaces can include more properties of color spaces that may help humans or different image processing techniques to better describe colors. Properties used in these spaces include tone, saturation, and intensity or brightness. *Tone* can be defined as the property that describes the way a color changes from other colors, *Saturation* describes how a color is faded by a white light, and *Intensity* or *Brightness* specifies how bright the pixel is, no matter the color. An example of a non-linear color space is HSL, which describe colors in terms of Hue (denoting tone), Saturation and Lightness (describing Intensity).



a) b)

Fig. 1. Color distribution in different color spaces. a) RGB color space. b) YCrCb color space.

2.3 Color image segmentation

Color image segmentation can be defined as the process of decomposing an image into regions with certain meaning according to the contents and the application for that specific image (Watt & Policarpo, 1998). Here, a region is a set of connected pixels sharing a set of attributes.
Classic segmentation techniques can be divided in global approaches, region-based approaches, and edge-based approaches.
Global, or threshold, approaches rely in the knowledge of pixel attributes. Thus, it is required to provide with a set of attributes that bind the classes that must be identified within an image. In the case of color images, this approach uses the color space as a 3D domain over which a set of groups or clusters will be identified for each color class. This technique is simple and efficient, but depends heavily on a good threshold definition.
Region-based approach consists in dividing an image in a set of regions that present similar properties. Techniques using this approach usually start by growing a region from an initial pixel, or *seed*, and expanding this region according to a set of homogeneity criteria. This approach presents two general problems. First, an initial seed should be picked, and this requires an additional process. And second, it is usually hard to define and parameterize the homogeneity criteria. An incorrectly defined homogeneity criterion may lead to *flooding* problems, where regions grow over the visible boundaries of the region, or to prematurely stop the growth process.
The edge-based approach uses edge detection to find a closed boundary that defines what lies inside and outside a region. The hypothesis in which this approach relies is that pixels in the boundary between two regions should be considerably different, regarding properties such as color or intensity. However, problems are produced in blurry areas of an image, where colors are not very contrasting. In addition, a problem with this approach is that failures are common when detecting closed boundaries, as borders are usually discontinuous.

3. Different approaches for color classification

A set of pixels forming a specific color image correspond to a specific set (or cloud) of points in the color space. Color classification needs to pre-define the geometry of sub-space (class), in which all contained points share the same property. Simpler the geometry of subspace is, easier the classification of pixels is but with a high risk of misclassified pixels.
There are many existing techniques to define, create and calibrate different color classes used to segment an image. Most of them try to fulfill two basic assumptions. First, resulting color classes should precisely define objects in the environment, having good generalization properties for conditions not considered in the class definition, but avoiding excessive generalization that may produce false positives. Second, color classes should be reliable enough to identify objects under variable light conditions. Definition of a color class is a complex task that usually involves adjusting a volume to a cloud of samples obtained from pixels belonging to objects of such color class. This presents a clustering problem, which may be solved either by using traditional or simplified methods. Specifically, by using simplified methods, less processing power from a mobile robot is need, allowing its use for online classification and calibration. The use of thresholds for color classification is a process that involves partitioning a color space. As presented by Castleman (Castleman, 1996), different objects on an image belong to individual groups or *clusters* of points in a histogram defined on the color space. These groups of points represent the samples from which color classes will be constructed and defined.

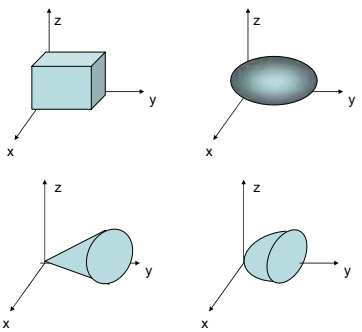


Fig. 2. Top left: Parallelepiped subspace, Top right: Ellipsoid subspace, Bottom left: Conic subspace, Bottom right: Paraboloid subspace

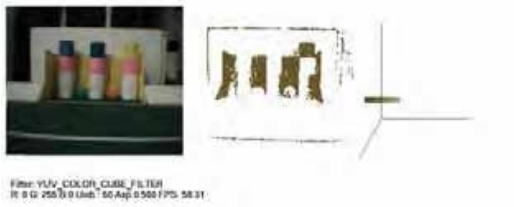
Color class construction process starts by taking color samples from objects in the environment. This sampling process consists on extracting the components of the color signal for each pixel in the objects, using images captured by the camera of the mobile robot for which the color classification is intended. These images are obtained trying to include most of the possible external conditions that the robot may find. Classic choices of subspaces geometries are shown in Figure 2.

One of the simplest techniques to define a color subspace is by setting a pair of thresholds for each component of a color signal. Knowing that linear color space is Cartesian, the subspace defined by these six thresholds (two for each coordinate) will produce a parallelepiped. The implementation of this approach produces a fast classification of pixels in the image (Bruce et al., 2000). The main advantage of this method lies on its simplicity and speed to define color subspaces, which is important if an online recalibration is expected. The most important drawback is that this volume adjusts poorly to the cloud of samples, leading to a poor classification of pixels and reduced generalization capabilities. This is due to a high risk of overlapping when many colors classes are needed, causing misclassification on pixels. A sample classification produced by this type of threshold is shown in Figure 3.a. As a result, a different subspace is need to better fit points of each color. Quadric surfaces are evaluated as a better option. These surfaces adapt more precisely to the apparent geometry of clouds of samples, for the particular color spaces used in this research. While a cloud of points in a color space has some spatial characteristics, the same has other spatial characteristic when other color space is considered. Different color spaces may benefit from other subspaces representations, as the shape of a color class may change according to the attributes analyzed for a color. In order to exemplify quadric subspaces, both RGB and YUV color spaces were used in the color image of Figure 3. Cones are a logical choice in RGB color space. If a vector is traced passing through the origin, it will intersect a set of values with the same chrominance, but with a different brightness. By using this vector to create a cone, the tolerance of the contained range will include a set of similar tones of the classified color. The radius of the cone will define the maximum threshold that the color class will allow. A sample conic threshold is shown in Figure 3.b.

However, when samples are close to the origin of the color space —when colors become very dark—, samples are more susceptible of being affected by image noise. A similar quadric

subspace that avoids the classification of samples near the origin is a paraboloid. While its shape largely resembles a cone, providing similar color classification capabilities, a variation in the base of the paraboloid denotes a criterion to discard dark samples subject to noise. The focus of the paraboloid will be calculated according to the mean value of the cloud of samples, and may be modified to include or discard dark samples. Paraboloid thresholds may be seen in Figure 3.c.

a) Parallelepiped color class



b) Cone color class

c) Paraboloid color class

d) Ellipsoid color class

Fig. 3. Use of different bounding subspaces as color classes for classification of yellow in different color spaces. From left to right: Original image, Pixels from the original image classified as yellow, 3D representation of the color subspace. a) YUV Parallelepiped color class. b) RGB Cone color class. c) RGB Paraboloid color class. d) YUV Ellipsoid color class.

Again, an issue that both of these approaches may find is their difficulties in classification of highlights or bright areas. By creating an ellipsoidal primitive, color regions are bounded more precisely, discarding highlights that may also be caused by noise. The use of ellipsoids adapt stylishly to different color space, such as RGB and YUV, as they provide a more precise approximation to the final color class. This is shown in Figure 3.d.

4. Color Image Classification through Implicit Surfaces

Though techniques based on quadric subspaces produced better results than the use of simple parallelepiped subspaces, the resulting classification still haves some areas of improvement.

First, the bounding quality of the surfaces to the cloud of samples depends highly on the distribution of the classified color in the selected color space. For instance, cones and paraboloids are more accurate describing color distribution in the RGB color space than in the YUV color space. However, ellipsoids produce better results in both color spaces.

Second, even when the color space is suited for a quadric subspace, its may not adjust as tightly as desired to a color class, which may lead to some classification problems. This is because shape of clouds of samples is affected by light incidence and reflection.

Hence, a better technique of classification was proposed in (Alvarez et al., 2004) to overcome some of these drawbacks. This new approach is based on a technique used for 3D object reconstruction (Lim et al., 1995), and the use of implicit surfaces as the threshold that bounds and defines color classes.

4.1 Implicit Surfaces

Formally, implicit surfaces are 2D geometric shapes existing in the 3D space, and defined according to a particular mathematical expression (Bloomenthal, 1997). Usually, this implies a function *f* that defines a surface. When *f* is equal to a certain threshold for a given point, this point lies on the surface, while when a point is below this threshold, it is contained by this surface.

An implicit surface is usually characterized by a skeleton and a blending function. The skeleton is a set of primitive elements—such as points and lines— that define individual implicit functions which will be the base for the final surface. The blending function defines the way in which these primitives will be combined to produce the final surface.

Implicit surfaces were selected for color classification as they easily allow evaluating when a sample lies inside or outside the volume defined by such surface, providing a natural interface for color classification. Besides, they present the property of being able to blend with other surfaces in order to produce a single one. In this way, a set of relatively sparse samples may produce a continuous surface containing all the intermediate regions to produce a single color class. Similarly, if groups of samples within a color class are very distant from each other, the surface will split, but the resulting clusters will still belong to the same class.

In this way, it is possible to adjust a surface to a cloud of samples if we can properly distribute a set of primitives, of correct dimensions, that work as the skeleton of our surface;

similarly to the work from Lim et al. (Lim et al., 1995) to reconstruct a 3D object from a set of points from its surface.

4.2 Construction Algorithm

This algorithm starts from a set of images from which a user selects a set of samples for a given color class. Then, a number of spherical primitives are uniformly distributed along the cloud of samples, to later apply the *k*-means algorithm to adjust their position. Once distributed, the radius of each primitive is estimated according to the standard deviation of samples related to each primitive. Finally, these primitives are blend to produce the final surface, or color class. The resulting color class will then be translated into a look-up table, in order to produce an efficient online color classification. Another option is to use the resulting surface function as the classifier, defining a threshold for this function as a tolerance criterion to add or exclude samples.

4.3 Generation of Primitives

Once a set of color samples has been selected, a set of primitives should be distributed among this cloud. An initial naïve approach could be the use of Delaunay tetrahedralization (Langetepe & Zachmann, 2006), which produces a set of tetrahedrons that connects every sample in the cloud. Adding a spherical primitive in the center of each tetrahedron would then produce a relatively close approximation of the sampled data. However, the resulting number of primitives would be extremely large, and discarding small tetrahedrons would produce a poor approximation of the surface.

Instead (Lim et al., 1995) propose to start the surface reconstruction by minimizing a cost function, expressed as a sum of squared terms. This cost function represents the error between the implicit surface, including the desired features of this surface, and the points from which it is constructed. Some functions may include the distance between the surface and the points, the curvature, the normal vectors of the surface, or the radius of the primitives. However, minimization of such function is neither simple nor fast, and this method is suited for samples from the surface rather than within the surface. A possible solution would include extracting the samples on the surface, but discarding samples within the surface would eliminate potentially valuable information for the process. Instead, (Alvarez et al., 2004) propose a different approach using the *k*-means algorithm.

The *k*-means algorithm, as defined by Bishop (Bishop, 1996), divide a set of *n* point samples (x_1, x_2, … x_n) in a set of *k* disjoint, non-hierarchic sets (Q_1 … Q_k), through the minimization of a distance criterion *d*. Usually, Euclidian distance metric is used, which produces spherical clusters that, in this case, suit well as primitives for an implicit function. The *k*-means algorithm is a minimization problem of a cost function:

$$J = \frac{1}{2}\sum_{j=1}^{n} d^2\left(x_j, c_{g(x_j)}\right) \tag{1}$$

where

$c_{g(x_j)}$ is the center of the cluster $g(x_j)$

$g(x_j)$ is the closer cluster to point x_j

The total cloud of samples is seen as a large cluster that will be approximated by a number of small clusters, defined by k primitives. The k-means algorithm distributes the primitives in the volume used by the cloud samples, iteratively adjusting its position to provide each cluster with a subset of the samples.
The proposed k-means algorithm is summarized as follow.

Step 1. An arbitrary set of clusters (primitives) is distributed randomly along the cloud of samples.
Step 2. Calculate the distance metrics from each sample (x_1, x_2, … x_n) to the center of each primitive (c_1, … c_k).
Step 3. Relate each sample to the closest primitive.
Step 4. Recalculate the center of primitives as the average of all of the samples related with such primitive.
Step 5. Repeat iteratively steps 2-4 until the change in the position of all centers lies below a certain threshold ε.

This process guarantees that each sample will belong to a group or primitive, and that the distribution will converge to a local minimum of (1).
Once the primitives have been distributed using the k-means algorithm, the radius for each primitive is estimated as follow. For each group, the standard deviation from the center of the primitive to all samples is calculated. The radius of the primitive is then set as a multiple of this standard deviation according to the confidence interval expected.

4.4 Construction of the color class

Once the spherical primitives are defined, centers and radius are known, they will be combined into a implicit surface, as proposed in (Lim et al., 1995).
The implicit function for each spherical primitive i is defined as:

$$f_i(P) = \frac{d^2\left(P - c_i\right)}{r_i^2} \tag{2}$$

where

P is an arbitrary point in the color space.

This function has the following properties:

$f_i(P) < \Gamma$ for points P inside the surface
$f_i(P) > \Gamma$ for points P outside the surface
$f_i(P) = \Gamma$ for points P on the surface

Here, Γ is the threshold, which may be understood as a scale parameter for the sphere. For example, if Γ is set to 1 the original scale of the spherical primitive will be considered.
Primitives are then joined into a unique implicit function, using a blending function to construct the final implicit surface. Here, a function presented by Ricci (Ricci, 1973) is used to blend the primitives:

$$f = \left\{\sum_{i=0}^{k} \frac{1}{f_i^{\rho}}\right\}^{-\frac{1}{\rho}} \text{ for } k \text{ primitives} \tag{3}$$

where

ρ is the blending degree

When ρ tends to infinity, this function converges to the absolute minimum. Graphically, this parameter controls how tight will the surface fit the primitives that compose it; when ρ tends to infinity, the function converges to the original set of primitives.
From the implicit function f, it is possible to precompute the color class and store it into a look-up table, which will permit a fast mechanism to perform online classification. However, the implicit surface function could also be used as the classifier, defining a threshold as required. Doing so will allow this approach to be more dynamic, and to adjust during the execution of online classification.
Moreover, the implicit function f can be seen as a possibility function, which provides more information about the classified sample other than its class, but also the degree of similarity between the sample and the color class, resembling a Gaussian mixture. Using other criteria —like Mahalanobis distance—would add different characteristics to this approach, which might be useful for certain application areas.
An example result for the yellow color class using this approach is shown in Figure 4.

Fig. 4. Color classification using the implicit surfaces technique. Shape on the left is the resulting implicit surface for the yellow color class.

One advantage of the implicit surfaces technique is the simplicity to modify the scale parameter Γ for spherical primitives, and the blending degree ρ, in order to modify the amount of samples identified as part of a color class. By modifying these thresholds, a set of self-contained color classes is obtained. Classes with different thresholds may be used for different segmentation stages on the same process.

5. Multilevel Classification for Image Segmentation

In general, segmentation techniques consume a lot of computer power by processing every pixel on an image, increasing the possibilities of recognizing external noise as possible objects of interest.
To avoid processing the entire image, Seed Region Growth algorithms (or *SRG*, for short) use only a set of pixels or seeds which are very likely to be part of an object of interest (von Hundelshausen & Rojas, 2003; Wasik & Saffiotti, 2002). Then, these seeds are used to grow regions of interest based on a homogeneity criterion, such as contrast between neighboring pixels. However, these techniques require a good homogeneity criterion to avoid flooding problems.
A good way to analyze a small set of pixels with high probabilities of being part of an object of interest is to use information on the camera position and orientation to only evaluate

pixels located in areas of the image where objects can be present, and discard areas where no object can be located. In particular, the use of scanlines, which are perpendicular to the horizon, discards objects above a certain height and provide different resolutions for image scanning, according to the expected distance to objects (Jüngel, 2004). However, this method alone may discard information that might be found obtaining complete regions.
The presented approach combines the advantages of scanlines and SRG algorithms by using a multilevel color classification (Alvarez et al., 2005). This classification takes advantage of the custom threshold of the implicit surfaces presented in section 4.
First, a set of scanlines will be used to extract a set of color seeds. Pixels on scanlines will be identified using a small threshold color class to reduce the number of identified pixels. Then, extracted seeds will be used by a region growth algorithm to identify regions of interest. The SRG algorithm will use a color class with a bigger threshold to better characterize entire regions.

5.1 Seed Extraction through Scanlines

According to the approach from Jüngel (Jüngel, 2004), a set of vertical scanlines will be used to locate objects of interest within an image. Density of these lines will be different according to their distance to the camera. When a line is closer to the horizon, pixels on it will probably represent objects farther from the camera. As the line moves downward away from the horizon, pixels in the scanline will probably belong to objects closer to the camera, so a smaller line density should be sufficient to locate these objects.
In order to obtain the horizon for the camera on each picture, the kinematic model of the camera should be known. This is highly dependent on the nature of the robot system and cannot be generalized, but by obtaining this kinematic model, a lot of undesired data can be discarded from the processing pipeline.
Once the horizon is known, a scan pattern is projected on the image. The scan pattern consists on a set of lines, perpendicular to the horizon, that start above the horizon and go down the interest area. Distance between scanlines depends on the projected size of objects on the screen. As the projected size of objects grows as they get closer to the camera, scanlines should become sparser as they get below the horizon. An intertwined pattern of short and long scanlines will deal with this varying scanline density. An example of this pattern is shown in Figure 5.a. Pixels found in these scanlines will be classified according to a color class with a low threshold, as seen in Figure 5.b.

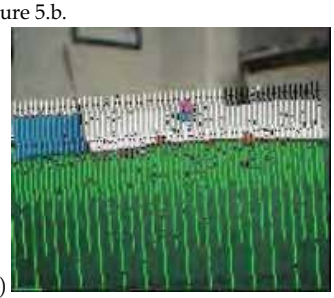
a)  b) 

Fig. 5. a) Scan pattern projected on a custom image. Scanline density is higher as pixels get closer to the horizon (dotted line). b) Seeds extracted from evaluating pixels on scanlines using a low-threshold color class.

The scanline algorithm avoids processing the entire image, reducing the number of processed pixels. In addition, this routine discards pixels above a certain height, eliminating possible sources of noise.

5.2 Seed Region Growth

Seed Region Growth is a region based technique used to perform image segmentation (Wasik & Saffiotti, 2002). A seed is selected from this initial set of color seeds. In our approach, the initial set of color seeds is obtained from the scanline seed extraction.
The selected seed is assigned to a new region, and this region is grown on a neighborhood. Usually, a 4-pixel neighborhood is selected, although an 8-pixel neighborhood can sometimes be used. When a neighboring pixel already belongs to the current region, it is ignored. If not, this pixel is evaluated through the homogeneity criterion to find if it belongs to the current region. If it does, this new pixel is added as a new seed for this region. Growth continues until there are no new seeds for the current region. Once a region is complete, a new seed is selected from the initial seed set, and new regions are created until no more initial seeds are found.
The homogeneity criterion, as mentioned before, is that the new pixel belongs to the same a high-threshold color class than the initial seed. This avoids color flooding in areas with similar contrast.

6. Applications and Results

The present technique was applied in the 4-legged league of the Robocup competition (www.robocup.org). The basic idea behind this league is to make four autonomous legged mobile robots to play soccer. This league uses a standard robot, Sony's AIBO ERS-7 robot, to guarantee a common hardware platform for each team, prohibiting the addition of additional hardware equipment for this robot. The soccer field is color tagged, providing specific colored and sized objects to allow the robot to identify the object type and location, and then to infer its position on the field.
Rules of this league state that the robot should play autonomously on the soccer field, so every decision must be taken based on an efficient evaluation of the environment based on the information from the built-in camera. The robot provides a maximum camera resolution of 208x160pixels, producing 25 frames per second. This image should be processed by an internal MIPS R7000 CPU at 576 Mhz, which should also deal with the control of all the motors of the robot and the general strategy of the soccer game. The vision algorithm should be very efficient in order to deal with as many images as possible.
An application to acquire samples from environment objects and build the color classes was programmed using the proposed algorithm. A screenshot from this application is shown in Figure 6. This application allows the selection of pixels from images to create clouds of samples for each desired color. The efficiency of the classification approach allows that, interactively, after selecting new samples from the image, the color class is updated, producing the implicit surface on the right area of the application, and classifying pixels on the image in the lower left area of the application. However, in order to produce a more efficient online classification, the application produces a look-up table

that will reduce the online color classification step to a single query in a 3D array for each pixel.

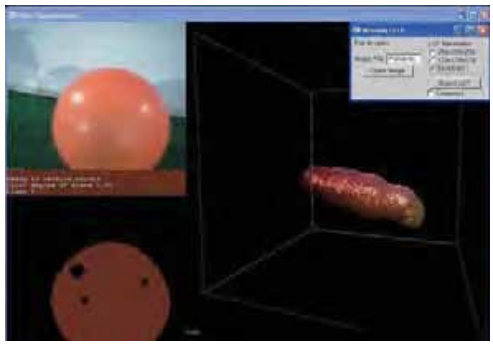


Fig. 6. Tool for color image classification.

The produced primitives and the resulting implicit surfaces fit tightly the samples used in the segmentation process, producing a precise representation of the color class. The blending degree can be also modified interactively.
Figure 7 provides an example on how changes on this parameter affect the result of the color class and of the classification itself. Smaller blending degree produces robust color identification, while larger blending degree produces more precise results. This attribute, together with the threshold for the primitive radius, is useful to resolve overlapping between nearby color classes, and provides better control on the class tolerance.

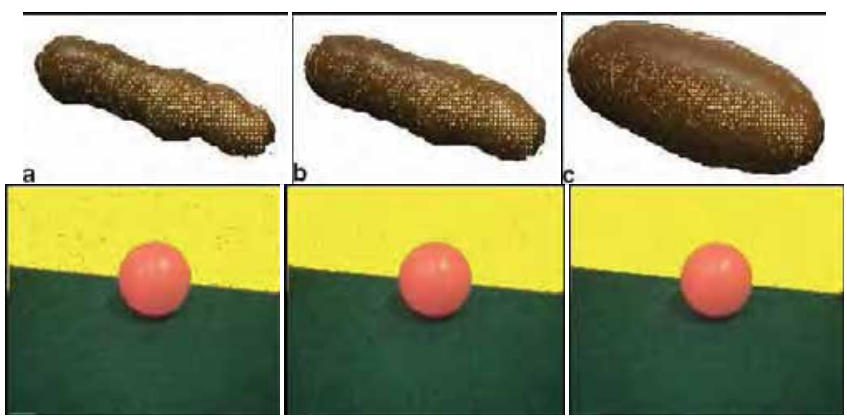


Fig. 7. Implicit surface with different blending degree, and associated segmentation for yellow in a sample image. a) $\rho = 2$, b) $\rho = 1.5$, c) $\rho = 1$.

Color classification was also tested under different lighting conditions comparing the results with the parallelepiped threshold technique. Images processed with the implicit surface approach are better identifying colors, even under extreme illumination changes. This test is shown in Figure 8. In addition, this figure also shows the color subspace produced by the

color class. The parallelepiped threshold approach produces some misclassification problems that are visible in the lower images from Figure 8.
In addition to a higher tolerance to illumination changes, the proposed approach could be extended to dynamically adapt the color subspace, by using a region growth approach as the one presented here in the segmentation algorithm.
The multilevel classification technique improved the efficiency of previous techniques while producing a better quality image segmentation, and higher tolerance to illumination changes. The evaluation of this algorithm was simulated on a Pentium IV at 1.6 Ghz desktop computer, using real information from the robot.
The seed extraction process used a set of color classes with a 0.5 threshold value. As the entire regions for the field lines and the field itself are not required, this step is used to identify lines and borders of the field, and these colors are discarded from the region growth step. The first step of this simulation takes an average of 16 ms.

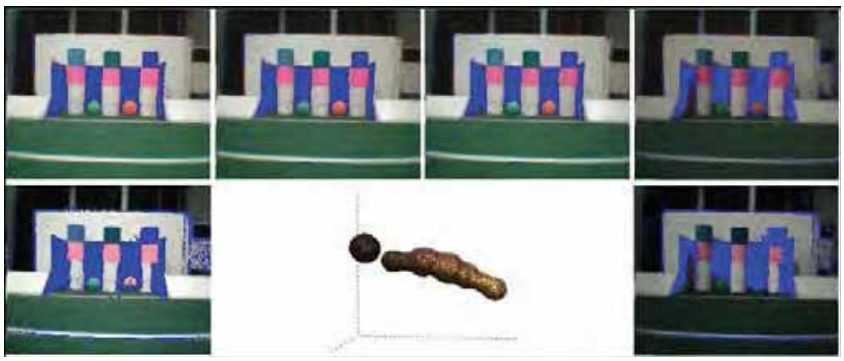


Fig. 8. Tolerance to changing lighting conditions. The yellow color is being classified and replaced by blue. Upper row: Color classification using an implicit surface threshold under different lighting conditions. Lower row, middle: Color subspace used for the images on the upper row. Lower row, left and right: Color classification using parallelepiped thresholds.

Once the seeds are extracted, the region growth stage is executed, using a set of color classes with a 1.0 threshold value. This step takes an average of 24 ms.
Some results of both steps of the segmentation are seen in Figure 9.

7. Conclusions and Future Work

The presented color classification technique shows a good approximation for color subspaces without the need of transforming the color signal of pixels. The produced implicit surface binds tightly the cloud of color samples, reducing possible overlapping problems with other color classes. The use of a look-up table was an efficient method that allows the classification of a single pixel with a single lookup on a 3D array.
In addition, evaluation of this algorithm under varying lighting conditions showed a better color classification than that produced by other color classification methods. The main reason behind this benefit is the tight approximation of the color class to the cloud of

samples; as overlapping problems are reduced, a larger number of samples with different color intensities and tones may be selected to produce the color class.

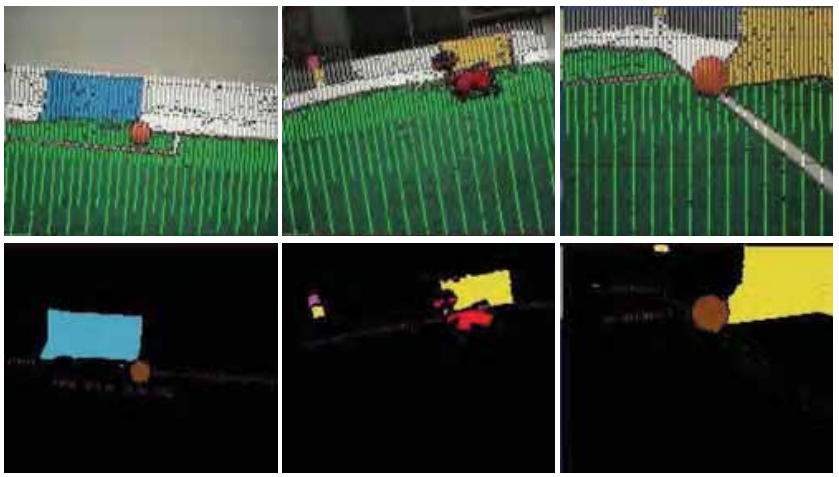


Fig. 9. Top row: Results of scanlines seed extraction stage. Bottom row: Results of region growth stage.

Implicit surface was shown as a well suited threshold surface for color classification, as it is simple to identify whether a color lies inside or outside the color class by just evaluating the value of the implicit function. In addition, once the parameters for the primitives of the implicit surface have been estimated, the tolerance to noise and to changing lighting condition can be easily customized by changing both the blending degree and the primitive threshold for the desired color class, requiring no further execution of the entire classification process.

While the entire classification process has not yet been implemented to be executed by a mobile robot, its offline classification process runs at interactive times. This may allow the later implementation of this technique on a mobile robot to dynamically modify color classes using some similarity criteria.

The presented color image segmentation technique combines the advantages of different techniques reducing, at the same time, some of their drawbacks. The use of scanlines reduces the number of pixels evaluated by the classification and segmentation process, and helps to discard external noise due to objects outside the areas of interest of the image.

Using a high-threshold color class as homogeneity criterion for the region growth algorithm provides with a greater control on the growth process, avoiding flooding problems on low-contrast boundaries.

Besides, the use of region growth completes the information extracted from scanlines. Pixels found for a region are the same that would be obtained by processing the entire image, but scanlines avoid processing pixels with no useful information, improving the efficiency of the segmentation phase.

In addition, scanlines should only find one pixel from an object to identify it, as the SRG step will identify the rest of the pixel. This reduces the required number of scanlines to obtain a good image segmentation.
Many possible improvements can be considered. First, the *k*-means algorithm requires an initial number of primitives. It would be desirable that the number of primitives would also adapt to the nature of the color class. For this purpose, different initialization algorithms may be incorporated to produce better results.
Another possible improvement lies on the nature of the primitives. Spherical clusters are well suited as an initial approach, as Euclidean distance is a simple metric to evaluate distance to a cluster. However, different distance metrics may provide with ellipsoidal or other-shaped clusters, which may produce as a result the use of less primitives and possibly a tighter adaptation of the color class to the samples.
In addition, the online adjustment of the implicit surface threshold may be used by an online dynamic color classification technique that will adapt to varying lighting conditions. The required samples for this segmentation may be obtained from the region growth algorithm. When new pixels are included into a region, the dynamic classification algorithm would add these pixels to the cloud of samples for a given color class. Then, after receiving a set of pixels, the color classification process would be executed and the color class updated, producing a new classification suitable for current illumination conditions. This mechanism would also require an additional step to discard invalid samples and to validate new samples, either by the frequency of a sample, its last appearance on a frame, or its distance to the previous color class. This technique would produce a reliable, non-supervised color segmentation technique that could adapt constantly to different conditions from the environment.

8. References

Alvarez, R.; Millán, E.; Swain-Oropeza, R. & Aceves-López, A. (2004). Color image classification through fitting of implicit surfaces, *9th Ibero-American Conf. on Artificial Intelligence (IBERAMIA*), Lecture Notes in Computer Science, Vol. 3315 (January 2004), pp. 677 – 686, ISSN 0302-9743.

Alvarez, R.; Millán, E. & Swain-Oropeza, R. (2005). Multilevel Seed Region Growth Segmentation. *MICAI 2005: Advances in Artificial Intelligence, 4th Mexican International Conference on Artificial Intelligence*, Lecture Notes in Artificial Intelligence, Alexander F. Gelbukh, Alvaro de Albornoz, Hugo Terashima-Marín, Ed., pp. 359 – 368, Springer, ISBN 3540298967, Berlin / Heidelberg.

Bishop, C. M. (1996) *Neural networks for pattern recognition*, Oxford University Press, ISBN: 0198538642, United States

Bloomenthal, J. & Wyvill, B. (1997) *Introduction to Implicit Surfaces*, Morgan Kaufmann Publishers Inc, ISBN: 155860233X.

Bruce, J.; Balch, T. & Veloso, M. M. (2000). Fast and inexpensive color image segmentation for interactive robots, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, Vol. 3, (October 2000), pp. 2061–2066, ISSN 0932-8092.

Castleman, K. (1996). *Digital image processing*, Prentice Hall, ISBN: 0132114674

Forsyth, D. & Ponce, J. (2002). *Computer Vision: A Modern Approach*, Prentice Hall, ISBN: 0130851981, United States

Jüngel, M. (2004). Using layered color precision for a self-calibrating vision system. *RoboCup 2004: Robot Soccer World Cup VIII*, Lecture Notes in Artificial Intelligence, Daniele Nardi, Martin Riedmiller, Claude Sammut, José Santos-Victor, Ed., pp. 209-220, Springer, ISBN 3540250468, Berlin / Heidelberg.

Langetepe, E. & Zachmann, G. (2006). *Geometric data structures for computer graphics*, A. K. Peters, Ltd., ISBN: 1568812353, United States.

Lim, C. T.; Turkiyyah, G. M.; Ganter, M. A. & Storti, D. W. (1995). Implicit reconstruction of solids from cloud point sets, *Proceedings of the third ACM symposium on Solid modeling and applications*, pp. 393–402, ISBN: 0-89791-672-7, Salt Lake City, Utah, United States, 1995, ACM Press.

Ricci, A. (1973) A constructive geometry for computer graphics, *The Computer Journal,* Vol. 16, No. 2, (May 1973), pp. 157–160, ISSN: 0010-4620.

Von Hundelshausen, F. & Rojas, R. (2003). Tracking regions. *RoboCup 2003: Robot Soccer World Cup VII*, Lecture Notes in Computer Science, Daniel Polani, Brett Browning, Andrea Bonarini, Kazuo Yoshida, Ed., pp 250-261, Springer, ISBN: 3540224432, Berlin / Heidelberg.

Wasik, Z. & Saffiotti, A. (2002). Robust color segmentation for the Robocup domain. *International Conference on Pattern Recognition*, pp. 651-654, ISBN: 076951695X, Quebec City, Canada.

Watt, A., & Policarpo, F. (1998). *The computer image*, Addison Wesley, ISBN: 0201422980

9

Intelligent Global Vision for Teams of Mobile Robots

Jacky Baltes and John Anderson
Autonomous Agents Laboratory, Department of Computer Science,
University of Manitoba
Winnipeg, Canada

1. Introduction: Global Vision

Vision is the richest of all human senses: we acquire most of our perceptual information through vision, and perform much of our own vision processing with little conscious effort. In contrast, dealing intelligently with the enormous volume of data that vision produces is one of the biggest challenges to robotics. Identifying an object of interest in a random camera image is a difficult problem, even in domains where the number of possible objects is constrained, such as robotic soccer. This difficulty increases in magnitude when attributes of interest involve change such as movement, and thus require both state information and examining change in visual images over time. Visual analysis also involves many subtle problems, from very low-level issues such as identifying colours under changing lighting conditions, to higher-level problems such as tracing the path of an object under conditions of partial occlusion, or distinguishing two objects that are next to one another but appear as one larger object.

Humans deal with vision through very specialized massively-parallel hardware, coupled with a broad range of commonsense knowledge. Neither of these is currently feasible to apply to a mobile robot platform. While mobile processors are becoming more powerful, we are still far below what is required to process vision at a human level, and common-sense knowledge has always been one of the most difficult problems associated with artificial intelligence. Vision on mobile robots is thus largely about producing heuristic solutions that are adequate for the problem domain and allow the available hardware to process vision at the frame rate required.

Vision in robots may be divided into two types. The first of these, *local vision*, involves providing a first-person perspective using a camera attached to the robot itself. Local vision shares many of the same problems that humans deal with in visual analysis, such as determining whether motion in a visual stream is due to the motion of objects captured by the camera, or the motion of the agent itself (ego motion). In a local vision setting, each member of a team of robots receives its own unique camera feed and be responsible for analyzing and responding to that feed.

There are a number of reasons why local vision may not be preferable in a given application, the foremost of which is the heavy requirement for computational resources. When each robot must perform its own visual processing, it must be able to carry enough on-board processing to do so, which may not be possible in smaller robots or applications with competing resource needs. The fact that each robot is entirely responsible for its own

vision will also mean that there will be significant redundancy in processing across a team of robots in many applications as well. Local vision may also be undesirable in applications where a large number of very simple robots may be able to do the job of a few complex robots, in environments where shared vision is amenable (that is, where a unique perspective for each individual is unnecessary), and in educational environments where it is desirable to separate the problems of computer vision from the rest of robotics. In these domains, the second form of vision, *global vision*, is often preferred. Global vision provides a single third-party view to all members of a robot team, analogous to the view of a commentator in a soccer game.

Global vision shares many of the problems associated with local vision. Objects of interest must be identified and tracked, which requires dealing with changes in appearance due to lighting variation and perspective. Since objects may not be identifiable in every frame, tracking objects across different frames is often necessary even if the objects are not mobile. The problem of identifying objects that are juxtaposed being viewed as one larger object rather than several distinct objects, and other problems related to the placement and motion of objects in the environment, are also common.

In domains such as robotic soccer, where pragmatic real-time global vision is large part of the application, many of the more difficult problems associated with global vision have been dealt with through the introduction of artificial assumptions that greatly simplify the situation. The cost of such assumptions is that of generality: such systems can only operate where the assumptions they rely upon can be made. For example, global vision systems for robotic soccer (e.g. (Bruce & Veloso, 2003; Browning et al., 2002; Simon et al., 2001; Ball et al., 2004)) generally require a camera to be mounted perfectly overhead in order to provide a simple geometric perspective (and thus ensure that any object is the same size in the image no matter where in the field of view it appears), simplify tracking, and eliminate complex problems such as occlusion between agents. If a camera cannot be placed perfectly overhead, these systems cannot be used. Such systems also typically recognize individuals by arrangements of coloured patches, where the colours (for the patches and other items such as the ball) must be pre-defined, necessitating constant camera recalibration as lighting changes. Such systems can thus only operate in environments where lighting remains relatively consistent.

While such systems will always be applicable in narrow domains where these assumptions can be made to hold, the generality lost in continuing to adhere to these assumptions serves to limit the applicability of these approaches to harder problems. Moreover, these systems bear little resemblance to human vision: children playing with remote-controlled devices, for example, do not have to climb to the ceiling and look down from overhead. Similarly, human vision does not require significant restrictions lighting consistency, nor any specialized markings on objects to be tracked. In order to advance the state of the art in robotics and artificial intelligence, we must begin to make such systems more generally intelligent. The most obvious first steps in this direction are considering the assumptions necessary to make a global vision system operate, and then to find ways of removing these.

Our approach to real time computer vision arises from a desire to remove these assumptions and produce a more intelligent approach to global vision for teams of robots, not only for the sake of technological advancement, but from a pragmatic standpoint as well. For example, a system that does not assume that a camera has a perfect overhead mount is not only more generally useful, but requires less set-up time in that a perfect overhead mount does not need to be made. Similarly, an approach that can function in a wide range of lighting conditions saves the time and expense of providing specialized

lighting for a robotic domain. Over the past six years, we have developed a series of real-time global vision systems that, while designed for the robotic soccer domain, are also generally useful anywhere global vision can be used. These systems have been used in RoboCup and FIRA robotic soccer competitions by ourselves and other teams, and have also been employed in such applications as robotic education and imitation learning. All are open source, and can be easily obtained by the reader for use or as a basis for further research work (Baltes & Anderson, 2006).

Each of the systems we have developed deals with some of the assumptions normally associated with global vision systems, and thus produces a more generally intelligent approach. This chapter overviews the work necessary to deal with these assumptions, and outlines challenges that remain. We begin by examining the steps necessary to deal with a more general camera position, how objects can be tracked when the camera is not perfectly overhead, and how an overhead view can be reconstructed from an oblique camera capture. This necessitates dealing with objects that are occluded temporarily as robots move around on the field, and also requires dealing with three dimensions rather than two (since the height of an object is significant when the view is not a perfect overhead one). We then turn to dealing with assumptions about the objects being tracked, in order to minimize the need for recalibration over time, and to make global vision less vulnerable to problems of lighting variability. We examine the possibility of tracking objects using only the appearance of the object itself, rather than specialized markers, and discuss the use of machine learning to teach a global vision system about the objects it should be tracking. Finally, we examine removing the assumption that specific colours can be calibrated and tracked at all, in order to produce a vision system that does not rely on perfect colour calibration to recognize objects.

2. Doraemon: Real-Time Object Tracking from an Oblique View

Doraemon (Anderson & Baltes, 2002; Baltes, 2002) is a global vision system that allows objects to be tracked from an oblique camera angle as well as from an overhead view. The system acts as a server, taking frames from a camera, and producing a description of the objects tracked in frames at regular intervals, sending these over a network to clients (agents controlling robots, for example) subscribing to this information stream. Fig. 1 is a sample visual frame used as input to Doraemon to illustrate the problems involved in interpreting visual images without using a perfect overhead viewpoint. The image is disproportionate in height because it is one raw field from the interlaced video stream provided by the camera. It is easy to see that colour features are hard to extract, in part because the shape of coloured patches are elongated by the visual perspective, and in part because colour is not consistent across the entire image.

Fig. 1. A sample visual frame taken from an oblique angle.

Fig. 2. Tsai Camera Calibration used in Doraemon.

In order to be able to track images from an oblique angle, a calibration must be provided that allows an appropriate translation from a particular pixel in a visual frame to a coordinate system in the real world. The calibration process used by Doraemon, described in detail in (Anderson & Baltes, 2002), utilizes the well-established Tsai camera calibration (Tsai, 1986), which can compute a camera calibration from a single image. This method computes six external parameters (the x, y , and z coordinates of the camera position, and angles of roll, pitch and yaw) and six internal parameters using a set of calibration points from an image with known world coordinates. This requires a set of coordinates to be imposed on the world via a sample visual image. Since Tsai calibration normally requires at least 15 calibration points (i.e. points with known x,y coordinates), a calibration carpet with a repetitive grid pattern is used to easily provide a significant number of points. The known grid size is input to the system, and the coloured squares can be then be selected by the user and the calibration points obtained from the square centers (Fig. 2). Even using an oblique view of the playing field, the calibration results in object errors of less than 1 cm. To make calibration more flexible, we also define a rotation matrix on the field that allows the calibration to be adjusted (for example if the camera shifts during play) without having to recalibrate using the carpet.

Objects in Doraemon are identified by the size and arrangement of coloured patches. The simplest objects may be simply a single coloured area of a given size - e.g. a ball might be described as an orange item 5cm in diameter. More sophisticated items (e.g. individual robots) are identified using unique arrangement of coloured patches on the top surface, as seen in Fig. 1 (e.g. a blue patch for the front of all robots on one team, with an arrangement of other colours uniquely identifying each team member). The system is thus heavily dependent on accurate colour models. Doraemon uses a sophisticated 12 parameter colour model that is based on red (R), green (G), and blue (B) channels as well as the difference channels red-green (R-G), red-blue (R-B), and green-blue (G-B). The channel differences are less sensitive to lighting variations than the raw channels, and allow more robust colour recognition than the raw channels alone. While there are other models that are less sensitive to brightness, (for example, HSI), this approach attempts to balance sensitivity with computational resources. The channel differences are similar to the hue values used in HSI, for example, while this model is less computationally expensive.
Colours of interest are defined using a colour calibration procedure, during which areas of the visual image intended to be matched to a particular named colour are selected. This reliance on colour, like that of other global and local vision systems, limits generality and forces recalibration to be performed when lighting conditions change. Moving beyond this

dependence on colour will be described in Sections 3 and 4. Once colours are defined, camera images can be colour thresholded and particular colour patches can be recognized in an image.

The size of any patch of colour can be determined by its position on the field, since the perspective of the field is known through calibration. This still requires a model describing the arrangements of the coloured patches marking objects as well as their heights above the playing field, since, for example, an object that is 50cm tall will have markers of the same height appear differently in the camera image than that of an object that is flush with the playing field surface. The descriptions of the size, colour, arrangement, and height of objects to be recognized are described in a configuration file.

Each frame is colour thresholded and the recognized patches are matched against the size and configuration information provided. Not every object will be recognized in every frame, since lighting fluctuations, for example, may make some spots difficult to recognize across the entire field area. To compensate for this, the locations of recognized objects in previous frames are used both to infer likely positions in future frames and to calculate the speed and orientation of motion of tracked objects.

Occlusion in robotic soccer is normally not an issue for tracking robots, even with an oblique camera, since the markers are on top of the robots and are thus the highest points on the field. Occlusion certainly happens when tracking the ball, however, and is also possible in any tracking scenario where obstacles on the field could be taller than robots. There is also the possibility that robots may abut one another, presenting a display of coloured patches that is similar to a different robot altogether, or presented in such a way that no one robot is easily recognizable. These situations are dealt with by tracking objects over time as well - an object may be lost temporarily as it passes behind an obstacle, or may be more momentarily unrecognized due to abutting other tracked objects - because objects are intended to be in motion, such losses will be momentary as new information allows them to be disambiguated.

Doraemon transmits information about tracked objects (position, orientation, velocity) in ASCII over Ethernet to any client interested in receiving it. A sample message is shown in Fig. 3.

```
7 6188 0.000290976 ; #defined objects, frame#, time diff. from last frame
1605.82 -708.394 1321.44 ; x, y, z coordinates of camera
2 spot1 Found 1232.5 416.374 0 0 0 0 ;information about each defined object
2 spot2 Found 1559.22 417.359 0 0 0 0
2 spot3 Found 1260.55 812.189 0 0 0 0
2 spot4 Found 902.726 1002.43 0 0 0 0
2 spot5 Found 746.045 735.631 0 0 0 0
1 ball1 Found 1677.99 1205.55 50 0 -2.75769 1.19908
0 car54 Found 1783.53 873.531 100 2.63944 1.47684 -6.49056
```

Fig. 3. A sample ASCII message from Doraemon.

The first line of each message contains the number of objects that video server is configured to track, followed by the video frame number and time difference in seconds between this message and the previous one. The next line contains the x, y, and z coordinates of the camera, and following this is a line for each object being tracked. Each of those lines consists of a numeric object class (e.g. a ball, robot, etc.), the unique defined identifier for the object, whether the object was located in the current frame or not, the x, y, and z coordinates of the

object, the orientation of the object in radians, and the velocity of the object in mm/second in the x and y dimensions.
Doraemon was later extended (Baltes & Anderson, 2005) to provide its own reconstructed overhead view through interpolation, which allows the perspective distortion created by the oblique camera angle to be corrected, allowing objects to be more accurately tracked. While this interpolation does slow down the vision process, it provides a remarkable improvement in image quality. As an example, Fig. 4 shows Doraemon´s reconstruction of the oblique view shown in Fig. 1.
Doraemon takes several steps beyond global vision systems that maintain a fixed overhead camera in terms of being able to deal with the real world. It is quick to calibrate and simple to recalibrate when this is necessary (e.g. due to camera shift or changing lighting during use).
However, there are still significant assumptions about the domain that affect the system's generality. Doraemon is heavily dependent on good colour models, something that is not easy to maintain consistently over time in real-world domains without recalibration, and relies on a fairly naive model for dealing with occlusion. Dealing with these assumptions is the subject of the remaining sections in this chapter.

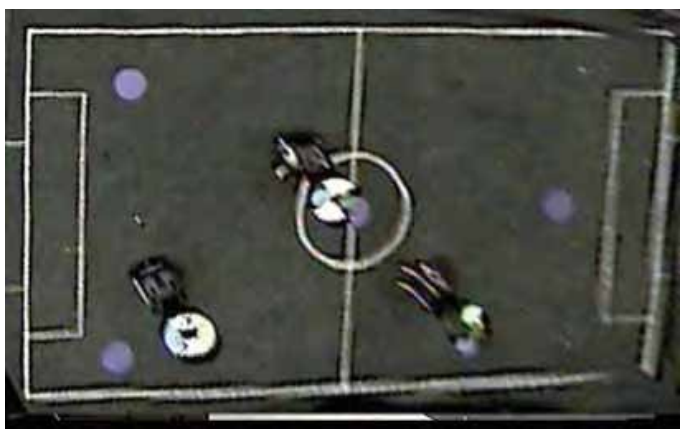


Fig. 4. Doraemon's overhead reconstruction (using average gradient interpolation) of the camera image shown in Fig. 1.

3. Ergo: Removing Dependence on Predefined Colours

The reliance on colour thresholding by both Doraemon and related systems places some severe restrictions on the applicability of a global vision system. Not only are lighting variations a problem, but the colours themselves must be chosen so that there is enough separation between them to allow them to be distinguished across the entire field of play, and the quality of the camera used is also a major issue. In practice, even with the extra colour channels employed by Doraemon tracking is practically limited to around 6 different colours by these restrictions.
To increase the applicability of global vision to a broader array of real-world tasks, as well as to increase the robustness of the system in robotic soccer, we focussed on two major

changes in approach: the use of motion detection to focus on areas of interest in the field, and different methods of marking objects that deemphasize the use of colour. These and other extensions were combined into a new generation of global vision software, known as Ergo (Furgale et al., 2005).

One additional pragmatic step was also necessary in Ergo in order to attain a comparable frame rate as that employed in the original Doraemon: the resolution of interpolated images was decreased, in order that interpolation did not inordinately slow down visual analysis. The result of this introduced an additional challenge, in that a typical 5cm soccer ball would now occupy only a 1-4 pixel range in the reduced resolution, allowing a ball to easily be interpreted as noise (Fig. 5).

Rather than performing direct colour thresholding of camera images, Ergo thresholds for motion across pixels in each frame compared to a background image. A number of common thresholding techniques (using pixel intensity and distance in colour space, with global and local thresholds) were experimented with under poor lighting conditions and with common domain elements such as the presence of field lines and aliasing between camera frames. None of the common approaches were adequate in avoiding losing information from dark parts of the image while removing noise from lighter portions. In the end, an adaptation of $\Sigma\Delta$ background estimation (Manzanera & Richefeu, 2004) was employed, which provides a computationally inexpensive means of recursively estimating the average colour and variance of each pixel in a camera image.

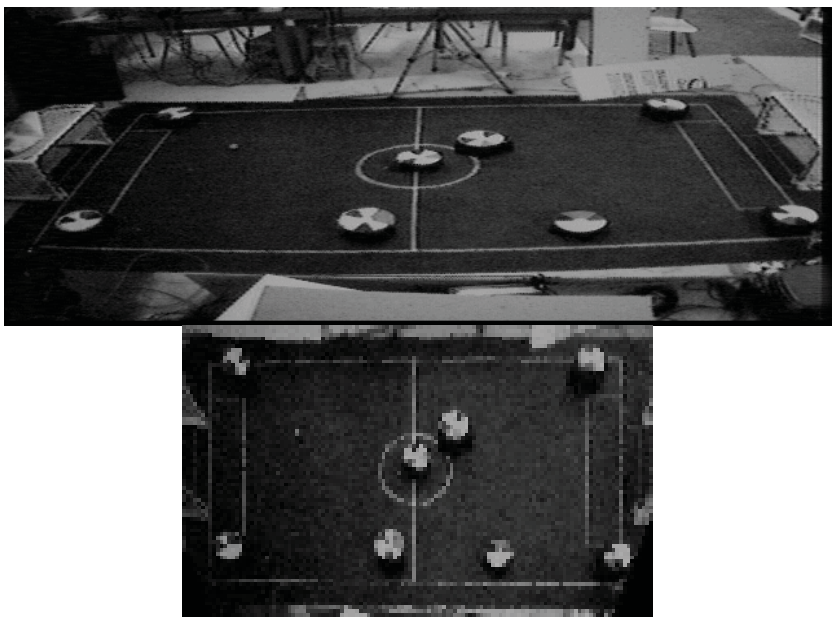


Fig. 5. Captured field and corresponding low-resolution interpolated image in Ergo. Note that the ball is easily visible in the upper image, but blends with noise on the field lines in the lower.

Detecting motion involves setting a threshold above which variation across pixels will be considered to be motion. In experimenting with this, it was found that increasing a global threshold enough that all noise would be eliminated also had the effect of eliminating any object of the size of a typical robotic soccer ball, since the size of such an object in the image (<=4 pixels) is easily interpreted as noise. To deal with this, a means was required to consider variation more locally and eliminate noise, while still being able to pick up the motion of small objects, and so a combination of local and global thresholding was employed. A threshold is set for each pixel by examining the variance for each pixel in the background image, then applying a convolution (1) in order to consider a pixel's variance across its 9-pixel neighbourhood. This local threshold is then scaled by a global threshold. To detect motion, each incoming image has its sum-squared error calculated across all pixels against the background image, the same convolution is applied to the result, and each value is compared to its corresponding pre-computed threshold. The use of the convolution has the effect of blending motion in small areas to eliminate noise, while making the movement of small objects such as the ball more obvious by also considering small changes in neighbouring pixels.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{1}$$

This thresholding process causes motion to be separated from the background, after which the region growing algorithm of Bruce et al. (2000) is employed to generate a list of regions to match against the descriptions of objects that Ergo is tracking.

Since Doraemon relied on patterns of coloured blobs to identify moving objects such as robots, a change in pattern representation was necessary in Ergo in order to remove the dependence on predefined colours. The two basic requirements of a representation are the determination of identity and orientation (since the remaining item of interest, velocity, can be obtained through knowing these over time). Previous research (Bruce & Veloso, 2003) has shown that asymmetrical patterns can be used to allow a range of objects can be identified with fewer colours, and these ideas were extended to develop a representation and associated matching mechanism for tracking objects while minimizing the need for predefined colours.

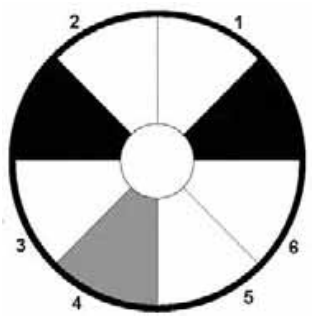


Fig. 6. A new approach to labeling objects for tracking (Furgale et al., 2005): fixed black areas allow orientation to be recognized, while white and non-white values in locations 1-6 represent identity

The marking approach designed for Ergo divides the marker for a robot (or similar moving object) into a circular series of wedges (Fig. 6). Two black wedges are the same on all robots, allowing a tracking algorithm to determine the labelled object's orientation. The remaining six wedges are marked with white and non-white (i.e. *any* colour other than white or black) to allow the determination of identity. Marking only two of these segments would allow up to twenty-one individuals to be identified uniquely (the centre is left open for a possible team identifier if desired).

An associated algorithm for identifying objects assumes that such a marking system is in use, and begins with a set of hypotheses of objects of interest, based on the regions of the camera image that have been flagged as motion. The original image is reinterpolated with a higher resolution in (only) three concentric circular strips of pixels (each 64 pixels long) around the centre of each region of motion. This allows enough high-resolution interpolated area to more accurately determine the marking pattern without the computational demands of large-scale interpolation. The mean is taken across these to reduce noise and error, resulting in a single array of 64 elements, providing an encoding for that region of motion that can be matched against the labeled pattern described above. To be able to match the pattern in this strip, two boundaries must be determined in this strip: the boundary between black and the marker that is neither black nor white, and the boundary between that and white. These boundaries are determined using a histogram of intensity values produced as part of the reinterpolation. The black-other threshold can be approximated based on the fact that any point near the centre will be 25% black. The other-white boundary is arrived at by starting a marker at the top of the range of the histogram, and then iteratively replacing that with that average of the weighted sum of the histogram counts above other-white and those below other-white. It is possible to avoid this process based on the pattern if a known pattern is being searched for, so it is not required in all cases.

Once these thresholds are available, the identification algorithm begins by looking at for the two black regions, and the average of the centre between these is the orientation. These wedges also provide the plane on which the pattern, and based on that plane the recorded centre of the object is refined. The remaining parts of the interpolated strip are then partitioned relative to the black wedges and the identification pattern can then be determined by counting the number of white wedges and the number of wedges that are neither white nor black.

This identification algorithm is very effective and computationally minimal, but is complicated in application by two factors. First, the list of regions of motion may be significantly larger than the number of objects to be tracked (due to extraneous movement by other objects, for example): large enough that this algorithm cannot process them all in real time in the data directed manner that would be ideal. Second, successful identification of an object relies on an accurate centre point.

If two or more moving objects appear in close proximity to one another (or even partly occlude one another), motion analysis will view this as one large region of motion, with a centre that will not be helpful in identifying anything. This algorithm thus needs to be applied in a more goal-directed manner, and have some means of dealing with clumps of objects.

Ergo deals with these problems by tracking objects across images, which provides for a goal directed application of this algorithm. Prior to motion analysis, every object found in the previous frame predicts its position in the next image based on velocity and time difference. Some objects may thus be found very quickly, since their centre point will be

predicted and can easily be confirmed using the identification algorithm. The area in the image occupied by object recognized during this phase is masked during motion analysis. This masking serves two purposes: it produces no hypothesis, since the object has already been dealt with, but it also may serve to remove one of a group of objects that may appear together in a moving region. Masking the area will then leave a smaller region and a smaller number of grouped objects (possibly only one, which can then be handled as any other object would).

For the remaining unrecognized objects, the region most closely predicted by each is selected. If an appropriate-sized region is found near the predicted location, it is passed to the identification algorithm, along with the hypothesized identity to speed up the identification process. This step, along with those detailed above, turns the identification process into a largely goal-directed one, and the vast majority of objects in any frame are recognized in this manner. Data-directed processing is still required, however, to deal with any objects that remain unidentified at this point.

There are realistically two possibilities for the remaining objects: a region of motion is outside the predicted area for the object, or it is part of a clump of objects occupying a larger region. To deal with the former, Ergo examines the sizes of all unexplained regions of motion, and if it is a size that could suitably match an object of interest, it is passed to the identification algorithm. In the case of multiple objects occupying the same space, the regions of interest will be those that are too large for any one object. If any of these regions were to contain more than one object, at least one recognizable object will be touching the edge of the region, and so the edge is where recognition efforts are focussed.

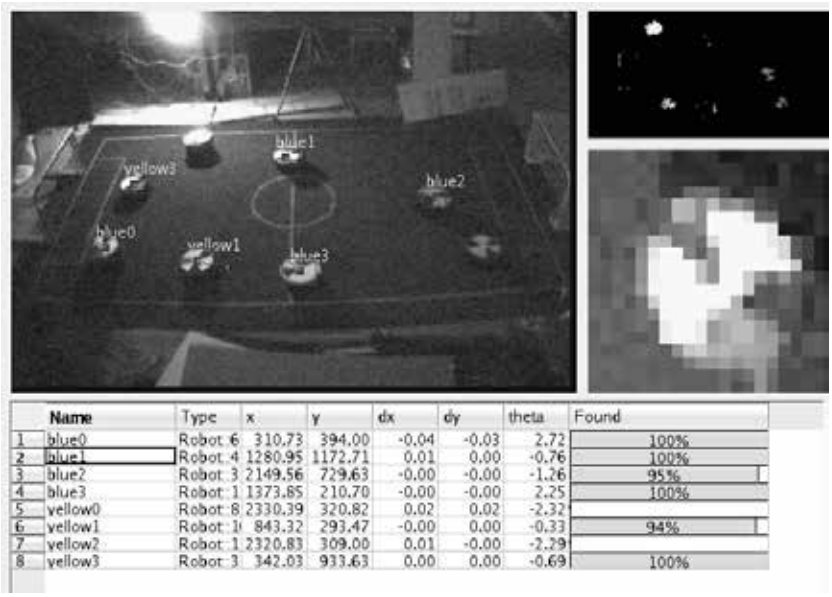
To analyze regions that could be multiple robots, extra samples are taken one object-radius in from the region's edge and obtain a set of encodings that should cross the centre of at least one object if multiple objects are in the region. From this, those that are at least one object diameter long are chosen, and the identification algorithm above is run on each of these using each pixel as the potential centre of the object. If any object is identified, it is masked from the region in the next frame, allowing further objects to be distinguished in subsequent frames. This could be repeated in the analysis of the same frame to distinguish further objects, but since Ergo can determine likely positions of unrecognized objects just as Doraemon could in frames where some objects were unrecognized, this strikes a balance toward computational efficiency.

Not every object is large enough to be labeled using the scheme shown in Fig. 7, nor do all objects need an encoding to uniquely identify them. In robotic soccer, for example, the ball is physically unique, and its nature does not require a pattern for identification. The use of motion tracking to distinguish an element as small as the ball has already been described.

In frames where this motion tracking does not allow the ball to be found, the ball's location is predicted from the previous frame, and an area eight times the ball's size is scanned for regions of the correct size and dimension after colour thresholding. Colour thresholding here is simply used to distinguish regions at all given that motion detection has failed, and no predefined colours are employed.

These techniques allow Ergo to perform well under very challenging conditions. Fig. 7 illustrates a screenshot from an extreme example, with lighting positioned across the viewing area, causing a wide disparity in brightness, and significant shadowing. Motion tracking is shown in the upper right, and the system output in the bottom of the image. All

robots are identified except for one completely hidden in shadow, and the other in complete glare from the lighting source.



	Name	Type	x	y	dx	dy	theta	Found
1	blue0	Robot 6	310.73	394.00	-0.04	-0.03	2.72	100%
2	blue1	Robot 4	1280.95	1172.71	0.01	0.00	-0.76	100%
3	blue2	Robot 3	2149.56	729.63	-0.00	-0.00	-1.26	95%
4	blue3	Robot 1	1373.85	210.70	-0.00	-0.00	2.25	100%
5	yellow0	Robot 8	2330.39	320.82	0.02	0.02	-2.32	
6	yellow1	Robot 1	843.32	293.47	-0.00	0.00	-0.33	94%
7	yellow2	Robot 1	2320.83	309.00	0.01	-0.00	-2.29	
8	yellow3	Robot 3	342.03	933.63	0.00	0.00	-0.69	100%

Fig. 7. Using Ergo under very poor lighting conditions (Furgale et al., 2005).

Ergo has gone a long way in making a global vision system more applicable to real-world situations, in that it has both removed the need for a fixed overhead camera as well as any predefined colours, and thus can operate across a much broader range of condition s than previous systems. There are still assumptions it operates under, the largest being that a pattern can be used to consistently identify objects that need to be tracked. In the remainder of this chapter, we will explore the possibility of tracking objects without such patterns.

4. Removing Dependence on Predefined Patterns

The ability to move beyond predefined colours or patterns for identifying objects is important in vision, for a number of reasons. From an immediate practical standpoint, scalability is always an issue. Even when using patterns without colour, there is a very finite amount of variation that can fit on a small pattern and be recognized reliably at a distance. While there are alternative approaches, such as just as arranging objects in predefined patterns before any movement begins and then tracking motion, such approaches can only operate for a short time before robots are misidentified as they move about. Once misidentified, there is no easy way to re-establish identity without stopping to do this manually.

The issue of generality is much more significant in the long term than scalability, however. While patterns are employed by humans during visual tracking (e.g. in soccer, teams wear

structured uniforms involving both colour and pattern to distinguish themselves visually for the benefit of players and spectators), such patterns do not have to be pre-programmed. We need only watch an ongoing soccer game for few seconds to understand the pattern and be able to track it without any significant conscious effort. Humans can also switch between activities quickly, while equally quickly adapting to the details necessary for visual tracking in the new domain.

In order to make a computer vision system truly intelligent, we must work toward this level of generality by removing assumptions of predefined patterns and demonstrating similar adaptability to that observed in human vision. In order for this to be achieved in a vision system, one of two things must happen: either additional sources of information must be exploited to make up for that provided by assumed patterns, or the system must be able to adapt to patterns itself over time. Both of these are significantly beyond the level of production vision systems at the present time, and represent some of the core ideas for improving this technology in future. In the following subsections, we present recent work in both of these areas, and then conclude by summarizing some of the issues yet remaining.

4.1 Object Tracking Based on Control Information

There are numerous techniques used by humans that can be exploited in an intelligent system for visually tracking objects. One of the most powerful can be seen any time a group of children operate remote-controlled vehicles. If the vehicles all look alike, a child quickly realizes that the one he or she is controlling can be identified by its response to the control commands being sent. While vision alone can be used to track the vehicle under control after it has been identified, when it is lost (e.g. has crossed paths with several other identical vehicles), this control information can be used to re-identify the vehicle. Such information can also be used to further confirm identity throughout the control process.

In recent years we have been working toward extending the abilities of our global vision systems by intelligently applying such control information. The original versions of Doraemon and Ergo both maintain the identity and velocity of objects being tracked, in the form of a hypothesis with an associated measure of likelihood. As already discussed in Sections 2 and 3, this information is used to predict the future positions of moving objects in subsequent frames, to allow a more goal-directed tracking process and to account for objects when they cannot be recognized in every frame. If objects are no longer visually distinct, in that there is no predefined identification pattern, there may also no longer be any way to obtain orientation information visually in a single frame (depending on whether markings are present to provide such information). However, the addition of control information affords a better predictor of future object locations, because control commands are presumably the single most important factor in future movement. This same control information can also indirectly supply orientation information if it is not otherwise available, since the orientation is also obtainable based on the object's response to commands. Control information is still imperfect, since commands may not be carried out correctly or even be received properly by a mobile device, and unpredictable events (collisions, outside forces) can affect objects' future positions as well. However, such information should provide enough information to reliably support object tracking even where objects are not otherwise visually distinct, much as it does for humans operating remote controlled vehicles.

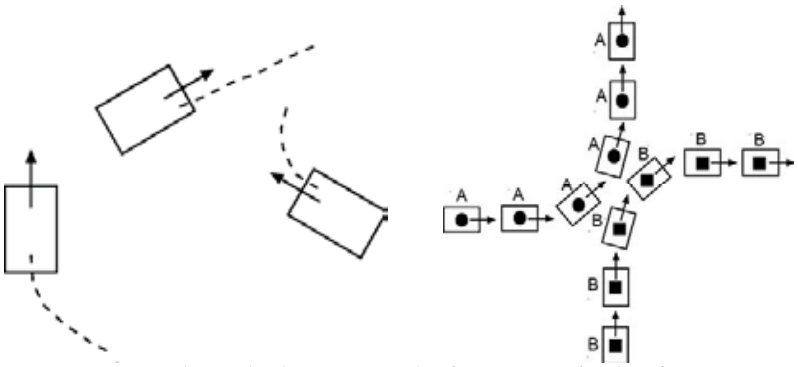


Fig. 8. Situations easily resolved using control information. Left: identification is simple given control command (dotted line) and velocity (arrow). Right: Two robots appear to cross paths unless control information is known.

In some situations, making use of control information is straightforward. Consider the situation depicted in the left side of Fig. 8. Here, three robots are depicted with current orientation indicated with an arrow, and the motion that would result from the current command shown by a dotted line. Assuming they are identical, they can easily be differentiated in subsequent frames, since the motion each will exhibit is very distinct. If a trail is maintained over time, there are equally straightforward situations that would not be obvious using vision alone. In the right side of Fig. 8, for example, two robots turn near one another, leading to the appearance of crossed paths. Using vision alone, misidentification during and after the turn is very likely, but this is easily resolved if the intended turns are known.

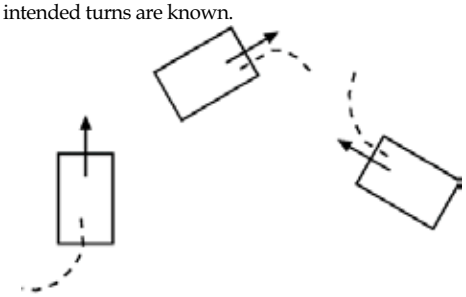


Fig. 9. A situation where control information alone cannot resolve object identities.

In other situations, this is not the case. Fig. 9, for example, shows the same robots as the left side of Fig. 8, but with different intended movements. Here, the two robots on the right cannot be differentiated based on control information, since they will both be moving similarly. The robot on the left in the same image can only be identified if orientation is known, since all robots are turning clockwise.

It can be seen from the latter example that current control information alone is not enough for reliable recognition. Even in a simple situation such as the first case, intended motion

still may not be seen in future frames, because of the external factors. An obstacle might prevent movement, robots might collide, a command might not be received, or the robot might not even be perceived in the next frame. Control commands are a valuable source of information, but we must deal with the fact that they are uncertain. If the result of each individual command is viewed as a separate piece of evidence supporting identity, situations such as that shown in Fig. 9 can be resolved using accumulated evidence over time (that is, an accumulated trace or trail supplied by an object's ongoing movement).

To extend our global vision systems to deal with the uncertainty involved with using command information, we experimented with moving from the *ad hoc* approach used in our production systems to a Bayesian approach (Baltes & Anderson, 2003a). This approach accumulates evidence in the form of traces of robot movement over time, and reasons probabilistically about the identities of robots given the traces of movement seen. The system uses evidence accumulated over a window of 100 frames, and computes an ongoing maximum likelihood hypothesis (h_{ML}) for each tracked object. The trace of the motion consists of the position, orientation, and state of the object, where state is one of turning clockwise, turning counter-clockwise, moving straight, or stopped.

$$P(h \mid D) = \frac{P(D \mid h) \cdot P(h)}{P(D)} \tag{2}$$

Bayes' formula (2, where $P(h \mid D)$ is the posterior probability of hypothesis h given the observation D, $P(D \mid h)$ is the prior probability of observing data D given hypothesis h, and $P(h)$ is the prior probability of hypothesis h) is used to compute the maximum posterior hypothesis for each robot's identity given prior observations.

In this application, the hypotheses are the form *Commands for robot 1 are observed as trace 1*, *Commands for robot 2 are observed as trace 2*, etc., where a trace is a set of positions and states over time. All traces are assumed to be equally likely, and so the prior probability $P(h)$ can be assumed to be uniform and thus ignored.

The system determines the maximum likelihood (ML) assignment of identities to robots that maximizes the posterior probability:

$$h_{ML} = \text{robot1} = (\text{trace1}, \text{command1}), \ldots = \arg\max_{h \in H} P(D \mid h) \tag{3}$$

To apply this calculation, $P(D \mid h)$ was determined through empirical observation to be approximately 0.7. In other words, the system detects and labels a stationary robot as stopped in 70% of the cases. To further simplify the calculation of the probabilities of the match of a command sequence and a motion trace, we assume that the probabilities of observing any feature are statistically independent.

The system computes the likelihood of all possible command traces to all observed traces and chooses the identity assignment that maximizes the likelihood for all robots. This approach was shown to work well in the soccer domain with a window of 100 frames and a small team of robots. The probability calculation as it stands is computationally expensive, which will result in limitations if scaled to teams of significant size, and so a significant element of future work will entail making such an approach efficient enough to use for large teams in real time. It is also possible that other applications might not require such a large evidential window, and thus may be less computationally demanding.

This is only one example of exploiting information to substitute for a pre-defined pattern. It is possible that other equally valuable sources remain to be discovered, and that this

approach could be used in conjunction with other specific means of exploiting information as well. In terms of generality, however, it is still reasonably specific, in that it assumes there is a manageable range of discrete commands whose effects could be understood and recognized by vision system. While object tracking in human vision for specific domains such as soccer does appear to include information analogous to that exploited here (in the form of what a player would be expected to do given the object and rules of the game, for example), a more general approach to object tracking would be able to move beyond this knowledge as well.

4.2 Inferring Orientation Without Prior Knowledge

Completely removing any type of predefined marker results in an object recognition problem that is an order of magnitude more difficult than those described thus far, since features for recognition must be discovered as part of the recognition process itself. These features may at times be shadowed or otherwise occluded, making the discovery of such features more difficult than the recognition process itself. However, a system that has the ability to discover useful patterns will be more generally applicable than any current system, and may also be more robust, in that subtle patterns can be used by such a system that would be very difficult to even attempt to represent in an object description.

The fact that what is being offered to the system is a set of subtle features spread across noisy images points to the potential for using a more decentralized approach that can consider recognition across the image as a whole, as opposed to representing specific features. The natural choice for such a task is a neural-net based approach: the robustness this approach in the face of noisy and uncertain data is well-known (Mitchell, 1997).

Neural nets have been used extensively for image processing tasks in the presence of noise, from close-range applications such as face recognition (Mohamed, 2002) to remote sensing applications such as oil spill detection or land classification (Kubat et al., 1998; Schaale & Furrer, 1995). It is important to consider such prior work in the context of applications such as that described in this chapter. Recognizing a face, detecting an oil spill, or classifying vegetation from single image, for example, is a much simpler recognition problem than dealing with the subtler issues of ongoing tracking over time that have been presented in the previous sections. Neural networks have also been applied, though less extensively, to tracking information over time (e.g. (Cote & Tatnall, 1997)), and this also supports their use as a viable choice in real-time robot tracking.

At this point in time, there are no systems that can recognize and track moving objects in real time (i.e. in the same fashion as Doraemon and Ergo) adaptively and with no prior knowledge. In working toward this goal, however, we have been working with a subset of the general object recognition problem – recognizing orientation alone, using only images of robots as opposed to pre-defined markings – as a means to gauge the applicability of neural nets to this particular task and as a foothold for more general future work (Baltes & Anderson, 2003b).

Rather than using an artificial set of high-resolution images to train a network, we used actual data that was obtained by Doraemon. The original Doraemon was modified (as a preliminary test for larger-scale motion detection in the development of Ergo) to examine the difference between frames in order to note likely locations for the coloured patterns the system tracks. Where a strong difference is noted, the sub-area of the image (64 x 32 pixels) is stored and the direction of change noted as a basis for the matching described in Section 2. These sub-images can be viewed on the interface in an enlarged fashion for colour calibration purposes, but in this case, this internal portion of the system serves to be able to gather a set of close-up images

of robots over time. A set of training data annotated with estimated orientation can be thus be recorded if it can be assumed that the robot is always facing forward when moving. Examples of these annotated images from training datasets are shown in Fig. 10. Note that lines on the field, as well as the fact that the images are quite small, both serve to promote noise. A training set of 200 images of each of two robot types was created (each network was ultimately trained to recognize orientation in only one type of robot).

Fig 10. Sample images taken and annotated by Doraemon. The left image is a remote controlled toy car; the right is a robot built from a Lego MindStorms kit.

The network employed to track robot orientations without patterns is a 3-layered feed-forward backpropagation network, as shown in Fig. 11. Since the robot-sized images from Doraemon will be ultimately used as input for visual recognition of orientation, the input layer must ultimately receive these images. However, an RGB image of 64 x 32 pixels results in 6144 individual colour-separated pixels. A neural net constructed with this many input nodes was attempted, but performance was found to be poor and training extremely slow, necessitating sub-sampling of the original image to allow fewer input nodes. This was done by averaging over each 4 x 4 pixel neighbourhood, resulting in 384 input nodes. The hidden layer is 32 nodes, or approximately 10% of the input layer. The number of hidden nodes was arrived at by experimentation (Baltes & Anderson, 2003b): using a learning rate of 0.3 and a momentum term of 0.2, a network with a 32 node hidden layer was the first that could learn 100% of a sample dataset after 2000 epochs (compared to 88% after 5000 epochs for the next best performing network topology, with 24 hidden nodes). The output layer is an encoding of the orientation angle discretized into 5-degree steps, resulting in 72 output nodes. The highest-strength output node is taken as the orientation classification.

For a test data set, we used one robot as a test subject and caused it to drive at random across the same field on which the training data were gathered. This introduced the same lines and other noise that were present in the training images. In addition, we placed stationary robots on the field so that the system was exposed to more varied examples of robot images, and to avoid overfitting.

To evaluate learning performance, we employed mean squared error (MSE), a common measure of error employed with neural nets. Mean squared error is the sum of the squared errors (SSE) over the output units, over the entire set of training examples (i.e. over one epoch), divided by the number of training patterns in the epoch. That is, MSE is the mean error for a given pattern.

One interesting element in object recognition using a sub-symbolic approach such as a neural network is the relative utility of the designer attempting to emphasize or likely useful information in images beforehand, as opposed to simply allowing the approach to operate completely unbiased. Since the edges in an image of a robot contain much information that is useful for orientation classification, we attempted to contrast the recognition process using the images already described, with one employing sub-sampled images that had 2 x 2 Sobel edge

detection performed on them. Fig. 12 illustrates the edge maps created by performing edge detection on the images in Fig. 10. Since edge detection also removes colour information from the original image, fewer input nodes were needed in this case.

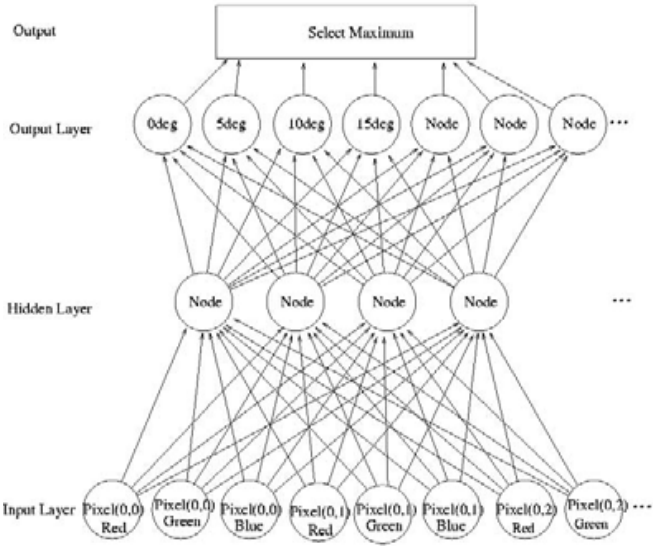


Fig. 11. The Neural Network Architecture.

Fig. 12. Training images from Fig. 10 after 2 x 2 Sobel edge detection.

We ran a comparison on this network once the optimal number of hidden units was decided upon, comparing the accuracy and speed of learning using the toy car training data under three different input representations: 4 x 4 sub-sampled colour images described above, 2 x 2 sub-sampled edge-detected images, and 2 x 2 sub-sampled grey scale images. The third representation was chosen in order to see the effect of removing colour information alone. As in preliminary experimentation, a learning rate of 0.3 and a momentum term of 0.2 were used. In all cases, training data was randomly reshuffled after each epoch to avoid over-fitting the neural network to the specific sequence of training images.
The results of this (Fig. 13) showed that the representation made little difference. Although edge detected images allowed faster MSE improvement over the first few hundred epochs ultimate performance was worse than other representations. The differences are not statistically significant, however, and all finish with an MSE of approximately 0.10.

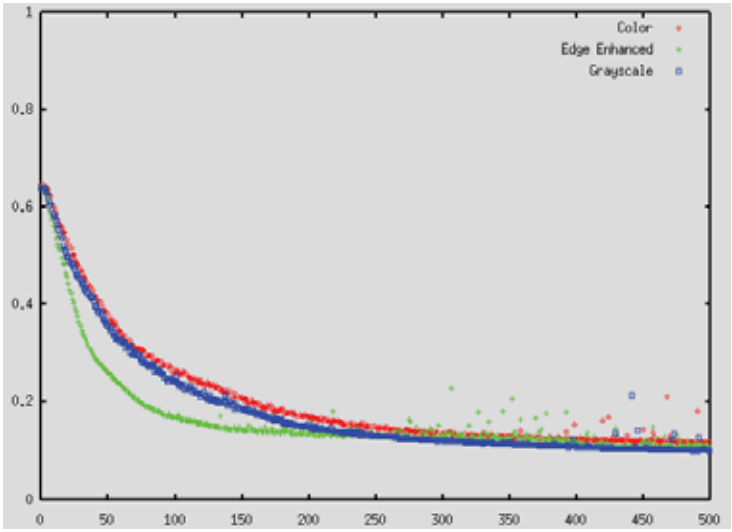


Fig. 13. Evolution of the MSE over training epochs under different input representations, using the toy car image training set.

The neural networks were then tested on accuracy of orientation classification. The results were also similar by representation. The network trained using colour images was able to classify 99% of all images within 5 degrees, while grey-scale or edge map trained networks classified 97% of all images correctly. There was no correlation between missed images over representations: that is, different images were misclassified by the three different representations.

Similar testing was done with Lego MindStorms robots, which as can be seen in Fig. 10, have more distinct visual features. This lead to networks trained with colour and grey-scale images to finish training much earlier than edge-detected images (Fig. 14). These two alternatives both terminated training early with 100% accuracy, while the network trained with edge-detected images still had only 93% accuracy after 5000 epochs.

These results seem to indicate that preprocessing and basic feature selection is not useful for a neural network, and may in fact decrease the performance. While this seems counter-intuitive, in retrospect the training images themselves seem to indicate that orientation is often about noting small pieces of evidence being combined into a consistent view, as opposed to extracting features such as edges. Edge-detection, while emphasizing some elements, loses many small details, especially with subjects such as the Lego robots, where much detail is present. Each pixel provides a small amount of information, but its relationship to other pixels makes this information important, and this relationship is diluted by preprocessing such as edge detection. This was fairly obvious in some cases: some images had shadows along one side, where this shadow was incorporated into the robot shape via edge detection, for example. Artificial neural networks, on the other hand, are especially well-suited to combining large amounts of input into a consistent view to deal with elements such as shadows.

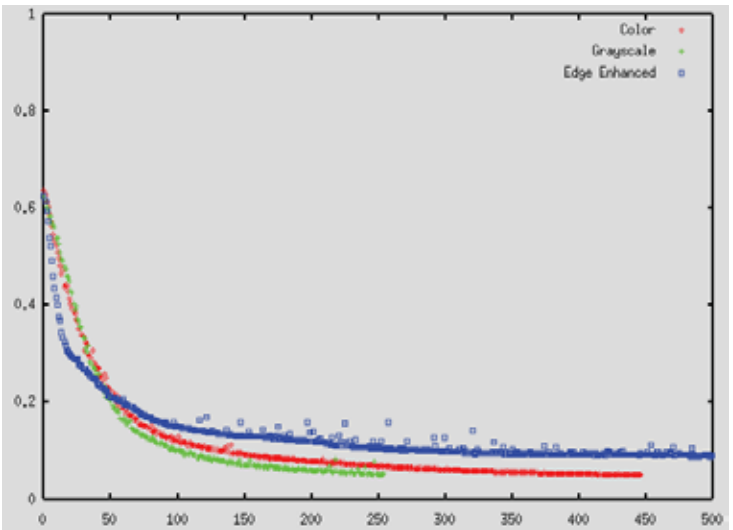


Fig. 14. Evolution of the MSE over training epochs under different input representations, using the Lego robot image training set.

We also examined the ability of these networks to generalize, by extracting 20 images (10%) at random from the set of 200 training images. After training the network on the training set minus these 20 images, the 20 unseen images were tested. These results were not as encouraging: the network as trained was unable to generalize well. Further investigation showed that the generalization ability was limited because there were few training examples for 5-degree turns compared to larger values.

These efforts focus on only one sub-problem of object recognition: orientation. The results could certainly be used in a production vision system, but this is still far from dealing with the larger identification and tracking problem. However, the results presented here do show that artificial neural networks are a promising approach to this problem.

One issue that will require significant work is that of training time. The work here was performed on a dual 1900+ MP Athlon system with 1 GB of RAM, and a training run took approximately 30 minutes. It is certainly conceivable to allow 30 minutes of observation before a system is used in some applications, but this would be unacceptable in others. Current neural network technology requires significant training time, and being able to classify images with very little training will ultimately require significant breakthroughs in many areas outside of computer vision. Another concern is the computational requirements of the neural network after training. Once trained, this same system could process a classification in around 0.07 msec, however, which would be fast enough to apply to a 5-on-5 robotic soccer game. Scaling this to larger teams would require additional resources or significant improvements in efficiency. One possibility to improve speed would be to extrapolate a small set of rules that approximate some of the knowledge in the neural net. Applying these in conjunction with a simpler network may be faster than calculating the output of the entire network employed here.

5. Conclusion

This chapter has reviewed some of the issues involved in creating pragmatic global vision systems. We have discussed the assumptions on which traditional systems are based, pointed out how these differ with the observed abilities of human vision, and described how these assumptions limit the applicability and generality of existing systems. We then described techniques that allow some of these assumptions to be discarded, and the embodiment of these techniques in our production global vision systems, Doraemon and Ergo.

Both Doraemon and Ergo are used in a number of ways. Doraemon has been in use every year by a number of teams from around the world in the F-180 (small-size) league at RoboCup. Ergo is the current global vision system in use in our own laboratories, and is currently being employed in a number of projects, such as imitation learning in groups of robots (Allen, 2007).

We have also described some of our recent work toward creating much more general global vision systems that take advantage of additional knowledge or adaptability in order to avoid the need for any type of predefined markings on objects. The latter work is very preliminary, but shows the potential for improved techniques to eventually be the basis for more general vision systems.

In working toward such generality today, there are a number of very important areas of immediate future work. Existing approaches to global vision are well-understood and immediately deployable. The fact that they rely heavily on elements such as the ability to recognize colour patches, for example, means that anything that can be done to improve these abilities will serve to improve existing systems. While systems such as Doraemon are already exploiting much in terms of maximizing flexibility while still assuming colours can be defined and matched, future work may still improve this further.

Any small steps that can be performed to wean existing systems away from their traditional assumptions will serve as a backbone for further future work. While Ergo is a significant improvement over the abilities of Doraemon, for example, it still conforms to some traditional assumptions in terms of relying on predefined patterns, and instead exploits different mechanisms to be more flexible and offer a better performance in a wider array of domains. There will be many similar steps as we move to more general vision systems.

Any single tracking or identification technique has its limitations, and just as neither Ergo nor Doraemon use a single mechanism to identify and track objects, future systems will require a synergy of techniques. Attempting to leverage the strengths of techniques off of one another will always be an important part of future work in this area. In our own work, we are currently attempting to employ the addition of control knowledge to the sub-symbolic orientation recognition described in Section 4.2. For example, if we are uncertain of a robot's location and orientation at the current time, we can start with the robot's last known location/orientation at previous time, and constrain the potential solution set by the likely outcome of the most recent command sent to the robot.

The iterative steps taken in improving global vision are in turn a useful source of future work in improving application areas as well. For example, the work on recognizing orientation without markers described in Section 4.2 was undertaken as convenient sub-problem of the overall vision task useful in robotic soccer, in order to track a team's own players for control purposes. The ability to infer robots' orientation without prior

knowledge, however, also allows a team to infer the orientation and identity of the opponent's robots. This in turn can allow for more sophisticated tactical decision making than would otherwise be possible. For example, robots that have a strong kicking device can be extremely dangerous. If an opponent's robot is oriented away from the current locus of activity, the situation is not as dangerous.

In both current and ongoing work, there is also a great need for improvements to computational efficiency. While computer power is always improving, the demands of more sophisticated techniques will always exceed this. While we have attempted in Ergo, for example, to have as much of the matching be done in a goal-directed fashion, data-directed processing is still required, and so there is still ample opportunity for improving the frame-rate in ergo through improvements in pattern-matching efficiency. In using control information to anticipate future movement, techniques that do not require the calculation of all possible robot assignments to all traces would be an enormous improvement.

Finally, it should be noted that despite the fact that we have emphasized global vision in this chapter, the techniques employed in object tracking and identification by Doraemon, Ergo, and the other work described here are all equally applicable to local vision. If I have a local vision robot playing a soccer game, the robot still must be able to track its teammates and opponents across its field of vision, and update an internal model of the state of play in order to make intelligent decisions. Thus advancement in technology in one area is immediately applicable to the other. Although it does not compare to the limitations of human vision, omni-vision (that is, vision based on a 360° image, usually done with a camera and a parabolic mirror) has become largely a standard in some local vision robotic soccer leagues, most notably the RoboCup middle-sized league. Such vision ultimately allows a reconstruction on a local basis that bears a strong analogy to global vision, especially once a camera is not placed overhead and issues such as occlusion and complex geometry come into play.

If readers are interested in using the work described here in their own future work, open-source code for Doraemon, Ergo, and other systems is available (Baltes & Anderson, 2006).

6. Acknowledgement

The authors would like to thank all of the students in our laboratories in Manitoba and Auckland that have worked on global vision software in the past, including Paul Furgale and Ben Vosseteig, as well as all of the students in our robotics courses that stress-test this software and continue to drive ongoing development.

This work was funded in part by Natural Sciences and Engineering Research Council of Canada (2002 onward), and the University of Auckland (prior to 2002).

7. References

Allen, J. (2007). *Imitation learning from multiple demonstrators using global vision.* Master's thesis, Department of Computer Science, University of Manitoba, Winnipeg, Canada (forthcoming).

Anderson, J. & Baltes, J. (2002). Doraemon user's manual. *http://robocup-video.sourceforge.net*, 2002.

Ball, D., Wyeth, G., & Nuske, S. (2004). A global vision system for a robot soccer team. *Proceedings of the 2004 Australasian Conference on Robotics and Automation (ACRA),* Canberra, Australia, December 2004.

Baltes, J. (2002). Doraemon: Object orientation and ID without additional markers. *Proceedings of the 2nd IFAC Conference on Mechatronic Systems,* pp. 845-850, Berkeley, CA, December, 2002. American Automatic Control Council.

Baltes, J., & Anderson, J. (2003a). Identifying robots through behavioral analysis. *Proceedings of the Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems (CIRAS)*, Singapore, December, 2003.

Baltes, J., & Anderson, J. (2003b). Learning orientation information using neural nets. *Proceedings of the 2003 FIRA Congress*, Vienna, Austria, October, 2003.

Baltes, J., & Anderson, J. (2005). Interpolation methods for global vision systems. In: Nardi, D., Riedmiller, M., & Sammut, C., (Eds.), *The Seventh RoboCup Competitions and Conferences*, pp. 434-442, Springer-Verlag, Berlin.

Baltes, J., & Anderson, J. (2006). Doraemon, Ergo, and related global vision systems. *http://robocup-video.sourceforge.net*.

Browning, B., Bowling, M., Bruce, J., Balasubramanian, R., & Veloso, M. (2002). CM-Dragons01: vision-based motion tracking and heterogenous robots. In: Birk, A., Coradeschi, S., & Tadokoro, S., (Eds.) *RoboCup-2001: Robot Soccer World Cup* V, pp. 567–570, Springer-Verlag, Berlin.

Bruce, J., Balch, T., & Veloso, M. (2000). Fast and inexpensive colour image segmentation for interactive robots. *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-00)*, volume 3, pp. 2061 – 2066, Takamatsu, Japan, November, 2000.

Bruce, J., & Veloso, M. (2003). Fast and accurate vision-based pattern detection and identification. *Proceedings of Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-03)*, pp. 567–570, Taipei, Taiwan, May, 2003.

Cote, S., & Tatnall, A. (1997). The Hopfield neural network as a tool for feature tracking and recognition from satellite sensor images. *International Journal of Remote Sensing*, Vol. 18, No. 4, 1997, pp. 871–885.

Furgale, P., Anderson, J., & Baltes, J. (2005). Real-time vision-based pattern tracking without predefined colours. *Proceedings of the Third International Conference on Computational Intelligence, Robotics, and Autonomous Systems (CIRAS)*, Singapore, December, 2005.

Kubat, M., Holte, R., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images *Machine Learning*, Vol. 30, No. 2-3, 1998, pp. 195–215.

Manzanera, A., & Richefeu, J. (2004). A robust and computationally efficient motion detection algorithm based on sigmadelta background estimation. *Proceedings of the 4th Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 46-51, Kolkata, India, December, 2004.

Mitchell, T. (1997). *Machine Learning*, McGraw-Hill, Boston.

Mohamed, R. (2002). Face detection in colour images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 5, 2002, pp. 696–706.

Schaale, M., & Furrer, R. (1995). Land surface classification by neural networks. *International Journal of Remote Sensing*, Vol. 16, No. 16, 1995, pp. 3003–3031.

Simon, M., Behnke, S., & Rojas, R. (2001). Robust real time colour tracking. In: Stone, P., Balch, T., & Kraetszchmar, G., (Eds.), *RoboCup-2000: Robot Soccer World Cup IV*, pp. 239–248, Springer-Verlag, Berlin.

Tsai, R (1986). An efficient and accurate camera calibration technique for 3D machine vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 364–374, Miami Beach, FL, June, 1986.

10

Contour Extraction and Compression-Selected Topics

Andrzej Dziech
AGH Universisty of Science and Technology, Telecommunication Dept., Cracow, Poland

The subject of this chapter is to describe selected topics on contour extraction, approximation and compression in spatial domain. Contours are treated as important image structure required in many applications, for example in analysis of medical image, computer vision, robot guidance, pattern recognition etc. Contour compression plays important role in many practical tasks associated with contour and image processing.
The main approach solving the problem of contour compression is based on polygonal approximation. In this chapter the traditional and new methods for contour approximation in spatial domain are presented. These methods are often much faster than the methods of compression based on transform coding. In order to extract the contours from the image some algorithms have been developed. The selected well known methods for contour extraction and edge detection as well as the new algorithms are also presented in this chapter. The author is grateful to his Ph.D. students A. Ukasha and B. Fituri for their contribution in preparation of this chapter.

1. Contour Analysis and Extraction

Contours and line drawings have been an important area in image data processing. In many applications, e.g., weather maps and geometric shapes, it is necessary to store and transmit large amounts of contours and line drawings and process the information by computers.
Several approaches have been used to extract and encode the boundary points of contours and line drawings. The extracted data is then used for further processing and applications. Contour approximation and compression are some of the processing operations performed on contours and has been considered by several authors.
In encoding contours and line drawings, efficient data compression and good reconstruction are both usually required. Freeman proposed an eight-directional encoding scheme for contour lines. The proposed chain code is obtained by superimposing a rectangular grid on the curve and then quantizing the curve to the nearest one of the eight possible grid points. The chain encoding scheme represents contour lines by 3 bits/link, where a link is defined as one of the eight possible straight-line segments between two adjacent quantized points.
Efforts have been made to improve the coding efficiency. Freeman proposed a chain difference coding scheme which assigned variable-length codewords to the difference between two consecutive links. This coding scheme represents contour lines by about 2 to 2.1 bits/link on average.

The purpose of this subchapter is to investigate selected methods for contour extraction. At the beginning some relationship between the image and contours that can be extracted from the image, are briefly described.

In the simplest case, an image may consist of a single object or several separated objects of relatively high intensity. This allows figure/ground separation by thresholding. In order to create the two-valued binary image a simple threshold may be applied so that all the pixels in the image plane are classified into object and background pixels. A binary image function can then be constructed such that pixels above the threshold are foreground ("1") and below the threshold are background ("0").

Binary images are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are used 0 for black and 1 for white. In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and "the rest". This latter group is also referred to as the background. The techniques that are used to find the objects of interest are usually referred to as *segmentation techniques* - segmenting the foreground from background.

In general there are two basic approaches for shape representation: by contours and by regions. Polygonal approximation, chain code, geometric primitives, parametric curves, Fourier descriptors and Hough transform are the examples of contour based shape representation methods. These methods share some common characteristics [1]:

(1) *Shape information extraction:* the representation would facilitate some contour characteristics comprehension.
(2) *Data compression:* data compression rates can vary in wide range depending on the method of compression and the structure of contours
(3) *Noise elimination:* digital curves can be corrupted with noise and/or undesirable details treated as redundancy elements. The method should filter the noise and redundancies.
(4) *Curvature evaluation:* this step is important in contour description. The major difficulty is due to the discrete nature of the curve, making the majority of the methods noisy and scale dependent. There are psychological results showing that curvature plays a fundamental role in human shape perception.

The most typical contour representations are illustrated below [2]:

1) Generalized representation (θ, l).

Fig. 1.1 shows the contour representation using the (θ, l) generalized chain coding scheme.

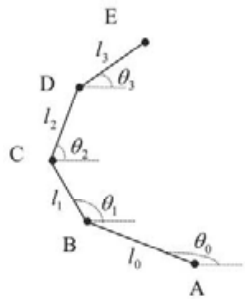


Fig. 1.1 Generalized representation of contour.

2) Polar representation (α, l) .

Fig. 1.2 shows the contour representation using the (α, l) polar chain coding scheme.

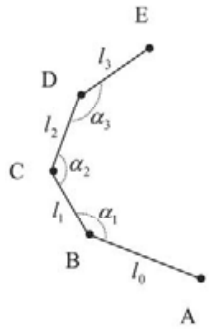


Fig. 1.2 Polar representation.

3) Cartesian representation

Cartesian representation of contour is shown in Fig. 1.3

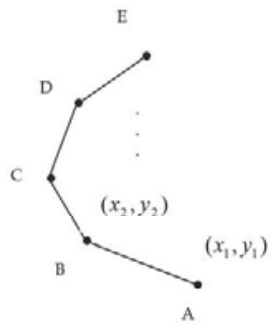


Fig. 1.3 Cartesian representation.

Cartesian representation leads to decomposition of two-dimensional contour (y, x) into two one –dimensional signals x(n) and y(n), where n is a variable representing the current length of contour, as it is shown below

One of the widely used procedures related to contour tracing is proposed by Freeman [3] . This procedure is based on an eight- or four-directional chain encoding scheme as shown in Fig. 1.5. An 8-directional chain-coding uses eight possible directions to represent all possible line segments connecting nearest neighbours according to the 8-connectivity scheme as shown in Fig. 1.5a. 4-directional chain-coding uses four possible directions to represent all possible line segments connecting nearest neighbours according to the 4-connectivity scheme as shown in Fig. 1.5b.

This scheme describes arbitrary geometric configurations in a simple and efficient method. The chain code is obtained by superimposing a rectangular grid on the curve and then quantizing the curve to the nearest grid point. The Freeman chain code may subsequently be described in cartesian or polar systems.

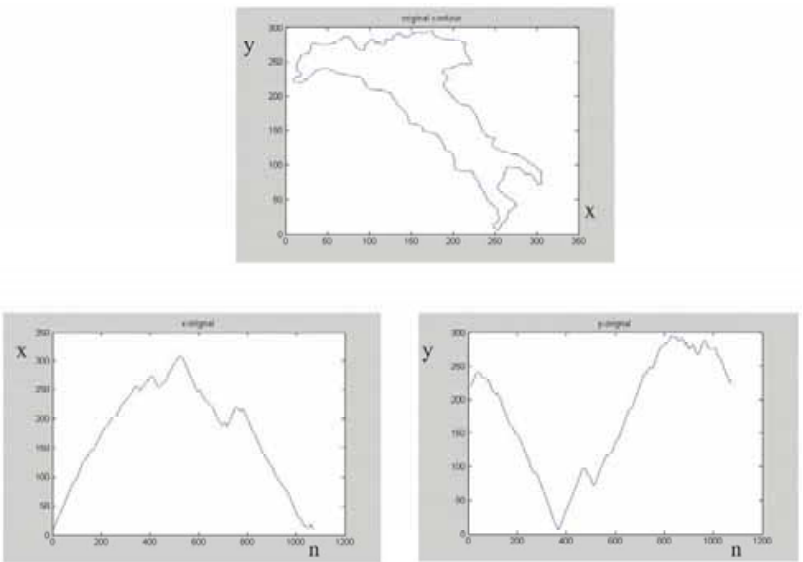
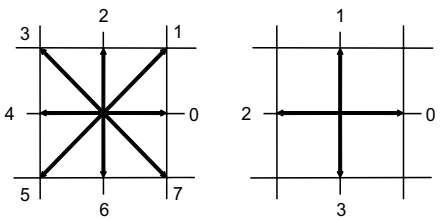


Fig. 1.4 Contour decomposition.



a) 8-directional chain code. b) 4-directional chain code.

Fig. 1.5 Freeman chain code.

Completely enclosed boundary regions can be coded with a simple modification to the basic chain coding. The outer boundary is first chain coded in a normal manner when this boundary has been closed ,a code group 0401 is inserted in the chain code, and an "invisible line" connecting the two boundaries is encoded. When the second boundary is reached, the code group 0402 is inserted in the chain code to indicate the end of the invisible line. The inner boundary is then chain coded in a normal manner. The prefix 04 of the "invisible line" code and the "visible line" code designates the rarely occurring event of a right shift followed by a left shift. This prefix is also used with other codes to indicate a variety of special cases.

-Length of a chain: The length of an 8-directional chain code may be found using the following relation:

$$L = T(n_e + n_o\sqrt{2})$$

where n_e -number of even valued elements (chains)

n_o --number of odd valued elements (chains)

T- scale factor proportional to grid spacing.

-Inverse chain: The inverse chain of an 8-directional chain code may be obtained using the following relation:

$$c_i^{-1} = c_i \oplus 4$$

where $\oplus$ -addition mod 8

Example:

-For the curve shown

(a) write the Freeman chain code using the 8-directional scheme.

(b) Find the length of the chain code.

(c) Find the inverse of the chain code.

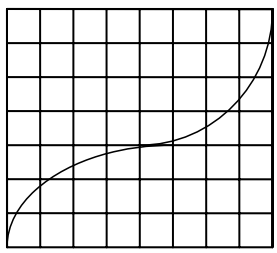
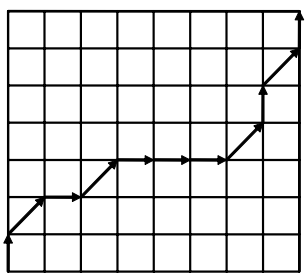


Fig. 1.6 Chain encoding example

(a) In the Figure below is shown the tracing of the curve using the 8-dierectional scheme.



The chain code is obtained as: (21010001212)

(b) The length of the chain code, assuming T=1, is :

$$L = T(n_e + n_o\sqrt{2}) = 1(7+4\sqrt{2}) = 12.657$$

(c) The inverse of the code is: . $c_i^{-1} = c_i \oplus 4$ =(65454445656)

1.1 OCF and MSCE Methods of Contour Extraction

The separation of objects from their background referred to as segmentation of gray scale images and as contour tracing (or boundary following) for binary images, often provide important features in pattern recognition and scene analysis and is used in variety of other applications, for example, recognition of human faces from essential contours.

In general contour extraction from two-dimensional images may be accomplished in two operations: 1)*Edge detection*; 2)*Contour tracing*.

1) *Edge detection*: The aim of edge detection is to identify and enhance edges (pixels) belonging to boundaries of object of interest in the original image. An edge element is defined as a picture element which lies on the boundary between objects or regions of different intensities or gray levels. Many edge detection methods have been proposed for detecting and enhancing edges in digital images. Most of these methods are implemented as some form of gradient operators. Some images can be characterized as containing some objects of interest of reasonably uniform brightness placed against a background of differing brightness. Typical examples include handwritten and typewritten text, and airplanes on the a runway. For such images, brightness is a distinguishing feature that can be utilized to locate the object. This method is termed as luminance thresholding.

2) *Contour tracing*: The contour tracing algorithm traces the contour and extracts the contour information which is then passed to subsequent processing. One of the most widely used procedures to follow contours and line drawings is that of Freeman, which provides a code that possesses some manipulative properties. The Freeman chain code can subsequently be described in cartesian or polar systems.

The problem of contour extraction from 2D-digital image has been studied by many researchers, and a large number of contour extraction methods have been developed. Most of the developed methods can be assigned to either of two major classes known as sequential methods or Object Contour Following (OCF), and parallel methods or Multiple Step Contour Extraction (MSCE). In Fig.1.6 is shown a block diagram of the contour extraction and processing from gray level images.

A brief description of the two main classes of contour extraction methods, OCF and MSCE, is given.

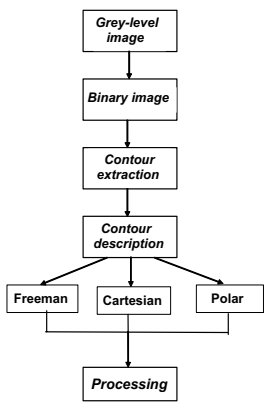


Fig. 1.6 Contour extraction and processing from gray level images.

Two main approaches can be used for the extraction of object contours. The Object Contour Following (OCF) and Multiple Step Contour Extraction (MSCE). The OCF methods, sequentially detect and extract object contour edges. By sequential methods, it is meant that the result at a point is dependent upon the result of the previously processed points. The MSCE methods are referred to as parallel schemes for object contour extraction. By parallel, it is meant that the decision of whether or not a point is on an edge is made on the basis of the gray level of the point and its neighbours. So, the edge detection operator in principle can be applied simultaneously everywhere in the picture. It should be noted that the definitions of sequential and parallel schemes are used with respect to edge detection. To produce a closed boundary, the extracted edges have to be connected together to form a closed curve.

(i) Object Contour Following(OCF)

The OCF methods, which are also called Bug Following, can be used to trace (follow) the contour edges of a 2-D digital image. The idea of these methods is illustrated in Fig.1.7. The extraction procedure consists of finding a starting point and then cross the edge between the white and black regions, record the co-ordinates of the black pixel then turn left continuously until a white pixel is found, record the black pixel co-ordinates as the next contour edge point. Start turning right until a black pixel is found. Terminate this procedure when the starting point of the contour is reached again.

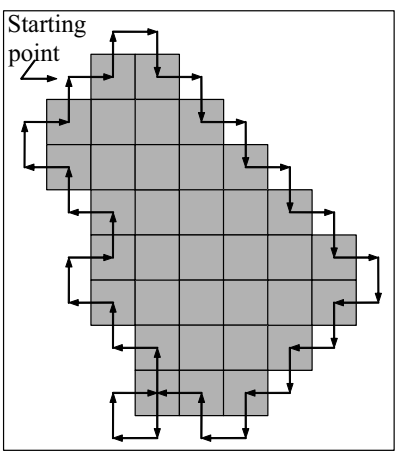


Fig. 1.7 The Object Contour Follower.

(ii) Multiple Step Contour Extraction (MSCE)

The key feature of MSCE methods is that the gradient between two pixels with different gray scale levels represents the difference between the two pixels, and the gradient will be zero for the pixels with the same gray scale level. A threshold value will determine if the gradient is interpreted as an object edge or not. An additional procedure is used to improve the overall contour structure by joining all disjoined edges and thinning the thick edges.

Fig.1.8 shows the steps required for extracting object contours by the MSCE methods. Although the method of gradient operator for generating edge elements is parallel, the method of connecting (tracing) these extracted edge elements is sequential.

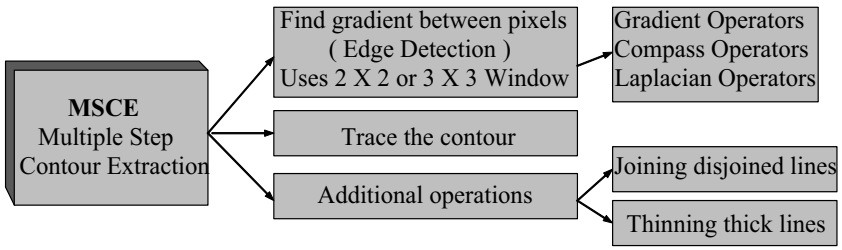


Fig. 1.8 Block Diagram of The Multiple Step Contour Extraction.

The three main steps of the MSCE methods are: edge detection, contour tracing and the additional procedures for joining disjoined lines, and thinning thick lines.

a) Edge Detection

Local discontinuities in image luminance or gray levels are called luminance edges. Global discontinuities are called boundary segments. Edges characterise object boundaries. They are used for segmentation and identification of objects in images. Edge points can be thought of as pixel locations with abrupt gray level change. For example, it is reasonable to define edge points in binary images as black pixels (object pixels) with at least one white nearest neighbour pixel (background pixel). Most techniques used for edge detection are limited to processing over the 2x2 or 3x3 windows shown in Fig. 1.9a and Fig. 1.9b respectively.

Note that the same pixel numbering will be used with all edge detection operators.

	i	i+1
j	P(i,j)	P(i+1,j)
j+1	P(i,j+1)	P(i+1,j+1)

Fig. 1.9a Pixel numbering for 2x2 edge detecting operators.

		i	
	P_0	P_1	P_2
j	P_7	P(i,j)	P_3
	P_6	P_5	P_4

Fig. 1.9b Pixel numbering for 3x3 edge detecting operators.

b) Gradient Operators:

These operators measure the gradient of the image G(i, j) in two orthogonal directions, horizontal, and vertical directions. Except with the case of Roberts cross operator where the gradient is taken across the diagonals.

The gradient is given by :

$$G(i,j) = \sqrt{g_v^2(i,j) + g_h^2(i,j)} \tag{1.1}$$

where g_v : is the gradient in the vertical direction.
g_h : is the gradient in the horizontal direction.
Instead of using the square root gradient given by Eq.(1.1), the gradient is often approximated using the absolute gradient given by the following equation :

$$G(i, j) = \left| g_v(i, j) \right| + \left| g_h(i, j) \right| \tag{1.2}$$

Eq.(1.2) is easier to perform and to implement in digital hardware.
The Roberts , Sobel , Prewitt [26] operators, showing the horizontal and vertical masks (diagonal masks in case of Roberts cross) together with the necessary equations for finding the gradient, are introduced next, as an example of edge detection using gradient operators.

Roberts cross gradient operator :
Roberts has used a simple window of 2x2 to introduce the square-root difference operator given in Fig.1.10, it is often called Roberts cross gradient operator. The edge detection is carried out using Eq.(1.1), where :

$$g_v = P(i, j) - P(i+1, j+1) \tag{1.3}$$
$$g_h = P(i, j+1) - P(i+1, j) \tag{1.4}$$

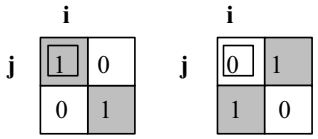
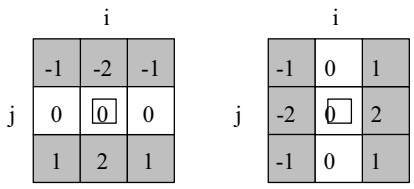


Fig. 1.10 Roberts cross gradient operator.

Sobel gradient operator :
The masks which are used by Sobel for finding the gradient of an image are shown in Fig. 1.11. The corresponding equation used for calculating the gradient is given by Eq.(1.1), where :

$$g_v = (P_2 + 2P_3 + P_4) - (P_0 + 2P_7 + P_6) \tag{1.5}$$
$$g_h = (P_0 + 2P_1 + P_2) - (P_6 + 2P_5 + P_4) \tag{1.6}$$



Horizontal mask. Vertical mask.

Fig. 1.11 Sobel gradient operator.

Laplacian operators :
Three different Laplacian operators [26][27], with the necessary equations for calculating the gradient, are shown in Fig.1.12. For images with smooth changes in gray level values the

Laplacian operators give better results than the previous operators. But it is more sensitive to noise, and produces double edges.

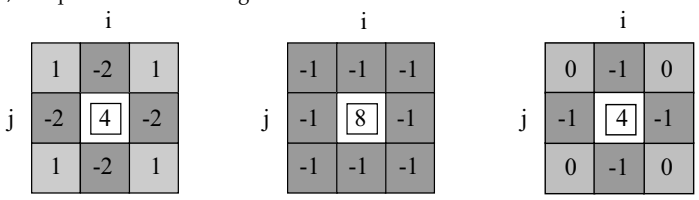


Fig. 1.12 Three different types of Laplacian operators.

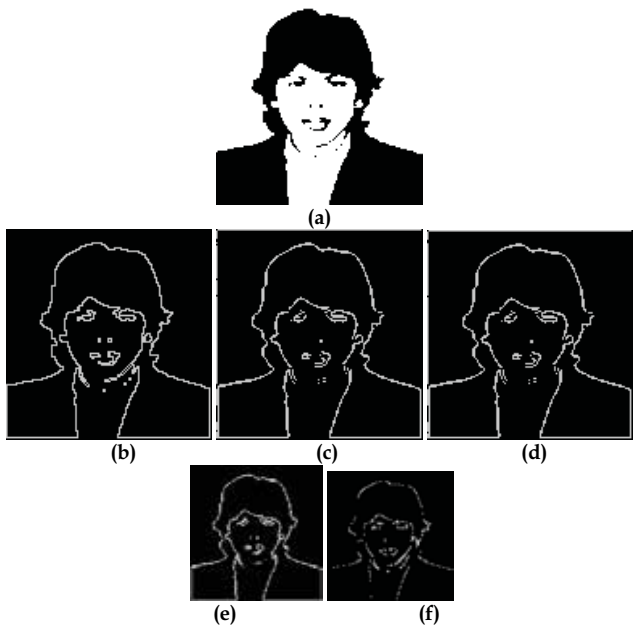
For the operator of Fig. 1.12a -c the edges are detected by calculating the gradients between pixels using the following formulas respectively :

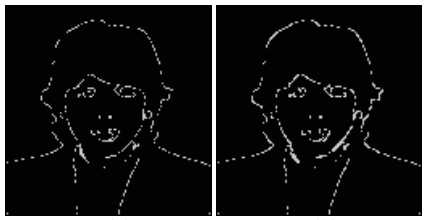
$$G(i,j) = 4F(i,j) + (P_0 + P_2 + P_4 + P_6) - (P_1 + P_3 + P_5 + P_7) \quad (1.7)$$

$$G(i,j) = 8F(i,j) - (P_0 + P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7) \quad (1.8)$$

$$G(i,j) = 4F(i,j) - (P_1 + P_3 + P_5 + P_7) \quad (1.9)$$

To compare the performance of the selected gradient operators, the binary image of Fig. 1.13a is used. The detected edges obtained by applying different gradient operators are shown in Fig. 1.13b-h.





(g) **(h)**

(a) Original image, (b) Prewitt gradient Operator, (c) Roberts cross gradient operator, (d) Sobel gradient operator, (e) Prewitt compass operator, (f - h) Three types of Laplacian operators

Fig. 1.13 Different edge detection results on a real binary image.

1.2 Object-oriented Contour Extraction OCE

This algorithm is based on 4x4 pixels window structure to extract the object contours by the four central pixels which are processed simultaneously. The algorithm uses the features of both OCF and MSCE methods to overcome most of the disadvantages they have.

It features a parallel implement and an effective suppression of noises. It can be realized in real-time [18].

The following three steps are needed for the extraction procedure:

Step1: The image is framed with zeros.

Step2: Eight rules of edge extraction are applied and are coded using 8-directional chain-code as shown in Listing 1.1.

Listing 1.1

Implementation of the eight rules for contour extraction (4x4 windows)

```
a(i,j) ← 0; i = 1,2,…..,N; j = 1,2,…..,N;
for i = 2,3,…..,N-1; j = 2,3,…..,N-1;
{
if b(i,j+1) and b(i+1,j+1) and [b(i,j+2) or b(i+1,j+2)]
then a(i,j+1) ← a(i,j+1) or 2^0                    { edge 0 }
if b(i+1,j) and b(i,j+1) and b(i+1,j+1)
then a(i,j+1) ← a(i,j+1) or 2^1                    { edge 1 }
if b(i+1,j) and b(i+1,j+1) and [b(i+2,j) or b(i+2,j+1)]
then a(i+1,j+1) ← a(i+1,j+1) or 2^2                { edge 2 }
if b(i,j) and b(i+1,j) and b(i+1,j+1)
then a(i+1,j+1) ← a(i+1,j+1) or 2^3                { edge 3 }
if b(i,j) and b(i+1,j) and [b(i,j-1) or b(i+1,j-1)]
then a(i+1,j) ← a(i+1,j) or 2^4                    { edge 4 }
if b(i,j) and b(i+1,j) and b(i,j+1)
then a(i+1,j) ← a(i+1,j) or 2^5                    { edge 5 }
if b(i,j) and b(i,j+1) and [b(i-1,j) or b(i-1,j+1)]
then a(i,j) ← a(i,j) or 2^6                        { edge 6 }
if b(i,j) and b(i,j+1) and b(i+1,j+1)
then a(i,j) ← a(i,j) or 2^7                        { edge 7 }
}
```

where:

b(i,j) is the binary value of a pixel point (i,j) and 2^k (*k*:0-7) is the extracted edge code.

Step3: The extracted contour edges are sorted and stored or optimized according to the application requirements. The extraction procedure is shown in Fig. 1.14.

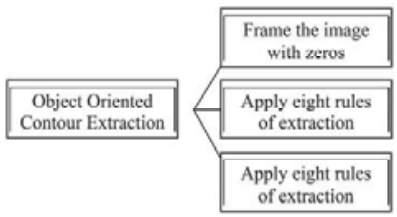


Fig. 1.14 Object-oriented Contour Extraction (OCE).

The problem in the OCE procedure is that contours extracted from the objects near the image boundary, i.e. objects within one pixel distance from the image border, are not closed and that is why the image should be framed with two background pixels to ensure the closure of the contours. Fig. (1.15a) shows that the extracted edges do not form closed contours; while Fig. (1.15b) shows that after framing the image with at least two underground pixels all extracted contours are closed.

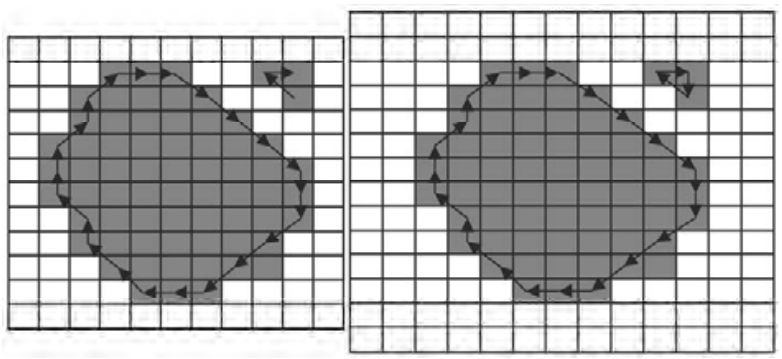


Fig. 1.15 OCE procedure (a) Without correcting the first step and (b) After correcting the first step.

1.3 Single Step Parallel Contour Extraction 'SSPCE' (3x3 windows)

There are two algorithms; the first one [35] [35] uses 8-connectivity scheme between pixels, and 8-Directional Freeman chain coding [3] scheme is used to distinguish all eight possible line segments connecting nearest neighbors. This algorithm uses the same principle of extraction rules as in the OCE algorithm. The second algorithm [35] uses the 4-connectivity scheme between pixels, and 4-Directional Freeman chain coding scheme is used to distinguish all four possible line segments. Both algorithms use an 3x3 pixels window structure to extract the object contours by using the central pixel to find the possible edge direction which connect the central pixel with one of the remaining pixels surrounding it.

The first algorithm gives exactly the same extracted contours as the OCE algorithms but is much faster ; while the second algorithm gives similar contours, but not identical and is also faster . Consider now the first algorithm in details.

The edges can be extracted by applying the definition that an object contour edge is a straight line connecting two neighboring pixels which have both a common neighboring object pixel and a common neighboring underground pixel [33]. By this definition, no edges can be extracted from the three following cases:

1- If all nine pixels are object pixels; i.e. the window is inside an object region.
2- If all nine pixels are background pixels; i.e. the window is inside a background region.

If the center pixel is an object pixel surrounded by background pixels; i.e. it is most probable that the center pixel in this case is a point noise caused by image digitalization.

So, this algorithm uses the same principle and steps of extraction rules as the OCE algorithm using 3x3 window. The eight rules of edge extraction are applied and are coded using 8-directional chain-code as shown in Listing 1.2.

Listing 1.2 (3x3 windows)
Implementation of the eight rules for contour extraction (3x3 windows)

```
a(i,j) ← 0; i = 1,2,.....,N; j = 1,2,.....,N;
for i = 2,3,.....,N-1; j = 2,3,.....,N-1;
{
if b(i,j) and b(i+1,j) and [b(i,j+1) or b(i+1,j+1)] and [ not [b(i,j-1) or b(i+1,j-1)]]
then a(i,j) ← a(i,j) or 2^0                                   { edge 0 }
if b(i,j) and b(i+1,j) and b(i+1,j-1) and [ not [b(i,j-1)]]
then a(i,j) ← a(i,j) or 2^1                                   { edge 1 }
if b(i,j) and b(i,j-1) and [b(i+1,j) or b(i+1,j-1)] and [ not [b(i-1,j) or b(i-1,j-1)]]
then a(i,j) ← a(i,j) or 2^2                                   { edge 2 }
if b(i,j) and b(i,j-1) and b(i-1,j-1) and [ not [b(i-1,j)]]
then a(i,j) ← a(i,j) or 2^3                                   { edge 3 }
if b(i,j) and b(i-1,j) and [b(i,j-1) or b(i-1,j-1)] and [ not [b(i,j+1) or b(i-1,j+1)]]
then a(i,j) ← a(i,j) or 2^4                                   { edge 4 }
if b(i,j) and b(i-1,j) and b(i-1,j+1) and [ not [b(i,j+1)]]
then a(i,j) ← a(i,j) or 2^5                                   { edge 5 }
if b(i,j) and b(i,j+1) and [b(i-1,j) or b(i-1,j+1)] and [ not [b(i+1,j) or b(i+1,j+1)]]
then a(i,j) ← a(i,j) or 2^6                                   { edge 6 }
if b(i,j) and b(i,j+1) and b(i+1,j+1) and [ not [b(i+1,j)]]
then a(i,j) ← a(i,j) or 2^7                                   { edge 7 }
}
```

1.4 Contour Extraction Based on 2x2 Windows

This algorithm is mainly used for gray scale images . It uses a smaller window for contour extraction than its predecessors, i.e. 2x2 window shown in Fig. 1.16.

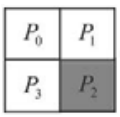


Fig. 1.16 Pixel numbering for 2x2 windows.

The processed pixel is the darker one. Two buffers are required for a real time contour extraction system. First buffer is used for the storage of a previously processed image line and the second one keeps pixel values of the currently processed image line.
The algorithm uses the 8-connectivity scheme, and the extracted edges are coded by using of the 8-directional chain coding. It does not require any storage of the scanned (processed) image.
The three main steps of the algorithm are:

- Frame the image with zeros,
- Extract contour edges using the eight rules
- Sort the extracted contour edges.

The eight rules of edge extraction are applied and are coded using 8-directional chain-code as shown in Listing 1.3.

Listing 1.3
Implementation of the eight rules for contour extraction (2x2 windows)

```
a(i,j) ← 0; I = 1,2,.....,N; j = 1,2,.....,N;
for i = 2,3,.....,N-1; j = 2,3,.....,N-1;
{
if ( b(i-1,j) = b(i,j) ) ⋂ ( b(i-1,j) ≠ b(i-1,j-1) ) ⋂ (b(i-1,j) ≠ b(i,j-1) )
then a(i-1,j) ← a(i-1,j) ⋃ b(i-1,j) ⋃ 2^0 { edge 0 }
if ( b(i-1,j) = b(i,j-1) ) ⋂ ( b(i-1,j) ≠ b(i-1,j-1) )
then a(i-1,j) ← a(i-1,j) ⋃ b(i-1,j) ⋃ 2^1 { edge 1 }
if ( b(i,j-1) = b(i,j) ) ⋂ ( b(i,j) ≠ b(i-1,j) ) ⋂ (b(i,j) ≠ b(i-1,j-1) )
then a(i,j) ← a(i,j) ⋃ b(i,j) ⋃ 2^2 { edge 2 }
if ( b(i-1,j-1) = b(i,j) ) ⋂ ( b(i,j) ≠ b(i-1,j) )
then a(i,j) ← a(i,j) ⋃ b(i,j) ⋃ 2^3 { edge 3 }
if ( b(i,j-1) = b(i-1,j-1 ) ) ⋂ (b(i,j-1) ≠ b(i,j) ) ⋂ (b(i,j-1) ≠ b(i-1,j) )
then a(i,j-1) ← a(i,j-1) ⋃ b(i,j-1) ⋃ 2^4 { edge 4 }
if ( b(i,j-1) = b(i-1,j) ) ⋂ ( b(i,j-1) ≠ b(i,j) )
then a(i,j-1) ← a(i,j-1) ⋃ b(i,j-1) ⋃ 2^5 { edge 5 }
if ( b(i-1,j-1) = b(i-1,j) ) ∩ ( b(i-1,j-1) ≠ b(i,j-1) ) ∩ (b(i-1,j-1) ≠ b(i,j))
then a(i-1,j-1) ← a(i-1,j-1) ∪ b(i-1,j-1) ∪ 2^6 { edge 6 }
if ( b(i-1,j-1) = b(i,j) ) ∩ ( b(i-1,j-1) ≠ b(i,j-1) )
then a(i-1,j-1) ← a(i-1,j-1) ∪ b(i-1,j-1) ∪ 2^7 { edge 7}
}
```

The algorithm does not require the storage of the scanned image, i.e. it can be used for real time applications.

1.5 Comparison of Contour Extraction Algorithms (Different Windows)

The comparison is made between the following three algorithms:

- Contour extraction CE referred to as the third algorithm (or 2x2 windows).
- SSPCE method; it will be referred to as the second algorithm (or 3x3 windows).
- OCE method; it will be referred to as the third algorithm (or 4x4 windows).

The comparison is performed with respect to the number of operation and number of contour edges. The binary test images are illustrated in Fig. 1.17.

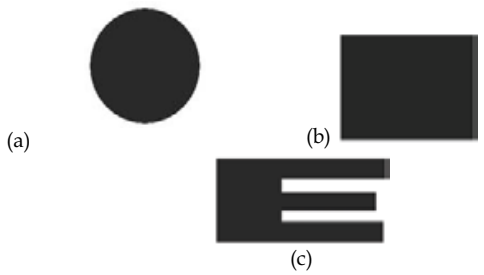


Fig. 1.17 Binary images (a) Circle (b) Square (c) E letter.

The comparison between the three algorithms with respect to the number of operations versus the number of edges for Circle, Square and E letter contours respectively is shown in Tab. 1.1, Tab. 1.2 and Tab. 1.3

NE	20	30	40	50	60	**AE**
1^{st} algor. (*NO*)	*276497*	*276581*	*276666*	*276748*	*276835*	*277060*
2^{nd} algor. (*NO*)	*89810*	*89894*	*89979*	*90061*	*90148*	*90373*
3^{rd} algor. (*NO*)	*1,065018*	*1,065092*	*1,065167*	*1,065239*	*1,065316*	*1,065514*

NE – Number of Edges, AE – All Edges and *NO* is the Number of Operations,

Table 3.1 Comparison between the algorithms for Circle image.

NE	50	100	150	200	**AE**
1^{st} algor. (*NO*)	*447287*	*447687*	*448087*	*448487*	*450351*
2^{nd} algor. (*NO*)	*446898*	*447298*	*447698*	*448098*	*448442*
3^{rd} algor. (*NO*)	*1, 726850*	*1, 727200*	*1, 727550*	*1, 727900*	*1, 728201*

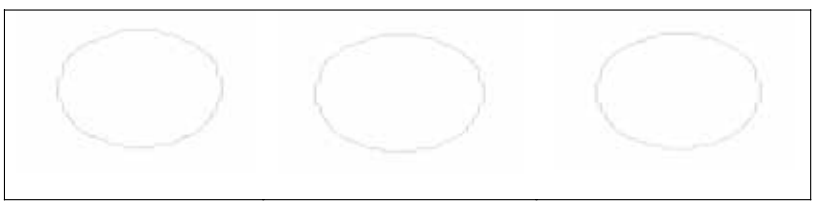
NE – Number of Edges, AE – All Edges and *NO* is the Number of Operations,

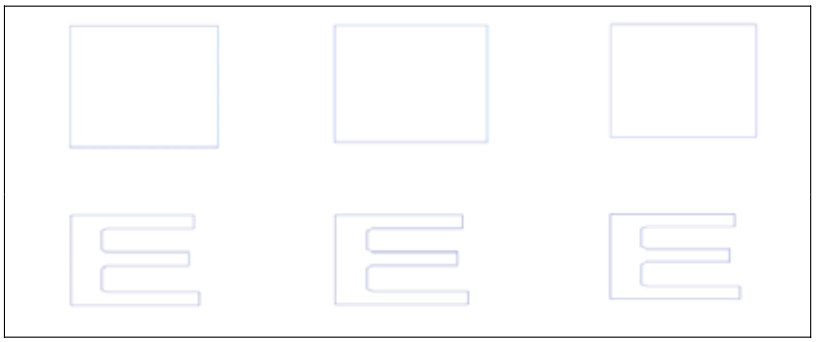
Table3.2 Comparison between the algorithms for Square image

NE	20	60	100	125	150	**AE**
1^{st} algor. (*NO*)	*109410*	*109732*	*110053*	*110254*	*110454*	*110718*
2^{nd} algor. (*NO*)	*56629*	*56951*	*57272*	*57473*	*57673*	*57937*
3^{rd} algor. (*NO*)	*407648*	*407930*	*408211*	*408387*	*408562*	*408793*

NE – Number of Edges, AE – All Edges and *NO* is the Number of Operations,

Table 3.3 Comparison between the algorithms for E letter image.



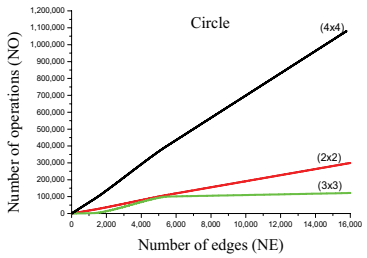


Extracted contours using first algorithm | Extracted contours using second algorithm | Extracted contours using third algorithm

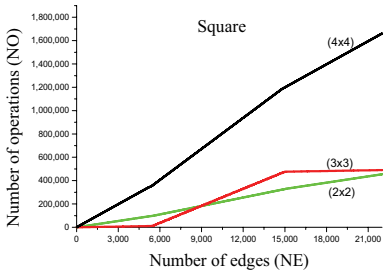
Fig. 1.18 Extracted contours using the three different algorithms.

The first column of Fig. 1.18 shows the extracted contours by the first algorithm. The second column shows the extracted contours by the second algorithm and the third one- the extracted contours by the third algorithm.

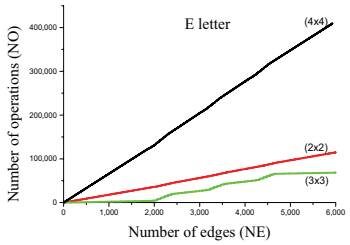
The comparison between the three algorithms with respect to the number of operations versus the number of edges for the binary images is illustrated in Fig. 1.19



(a)



(b)



(c)

Fig. 1.19 Number of operations versus number of edges for the all algorithms for the shapes of (a) Circle (b) Square (c) E letter.

The results presented in Fig.1.19 show that the fastest algorithm is dependent on the structure of contour.

2. Methods of Contour Approximation and Compression in Spatial Domain

In many applications of contour processing and analysis it is desirable to obtain a polygonal approximation of an object under consideration. In this chapter we briefly consider the algorithms that have been introduced for polygonal approximation of extracted contours. The algorithm presented by Ramer uses the maximum distance of the curve from the approximating polygon as a fit criterion [45]. There exist algorithms referred to as the Triangle family of contour approximation . The first algorithm is based on the ratio between the height and length triangle distances for each segment, and this ratio is used as the fit criterion of the algorithm which is referred to as height over length triangle ratios algorithm [46] and [47]. The second algorithm is based on the height triangle distance for each segment as the fit criterion of the algorithm which is referred to as height length triangle algorithm. The third algorithms is related to the square of the height triangle distance for each segment as the fit criterion of the algorithm which is referred to as height square length triangle algorithm. The fourth algorithm is associated with the area for each triangle segment as the fit criterion of the algorithm which is referred to as the area triangle algorithm.

2.1 Polygonal approximation

A digitized picture in a 2D array of points is often desired to be approximated by polygonal lines with the smallest number of sides, under the given error tolerance E.

There are several algorithms available for determining the number and location of the vertices and also to compute the polygonal approximation of a contour. The Ramer method is based on the polygonal approximation scheme. The simplest approach for the polygonal approximation is a recursive process (Splitting methods). Splitting methods is performed by first drawing a line from one point on the boundary to another. Then, we compute the perpendicular distance from each point along the segment to the line. If this exceeds some threshold, we break the line at the point of greatest error. We then repeat the process recursively for each of the two new lines until we don't need to break any

more. For a closed contour, we can find the two points that lie farthest apart and fit two lines between them, one for one side and one for the other. Then, we can apply the recursive splitting procedure to each side. First, use a single straight line to connect the end points. Then find the edge point with the greatest distance from this straight line. Then split the straight line in two straight lines that meet at this point. Repeat this process with each of the two new lines. Recursively repeat this process until the maximum distance of any point to the poly-line falls below a certain threshold. Finally draw the lines between the vertices of an edge of the reconstructed contour to obtain the polygonal approximating contour.

The approximation of arbitrary two-dimensional curves by polygons is an important technique in image processing. For many applications, the apparent ideal procedure is to represent lines and boundaries by means of polygons with a minimum number of vertices and satisfying a given fit criterion. An approximation algorithm is presented which uses an iterative method to produce a small - but not minimum - number of vertices that lie on the given curve. The maximum distance of the curve from the approximated polygon is chosen as the fit criterion.

Analysis of multiple views of the same scene is an area of active research in computer vision. The study of the structure of points and lines in two views received much attention in the eighties and early nineties [38], [39] and [40]. Studies on the constraints existent in three and more views have followed since then [41], [42], [43] and [44]. These multiview studies have concentrated on how geometric primitives like points, lines and planes are related across views. Specifically, the algebraic constraints satisfied by the projections of such primitives in different views have been a focus of intense studies.
Polygonal approximation is illustrated in Fig. 2.1

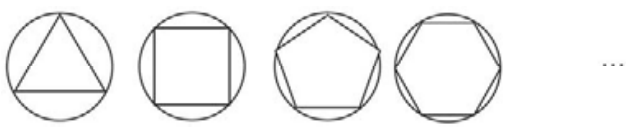


Fig. 2.1 Polygonal approximation.

Some practical methods for contour approximation are analyzed below.

2.2 Ramer Method

The algorithm is based on the maximum distance of the curve from the approximating polygon, and this distance is used as the fit criterion. The algorithm produces a polygon with a small number of edges for arbitrary two-dimensional digitized curves. The segment of the curve is approximated by a straight-line segment connecting its initial and terminus. If the fit is not fulfilling, the curve segment is terminated into two segments at the curve point most distant from the straight-line segment. This loop is repeated until each curve segment can be approximated by a straight-line segment through its endpoints. The termini of all these curve segments then are the vertices of a polygon that satisfies the given maximum-distance approximation criterion.

This type of polygonal curve representation exhibits two important disadvantages. First the polygons contain a very large number of edges and, therefore, are not in as compact a

form as possible. Second, the length of the edges is comparable in size to the noise introduced by quantization.

The idea is illustrated in the following Figures (see Fig. 2.2 to Fig. 2.11).

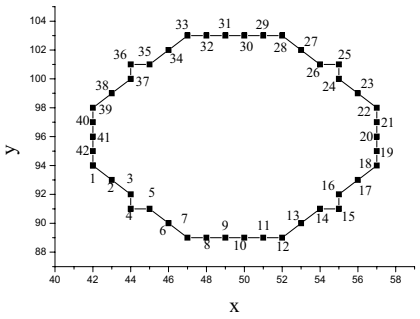


Fig. 2.2 The original contour.

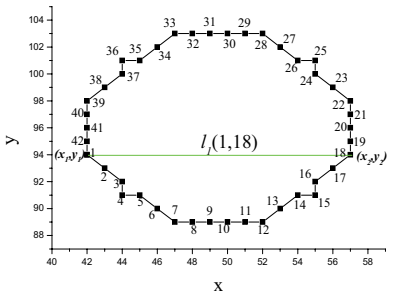


Fig. 2.3 The curve segment of straight line l_1.

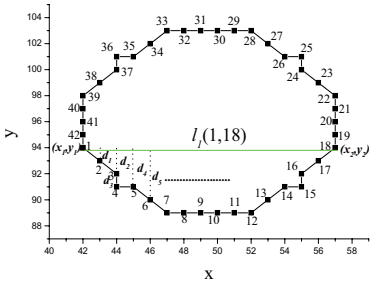


Fig. 2.4 Computation of the perpendicular distance between points in the curve segment and a line segment l_1.

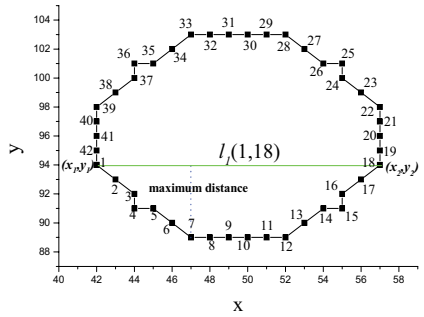


Fig. 2. 5 The maximum distance for the segment curve segment from the straight line l_1.

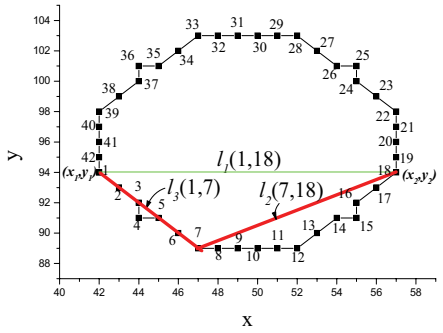


Fig. 2.6 The curve segments for the two new straight lines l_2 and l_3.

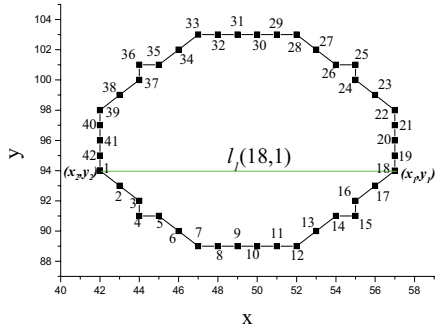


Fig. 2.7 The curve segment of straight line l_1.

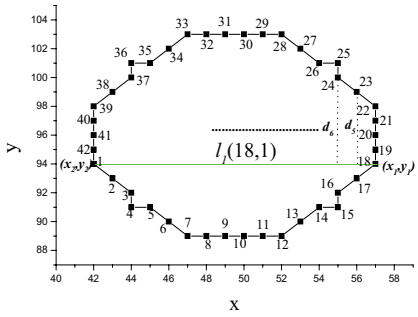


Fig. 2.8 Computation of the perpendicular distance between points in the curve segment and a line l_1.

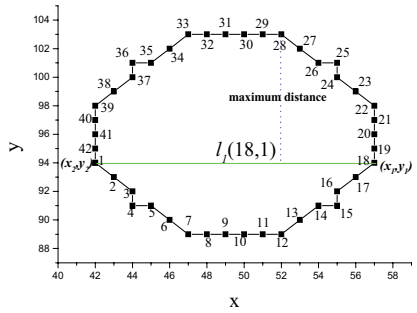


Fig. 2.9 The maximum distance for the curve segment from the straight line l_1.

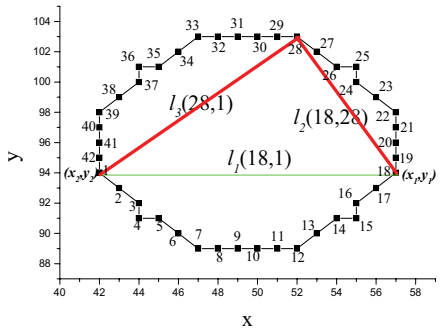


Fig. 2.10 The curve segments for the two new straight lines l_2 and l_3.

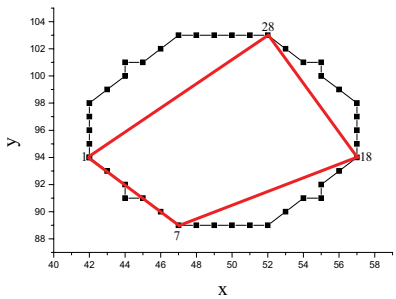


Fig. 2.11 The Original and approximated contours.

2.3 Triangle Methods of Contour Compression

The proposed algorithms belong to a family of polygonal methods of approximation. An input contour for the algorithms is extracted from 256 x 256 grey-scale images using Single Step Parallel Contour Extraction (SSPCE) method [34].

The approximation procedure starts at the time, when the first and last points of a segment are determined. The proposed criterion can be modified depending on contour representation methods. The most popular contour description methods are Freeman's chain coding, polar and Cartesian descriptions. Freeman chain coding can be used to distinguish all possible connections for both 8-connectivity and 4-connectivity schemes. A commonly used chain coding representation is the 8-Directional chain coding which uses eight possible directions to present all possible line segments connecting the nearest neighbors according to the 8-connectivity scheme. The contour extraction by these algorithms is based on (3 x 3) pixels window.

The Triangle family contains four methods of contour compression which are very similar to each other and the first method will be described in details in the following section.

A) Height Over Length Triangle Ratio Method

The algorithm refers to a quite new polygonal approximating method called the height over length ratio triangle method [46] and [47].

The idea of this method consists in segmentation of the contour points to get a triangle shape. The ratio of the height of the triangle (h) and the length of the base of the triangle (b) is then compared with the given threshold value as follows:

$$(h/b) < th \tag{4.1}$$

Where:

th - given threshold value.

The first point of each segment is called the starting point (SP) and the last one is called the ending point (EP). To calculate these values a simple trigonometric formula is used.

If the ratio value is smaller than the threshold according to Eqs. (4.1) the EP of the triangle is stored and SP is shifted to the EP, then a new segment is drawn. Otherwise the second point

(B) is stored and the SP is shifted to the B point of the triangle. Then a new segment is drawn. The stored points determine the vertices of an edge of the approximating polygon.
The algorithm scans contour points only once i.e. it does not require the storage of the analysed contour points. The original points of the contour are discarded as soon as they are processed. Only the co-ordinates of the starting point of the contour segment, and the last processed point are stored. The idea of the proposed algorithm is illustrated in Fig. 2.12. A flowchart of the proposed algorithm is depicted in Fig. 2.13.

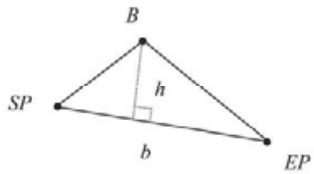


Fig. 2.12 Illustration of the basic triangle for the proposed algorithm where h and b are height and length of the triangle respectively.

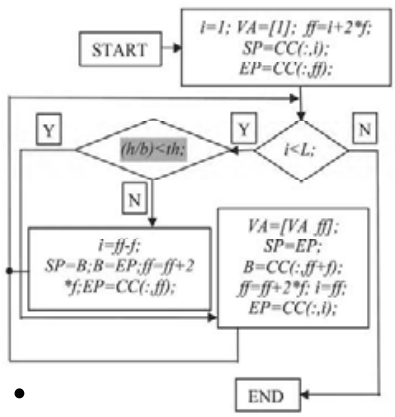


Fig. 2.13 Flowchart of the proposed algorithm.

where:
VA - sequence of indices of the final vertices;
CC - sequence of the input for the contour;
SP - starting point;
EP - ending point;
h ,*b* and *th* - as mentioned before (see Fig.4.13 and Eqs.4.1);
f - length between each two points of the triangle.

B) Height Triangle Method

The second algorithm refers to a recent polygonal approximating method called the height triangle method. The idea of this method is very similar to the previous algorithm. The only

difference is that the threshold is compared with the height of the triangle (shadow region in Fig. 2.13).
The third algorithm refers to a polygonal approximating method called the height square triangle method. The idea of this method is very similar to the previous algorithm. The difference is that the threshold is compared with the square height of the triangle (shadow region in Fig. 2.13).
The fourth algorithm refers to a recent polygonal approximating method called the triangle area method. In this case the threshold is compared with the area of the triangle (shadow region in Fig. 2.13).

2.4 Comparison Between the Triangle Family and Ramer Algorithms

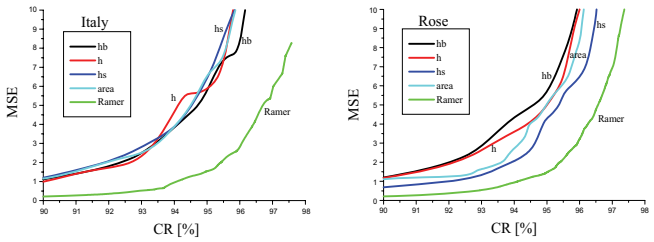
The computational complexity is one of the most important factors in evaluating a given method of approximation. High computational complexity leads to high implementation cost. The MSE (Mean Square Error) and SNR (Signal to Noise Ratio) criterions versus compression ratio are also used to evaluate the distortion.
The comparison is done for some test contours (Italy & Rose) which was extracted by using the "SSPCE" (Single Step Parallel Contour Extraction). The comparison is made between the following five algorithms:

- Height over length triangle (hb) method; it will be referred to as the first algorithm.
- Height triangle (h) method; it will be referred as the second algorithm.
- Height square triangle (hs) method; it will be referred as the third algorithm.
- Area triangle (area) method; it will be referred as the fourth algorithm.
- Ramer method; it will be referred as the fifth algorithm.

To visualise the experimental results a set of two test contours was selected. Selected contours are shown in Fig. 2.14.

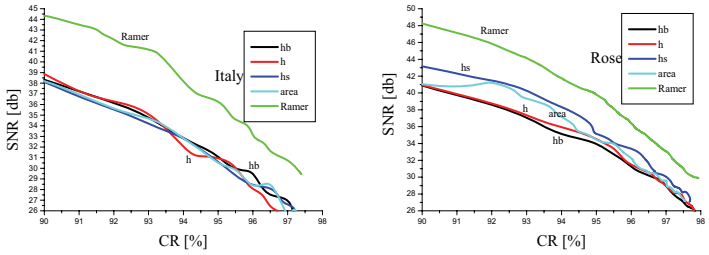
Fig. 2.14 Test contours: a) Italy b) Rose.

The comparison of the compression abilities versus the MSE & SNR are shown in the Fig. 2.15 & Fig. 2.16 respectively.



(a) (b)

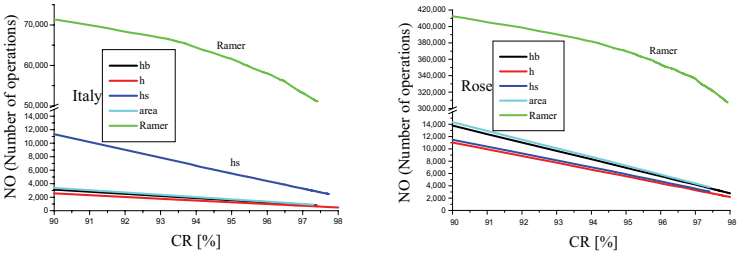
Fig. 2.15 MSE versus compression ratio for (a) Italy contour (b) Rose contour.



(a) (b)

Fig. 2.16 SNR versus compression ratio for (a) Italy contour (b) Rose contour.

Comparison of the compression abilities versus the number of operations is presented in Fig. 2.17.



(a) (b)

Fig. 2.17 NO versus compression ratio for (a) Italy contour (b) Rose contour.

The plots show that SNR using the Ramer algorithm is close to the triangle family methods for the rose contour; the reconstruction quality by the triangle family algorithms are very similar but the (hs) method is much better for complicated contour as in Rose contour. The number of operations is very similar between the triangle family algorithms at high compression. The triangle family algorithms are many times faster than that of Ramer

method. The compression ratio using triangle family methods can be even greater than 97% without significant lose of quality of compressed contour, but the complexity is much less than that of the Ramer algorithm.

3. References

[1] Luciano da Fontoura Costa and Roberto M. Junior, "*Shape Analysis and Classification: Theory and Practice", CRC Pr., 2001.*

[2] A. K. Jain, "Fundamentals of Digital Image Processing", New Jersey: Prentice Hall International, 1989.

[3] H. Freeman, "Application of the Generalized Chain Coding Scheme to Map Data Processing", Proc. of the IEEE Conference on Pattern Recognition, Image Proc..,Chicago,May,1987

[4] Milan Sonka, Vaclav Hlavac, and Roger Boyle, "Image processing, Analysis, and Machine Vision", Books and Cole Publishing 1998.

[5] Clarke R. J., "Transform Coding of Images", Academic Press, 1985.

[6] Brigham E.O., "The Fast Fourier Transform", Prentice-Hall, Englewood Cliffs, 1974.

[7] Walsh, J. L. "A Closed Set of Normal Orthogonal Functions", *Amer. J. Math.* 45, 5-24, 1923.

[8] Wolfram, S., "A New kind of Science", Champaign, IL: Wolfram Media, pp. 573 and 1072-1073, 2002.

[9] A. Dziech, F. Belgassem & H. J. Nern, "Image data compression using zonal sampling and piecewise-linear transforms," *Journal of Intelligent And Robotic Systems. Theory & Applications,* 28(1-2), Kluwer Academic Publishers, June 2000, 61-68.

[10] Kunt, M., *"TRAITEMENT NUMERIQUE DES SIGNAUX"*, Editions Presses polytechniques romandes, 1981 ISBN 2-88074-060-6.

[11] D. H. Ballard and C. M. Brown, "Computer Vision", Prentice Hall, Englewood Cliffs, NJ, 1982.

[12] R. M. Haralick and L.G. Shapiro, "Computer and Robot Vision", Addison-Wesley Publishing Co., 1992.

[13] B.K.P. Horn, "Robot Vision", The MIT Press, Cambridge, MA, 1986.

[14] W.K. Pratt, "Digital Image Processing", Wiley, New York, 1991.

[15] Capson D., Kitai R., "Performance Comparisons of Contour Extraction Algorithm", IEEE Instrumentation and Measurement Technology Conf., Boulder, USA, pp.99-103, March 1986.

[16] Fu. K. S., Mui J. K., "A Survey on Image Segmentation", Pattern Recognition, Vol.13, pp.3-16, 1981.

[17] Prager J. M., "Extracting and Labelling Boundary Segments in Natural Scenes", IEEE Transactions on PAMI, Vol. PAM1-2, No.1, pp.16-27, Jan. 1980.

[18] A. Nabout and H. A. Nour Eldin, "The Topological Contour Closure Requirement for Object Extraction from 2D-Digital Images", *Proc. 1993 IEEE International Conference on Systems, Man and Cybernetics (Le Touquet, France), 1993,* pp. 120-125.

[19] Canny, J., "A Computational Approach To Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

[20] Lindeberg, Tony, "Edge detection and ridge detection with automatic scale selection", International Journal of Computer Vision, 30, 2, pp. 117-154, 1998.
[21] Rhys Lewis, "*Practical Digital Image Processing",* Ellis Horwood, New York ,1990.
[22] J. Canny, "A Computational Approach to Edge Detection", *IEEE Trans. on Pattern* Analysis and Machine Intelligence **8:6** (1986) 679-698.
[23] J. S. Chen and G. Medioni, "Detection, Localization, and Estimation of Edges", *IEEE Trans. On Pattern Analysis and Machine Intelligence* **11:2** (February 1996) 191-198.
[24] W. E. L. Grimson and T. Pavlidis, "Discontinuity detection for visual surface Reconstruction", *Computer Vision, Graphics, and Image Processing* **30** (1985) 316-330.
[25] W. H. H. J. Lunscher and M. P. Beddoes, "Optimal edge detector design : parameter selection and noise effects", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **8:2** (March 1986) 164-177.
[26] D. Marr, E. Hildreth, "Theory of edge detection", *Proceedings of Royal Society of London,* **B:207** (1980) 187-217.
[27] L. S. Davis, "A survey of edge detection techniques", *Computer Vision, Graphics, and image processing 4 (1975),* pp. 248-270.
[28] R. M. Haralick, "Edge and region analysis for digital image data", *Computer Vision, Graphics and Image Processing (1980),* pp. 60-73.
[29] Sadayuki Hongo, Mitsuo Kawato, Toshi Inui and Sei Miyake, "Contour extraction of images on parallel computer-local, Parallel and stochastic algorithm which learns energy parameters", *International Joint Conference on Neural Networks (Washington D.C.), 1989,* pp. 161-168.
[30] P. K. Sahoo, S. Soltani, A. K.C. Wong and Y.C. Chen, "A survey of thresholding techniques", *Computer Vision, Graphics and Image Processing (1988),* pp. 233-260.
[31] Duda R. O., and Hart P. E., "Pattern Classification and Scene Analysis", John Wiley & Sons, New York, 1973.
[32] Heijden F., "Image Based Measurement Systems", John Wiley, 1995.
[33]] Nabout A., Su B., Nour Eldin H. A., "A Novel Closed Contour Extractor, Principle and Algorithm", *Proc. Of the International IEEE/ISCAS Conf. On Circuits and Systems, April 29 – May 3, Seattle, USA, Vol.1,* pp. 445-448, 1995.
[34] A. Dziech, W. S. Besbas, "Fast Algorithm for Closed Contour Extraction", *Proc. Of the Int. Workshop on Systems, Signals and Image Processing*, Poznań, Poland,1997, pp. 203-206.
[35] W. Besbas, "Contour Extraction, Processing and Recognition", Poznan University of Technology, Ph. D. Thesis, 1998.
[36] M. Bennamoun, B. Boashash, and A. Bower, "Object contour Extraction from motion parameters", *Proc. 2nd Australian and New Zealand Conference on intelligent Information Systems (Brisbane, Australia),* pp. 337- 341.
[37] M. Fleck, "Some defects in finite-difference edge finders", *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992),* pp. 337-345.
[38] O. Faugeras and Q. Luong, "The Geometry of Multiple Images", MIT Press, 2001.
[39] R. Hartley and A. Zisserman, "Multiple View Gemoetry in Computer Vision", Cambridge University Press, 2000.

[40] H. C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from two Projections", *Nature*, 293:133-135, 1981.

[41] R. Hartley, "Lines and points in three views: An integrated approach", *Proc. ARPA Image Understanding Workshop*, 1994.

[42] A. Shashua, "Algebraic Functions for Recognition", *IEEE Tran. Pattern Anal. Machine Intelligence*, 16:778-790, 1995.

[43] A. Shashua, "Trilinear tensor: The Fundamental Construct of Multiple-view Geometry and its applications", *Int. Worskshop on AFPAC*, 1997.

[44] B. Triggs, "Matching Constraints and the Joint Image", *International Conference on Computer Vision*, pages 338-343, 1995.

[45] U. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plane Curves", *Computer Graphics and Image Proc., Academic Press*, 1972, pp.244-256.

[46] Andrzej Dziech, Ali Ukasha and Remigiusz Baran, "A New Method For Contour Compression", *WSEAS Int. Conf. on Signal, Speech and Signal Processing (SSIP 2005),* Corfu Island, Greece, August 17-19, 2005, pp.282-286.

[47] Andrzej Dziech, Ali Ukasha and Remigiusz Baran, "Fast Method For Contour Approximation And Compression", *WSEAS Transactions on Communications Issue 1, Volume 5,* January 2006, pp. 49 - 56.

11

Comparative Analysis of Mobile Robot Localization Methods Based On Proprioceptive and Exteroceptive Sensors

Gianluca Ippoliti, Leopoldo Jetto, Sauro Longhi, Andrea Monteriù
Dipartimento di Ingegneria Informatica Gestionale e dell'Automazione
Università Politecnica delle Marche
Via Brecce Bianche – 60131 Ancona
Italy

1. Introduction

Two different approaches to the mobile robot localization problem exist: relative and absolute. The first one is based on the data provided by sensors measuring the dynamics of variables internal to the vehicle; absolute localization requires sensors measuring some parameters of the environment in which the robot is operating. If the environment is only partially known, the construction of appropriate ambient maps is also required. The actual trend is to exploit the complementary nature of these two kinds of sensorial information to improve the precision of the localization procedure (see e.g. (Bemporad et al., 2000; Bonci et al., 2004; Borenstein et al., 1997; Durrant-Whyte, 1988; Gu et al., 2002; Ippoliti et al., 2004)) at expense of an increased cost and computational complexity. The aim is to improve the mobile robot autonomy by enhancing its capability of localization with respect to the surrounding environment.

In this framework the research interests have been focused on multi-sensor systems because of the limitations inherent any single sensory device that can only supply partial information on the environment, thus limiting the ability of the robot to localize itself. The methods and algorithms proposed in the literature for an efficient integration of multiple-sensor information differ according to the a priori information on the environment, which may be almost known and static, or almost unknown and dynamic.

In this chapter both relative and absolute approaches of mobile robot localization are investigated and compared. With reference to relative localization, the purpose of this chapter is to propose and to compare three different algorithms for the mobile robot localization only using internal sensors like odometers and gyroscopes. The measurement systems for mobile robot localization only based on relative or dead-reckoning methods, such as encoders and gyroscopes, have the considerable advantage of being totally self-contained inside the robot, relatively simple to use and able to guarantee a high data rate. A drawback of these systems is that they integrate the relative increments and the localization errors may considerably grow over time if appropriate sensor-fusion algorithms are not used (De Cecco, 2003). Here, different methods are analysed and tested. The best performance has been obtained in the stochastic framework where the localization problem has been formulated as a state estimation problem and the Extended Kalman Filtering (EKF)

is used. The EKF fuses together odometric and gyroscopic data. A difference with respect to other EKF based techniques is that the approach followed here derives the dynamical equation of the state-space form from the kinematic model of the robot, while the measure equation is derived from the numerical integration equations of the encoder increments. This allows to fuse together all the available informative content which is carried both by the robot dynamics and by the acquired measures.
As previously mentioned, any relative localization algorithm is affected by a continuous growth in the integrated measurement error. This inconvenience can be reduced by periodically correcting the internal measures with the data provided by absolute sensors like sonar, laser, GPS, vision systems (Jarvis, 1992; Talluri & Aggarwal, 1992; Zhuang & Tranquilla, 1995; Mar & Leu, 1996; Arras et al., 2000; Yi et al., 2000; Panzieri et al., 2002). To this purpose, a localization algorithm based on a measure apparatus composed of a set of proprioceptive and exteroceptive sensors, is here proposed and evaluated. The fusion of internal and external sensor data is again realized through a suitably defined EKF driven by encoder, gyroscope and laser measures.
The developed algorithms provide efficient solutions to the localization problem, where their appealing features are:

- The possibility of collecting all the available information and uncertainties of a different kind into a meaningful state-space representation,
- The recursive structure of the solution,
- The modest computational effort.

Significant experimental results of all proposed algorithms are presented here, and their comparison concludes this chapter.

2. The sensors equipment

In this section the considered sensor devices are introduced and characterized.

2.1 Odometric measures

Consider a unicycle-like mobile robot with two driving wheels, mounted on the left and right sides of the robot, with their common axis passing through the center of the robot (see Fig. 1).

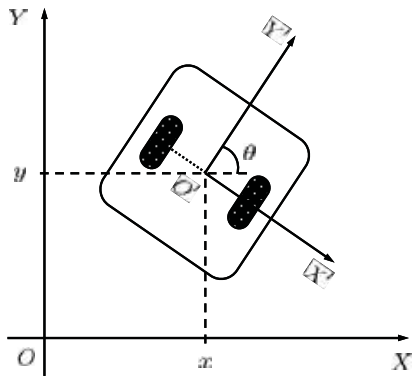


Fig. 1. The scheme of the unicycle robot.

Localization of this mobile robot in a two-dimensional space requires the knowledge of coordinates x and y of the midpoint between the two driving wheels and of the angle θ between the main axis of the robot and the X-direction. The kinematic model of the unicycle robot is described by the following equations:

$$\dot{x}(t) = v(t)\cos\theta(t) \tag{1}$$

$$\dot{y}(t) = v(t)\sin\theta(t) \tag{2}$$

$$\dot{\theta}(t) = \omega(t) \tag{3}$$

where $v(t)$ and $\omega(t)$ are, respectively, the displacement velocity and the angular velocity of the robot and are expressed by:

$$v(t) = \frac{\omega_r(t) + \omega_l(t)}{2} r \tag{4}$$

$$\omega(t) = \frac{\omega_r(t) - \omega_l(t)}{d} r \tag{5}$$

where $\omega_r(t)$ and $\omega_l(t)$ are the angular velocities of the right and left wheels, respectively, r is the wheel radius and d s the distance between the wheels.

Assuming constant $\omega_r(t)$ and $\omega_l(t)$ over a sufficiently small sampling period $\Delta t_k := t_{k+1} - t_k$, the position and orientation of the robot at time instant t_{k+1} can be expressed as:

$$x(t_{k+1}) = x(t_k) + \bar{v}(t_k)\Delta t_k \frac{\sin\dfrac{\bar{\omega}(t_k)\Delta t_k}{2}}{\dfrac{\bar{\omega}(t_k)\Delta t_k}{2}} \cos\left(\theta(t_k) + \frac{\bar{\omega}(t_k)\Delta t_k}{2}\right) \tag{6}$$

$$y(t_{k+1}) = y(t_k) + \bar{v}(t_k)\Delta t_k \frac{\sin\dfrac{\bar{\omega}(t_k)\Delta t_k}{2}}{\dfrac{\bar{\omega}(t_k)\Delta t_k}{2}} \sin\left(\theta(t_k) + \frac{\bar{\omega}(t_k)\Delta t_k}{2}\right) \tag{7}$$

$$\theta(t_{k+1}) = \theta(t_k) + \bar{\omega}(t_k)\Delta t_k \tag{8}$$

where $\bar{v}(t_k)\Delta t_k$ and $\bar{\omega}(t_k)\Delta t_k$ are:

$$\bar{v}(t_k)\Delta t_k = \frac{\Delta q_r(t_k) + \Delta q_l(t_k)}{2} r \tag{9}$$

$$\bar{\omega}(t_k)\Delta t_k = \frac{\Delta q_r(t_k) - \Delta q_l(t_k)}{d} r. \tag{10}$$

The terms $\Delta q_r(t_k)$ and $\Delta q_l(t_k)$ are the incremental distances covered on the interval Δt_k by the right and left wheels of the robot respectively. Denote by $y_r(t_k)$ and $y_l(t_k)$ the measures of $\Delta q_r(t_k)$ and $\Delta q_l(t_k)$ respectively, provided by the encoders attached to wheels, one has

$$y_r(t_k) = \Delta q_r(t_k) + s_r(t_k) \tag{11}$$

$$y_l(t_k) = \Delta q_l(t_k) + s_l(t_k) \tag{12}$$

where $s_r(\cdot)$ and $s_l(\cdot)$ are the measurement errors, which are modelled as independent,

zero mean, gaussian white sequences $\left(s_r(\cdot) - N\left(0,\sigma_r^2\right),\ s_l(\cdot) - N\left(0,\sigma_l^2\right)\right)$ (Wang, 1988). It follows that the really available values $y_v(t_k)$ and $y_\omega(t_k)$ of $\bar{v}(t_k)\Delta t_k$ and $\bar{\omega}(t_k)\Delta t_k$ respectively are given by:

$$y_v(t_k) = \frac{y_r(t_k) + y_l(t_k)}{2} r = \bar{v}(t_k)\Delta t_k + \eta_v(t_k) \tag{13}$$

$$y_\omega(t_k) = \frac{y_r(t_k) - y_l(t_k)}{2} r = \bar{\omega}(t_k)\Delta t_k + \eta_\omega(t_k) \tag{14}$$

where $\eta_v(\cdot)$ and $\eta_\omega(\cdot)$ are independent, zero mean, gaussian white sequences $\left(\eta_v(\cdot) \sim N\left(0,\sigma_v^2\right), \eta_\omega(\cdot) \sim N\left(0,\sigma_\omega^2\right)\right)$, where, by (9) and (10), $\sigma_v^2 = \left(\sigma_r^2 + \sigma_l^2\right) r^2 / 4$ and $\sigma_\omega^2 = \left(\sigma_r^2 + \sigma_l^2\right) r^2 / d^2$.

2.2 The Fiber optic gyroscope measures

The operative principle of a Fiber Optic Gyroscope (FOG) is based on the Sagnac effect. The FOG is made of a fiber optic loop, fiber optic components, a photo-detector and a semiconductor laser. The phase difference of the two light beams traveling in opposite directions around the fiber optic loop is proportional to the rate of rotation of the fiber optic loop. The rate information is internally integrated to provide the absolute measurements of orientation. A FOG does not require frequent maintenance and have a longer lifetime of the conventional mechanical gyroscopes. In a FOG the drift is also low. A complete analysis of the accuracy and performances of this internal sensor has been developed in (Killian, 1994; Borenstein & Feng, 1996; Zhu et al., 2000; Chung et al., 2001). This internal sensor represents a simple low cost solution for producing accurate pose estimation of a mobile robot. The FOG readings are denoted by $y_\theta(\cdot) = \theta_g(\cdot) + \eta_\theta(\cdot)$, where $\theta_g(\cdot)$ is the true value and $\eta_\theta(\cdot)$ is an independent, zero mean, gaussian white sequence $\left(\eta_\theta(\cdot) - N\left(0,\sigma_\theta^2\right)\right)$.

2.3 Laser scanner measures

The distance readings by the Laser Measurement System (LMS) are related to the in-door environment model and to the configuration of the mobile robot.

Denote with l the distance between the center of the laser scanner and the origin O' of the coordinate system (O', X', Y') fixed to the mobile robot, as reported in Fig. 2.

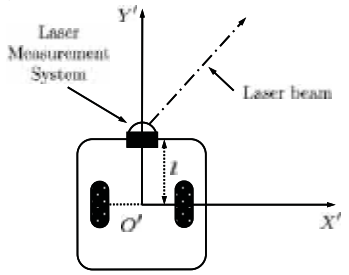


Fig. 2. Laser scanner displacement.

At the sampling time t_k, the position x_s, y_s and orientation θ_s of the center of the laser scanner, referred to the inertial coordinate system (O,X,Y), have the following form:

$$x_s(t_k) = x(t_k) + l\cos\theta(t_k) \tag{15}$$

$$y_s(t_k) = y(t_k) + l\sin\theta(t_k) \tag{16}$$

$$\theta_s(t_k) = \theta(t_k) \tag{17}$$

The walls and the obstacles in an in-door environment are represented by a proper set of planes orthogonal to the plane XY of the inertial coordinate system. Each plane P^j, $j \in \{1,2,\ldots,n_p\}$ (where n_p is the number of planes which describe the indoor environment), is represented by the triplet P_r^j, P_n^j and P_v^j, where P_r^j is the normal distance of the plane from the origin O, P_n^j is the angle between the normal line to the plane and the X-direction and P_v^j is a binary variable, $P_v^j \in \{-1,1\}$, which defines the face of the plane reflecting the laser beam. In such a notation, the expectation of the i-th $(i = 1,2,\ldots,n_s)$ laser reading $d_i^j(t_k)$, relative to the present distance of the center of the laser scanner from the plane P^j, has the following expression (see Fig. 3):

$$d_i^j(t_k) = \frac{P_v^j\left(P_r^j - x_s(t_k)\cos P_n^j - y_s\left(t_k \sin P_n^j\right)\right)}{\cos\theta_i^j} \tag{18}$$

where

$$\theta_i^j = P_n^j - \theta_i^* \tag{19}$$

with $\theta_i^* \in [\theta_0, \theta_1]$ given by (see Fig. 4):

$$\theta_i^* = \theta_s + \theta_i - \frac{\pi}{2}. \tag{20}$$

The vector composed of geometric parameters P_r^j, P_n^j and P_v^j, $j \in \{1,2,\ldots,n_p\}$, is denoted by $\prod$.

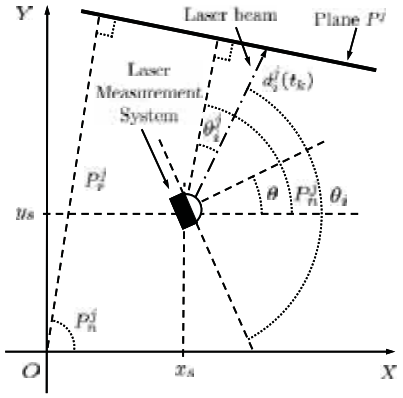


Fig. 3. Laser scanner measure.

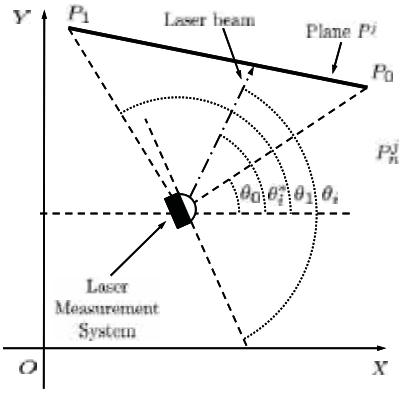


Fig. 4. Laser scanner field of view for plane P^j.

The laser readings $y_{d_i^j}(t_k)$ are denoted by $y_{d_i^j}(\cdot) = d_i^j(\cdot) + \eta_i(\cdot)$, where $d_i^j(\cdot)$ is the true value expressed by (18) and $\eta_i(\cdot)$ is an independent, zero mean, gaussian white sequence $\left(\eta_i(\cdot) \sim N\left(0, \sigma_i^2\right)\right)$.

3. Relative approaches for mobile robot localization

The purpose of this section is to propose and to compare three different algorithms for the mobile robot localization only using internal sensors like odometers and gyroscopes. The first method (Algorithm 1) is the simplest one and is merely based on a numerical integration of the raw encoder data; the second method (Algorithm 2) replaces the gyroscopic data into the equations providing the numerical integration of the increments provided by the encoders. The third method (Algorithm 3) operates in a stochastic framework where the uncertainty originates by the measurement noise and by the robot model inaccuracies. In this context the right approach is to formulate the localization problem as a state estimation problem and the appropriate tool is the EKF (see e.g. (Barshan & Durrant-Whyte, 1995; Garcia et al., 1995; Kobayashi et al., 1995; Jetto et al., 1999; Sukkarieh et al., 1999; Roumeliotis & Bekey, 2000; Antoniali & Oriolo, 2001; Dissanayake et al., 2001)). Hence, Algorithm 3 is a suitably defined EKF fusing together odometric and gyroscopic data. In the developed solution, the dynamical equation of the state-space form of the robot kinematic model, has been considered. The numerical integration equations of the encoder increments have been considered for deriving the measure equation. This allows to fuse together all the available informative content which is carried both by the robot dynamics and by the acquired measures.

3.1 Algorithm 1

Equations (6)-(8) have been used to estimate the position and orientation of the mobile robot at time t_{k+1} replacing the true values of $\bar{v}(t_k)\Delta t_k$ and $\bar{\omega}(t_k)\Delta t_k$ with their measures $y_v(t_k)$ and $y_\omega(t_k)$ respectively, provided by the encoders. An analysis of the accuracy of this

estimation procedure has been developed in (Wang, 1988; Martinelli, 2002), where it is shown that the incremental errors on the encoder readings especially affect the estimate of the orientation $\theta(t_k)$ and reduce its applicability to short trajectories.

3.2 Algorithm 2

This algorithm is based on the ascertainment that the angular measure $y_\theta(t_k)$ provided by the FOG is much more reliable than the orientation estimate obtainable with Algorithm 1. Hence, at each time instant, Algorithm 2 provides an estimate of the robot position and orientation $\left[x(t_{k+1}), y(t_{k+1}), y_\theta(t_{k+1})\right]$, where $y_\theta(t_{k+1})$ is the FOG reading, $x(t_{k+1})$ and $y(t_{k+1})$ are computed through equations (6), (7), replacing $\bar{v}(t_k)\Delta t_k$ with its measure $y_v(t_k)$, $\theta(t_k)$ with $y_\theta(t_k)$ and $\bar{\omega}(t_k)\Delta t_k$ with $y_\theta(t_{k+1}) - y_\theta(t_k)$.

3.3 Algorithm 3

This algorithm operates in a stochastic framework exploiting the same measures of Algorithm 2. A state-space approach is adopted with the purpose of defining a more general method merging the information carried by the kinematic model with that provided by the sensor equipment. The estimation algorithm is an EKF defined on the basis of a state equation derived from (1)-(3) and of a measure equation inglobing the incremental measures of the encoders $y_v(t_k)$ and the angular measure of the gyroscope $y_\theta(t_k)$. This is a difference with respect to other existing EKF based approaches, (Barshan & Durrant-Whyte, 1995; Kobayashi et al., 1995; Sukkarieh et al., 1999; Roumeliotis & Bekey, 2000; Antoniali & Oriolo, 2001; Dissanayake et al., 2001b), where equations (1)-(3) are not exploited and the dynamical equation of the state-space model is derived from the numerical integration of the encoder measures.

Denote with $X(t) := \left[x(t), y(t), \theta(t)\right]^T$ the true robot state and with $U(t) := \left[v(t), \omega(t)\right]^T$ the robot control input. For future manipulations it is convenient to partition $X(t)$ as $X(t) := \left[X_1(t), \theta(t)\right]^T$, with $X_1(t) := \left[x(t), y(t)\right]^T$. The kinematic model of the robot can be written in the compact form of the following stochastic differential equation

$$dX(t) = F\left(X(t), U(t)\right)dt + d\eta(t) \tag{21}$$

where $F\left(X(t), U(t)\right)$ represents the set of equations (1)-(3) and $\eta(t)$ is a Wiener process such that $E\left(d\eta(t)\, d\eta(t)^T\right) = Q dt$. Its weak mean square derivative $d\eta(t)/dt$ is a white noise process $\sim N(0, Q)$ representing the model inaccuracies (parameter uncertainties, slippage, dragging). It is assumed that $Q = \sigma_\eta^2 I_3$, where I_n denote the $n \times n$ identity matrix. The diagonal form of Q understands the hypothesis that model (21) describes the true dynamics of the three state variables with nearly the same degree of approximation and with independent errors.

Let $\Delta t_k = T$ be the constant sampling period and denote t_k by kT, assume

$U(t)=U(kT):=U(k)$, for $t\in[kT,(k+1)T]$ and denote by $X(k)$ and by $\hat{X}(k,k)$ the current state and its filtered estimate respectively at time instant $t_k=kT$. Linearization of (15) about $U(k-1)$ and $\hat{X}(k,k)$ and subsequent discretization with period T results in the following equation

$$X(k+1)=A_d(k)X(k)+L(k)U(k)+D(k)+W(k) \tag{22}$$

Partitioning vectors and matrices on the right hand side of equation (22) according to the partition of the state vector one has

$$A_d(k)=\exp\big(A(k)T\big)=\begin{bmatrix}A_{1,1_d}(k) & A_{1,2_d}(k)\\ A_{2,1_d}(k) & A_{2,2_d}(k)\end{bmatrix},\ L(k)=\begin{bmatrix}L_1(k)\\ L_2(k)\end{bmatrix},\ D(k)=\begin{bmatrix}D_1(k)\\ D_2(k)\end{bmatrix} \tag{23}$$

$$A(k):=\left[\frac{\partial F\big(X(t),U(t)\big)}{\partial X(t)}\right]_{\substack{X(t)=\hat{X}(k,k)\\ U(t)=U(k-1)}}=\begin{bmatrix}0 & 0 & -\nu(k-1)\sin\hat{\theta}(k,k)\\ 0 & 0 & \nu(k-1)\cos\hat{\theta}(k,k)\\ 0 & 0 & 0\end{bmatrix} \tag{24}$$

$$A_{1,1_d}(k)=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}:=I_2,\ A_{1,2_d}(k)=\begin{bmatrix}-\nu(k-1)\sin\hat{\theta}(k,k)T\\ \nu(k-1)\cos\hat{\theta}(k,k)T\end{bmatrix} \tag{25}$$

$$A_{2,1_d}(k)=[0\quad 0],\ A_{2,2_d}(k)=1 \tag{26}$$

$$L_1(k)=\begin{bmatrix}T\cos\hat{\theta}(k,k) & -0.5\nu(k-1)T^2\sin\hat{\theta}(k,k)\\ T\sin\hat{\theta}(k,k) & 0.5\nu(k-1)T^2\cos\hat{\theta}(k,k)\end{bmatrix},\ L_2(k)=[0\quad T] \tag{27}$$

$$D_1(k)=\begin{bmatrix}T\nu(k-1)\hat{\theta}(k,k)\sin\hat{\theta}(k,k)\\ -T\nu(k-1)\hat{\theta}(k,k)\cos\hat{\theta}(k,k)\end{bmatrix},\ D_2(k)=0 \tag{28}$$

$$W(k):=\int_{kT}^{(k+1)T}\exp\big(A(k)\big[(k+1)zT-\tau\big]\big)\eta(\tau)\,d\tau=\begin{bmatrix}W_1(k)\\ W_2(k)\end{bmatrix} \tag{29}$$

with $W_1(k)\in\Re^2$, $W_2(k)\in\Re^1$, $k=0,1,2,\ldots$.

The integral term $W(k)$ given (29) has to be intended as a stochastic Wiener integral, its covariance matrix is $E\big[W(k)W(k)^T\big]:=Q_d(k)=\sigma_\eta^2(k)\bar{Q}(k)$, where

$$\bar{Q}(k)=\begin{bmatrix}Q_{1,1}(k) & Q_{1,2}(k)\\ Q_{2,1}(k) & Q_{2,2}(k)\end{bmatrix} \tag{30}$$

$$Q_{1,1}(k)=\begin{bmatrix}T+\nu^2(k-1)\dfrac{T^3}{3}\sin^2\hat{\theta}(k,k) & -\nu^2(k-1)\dfrac{T^3}{3}\cos\hat{\theta}(k,k)\sin\hat{\theta}(k,k)\\ -\nu^2(k-1)\dfrac{T^3}{3}\cos\hat{\theta}(k,k)\sin\hat{\theta}(k,k) & T+\nu^2(k-1)\dfrac{T^3}{3}\cos^2\hat{\theta}(k,k)\end{bmatrix} \tag{31}$$

$$Q_{1,2}(k)=\begin{bmatrix}-\nu(k-1)\dfrac{T^2}{2}\sin\hat{\theta}(k,k)\\ -\nu(k-1)\dfrac{T^2}{2}\cos\hat{\theta}(k,k)\end{bmatrix},\ Q_{2,1}(k)=Q_{1,2}(k)^T,\ Q_{2,2}(k)=T. \tag{32}$$

Denote by $Z(k)=\left[z_1(k),z_2(k)\right]^T$ the measurement vector at time instant kT, the elements of $Z(k)$ are: $z_1(k)\equiv y_v(t_k)$, $z_2(k)\equiv y_\theta(t_k)$, where $y_v(t_k)$ is the measure related to the increments provided by the encoders through equations (9) and (13), $y_\theta(t_k)$ is the angular measure provided by the FOG. The observation noise $V(k)=\left[\eta_v(k),\eta_\theta(k)\right]^T$ is a white sequence $\sim N(0,R)$ where $R=\mathrm{diag}\left[\sigma_v^2,\sigma_\theta^2\right]$. The diagonal form of R follows by the independence of the encoder and FOG measures. As previously mentioned, the measure $z_2(k)$ provided by the FOG is much more reliable than $z_1(k)$, so that $\sigma_\theta^2 \ll \sigma_v^2$. This gives rise to a nearly singular filtering problem, where singularity of R arises due to the very high accuracy of a measure. In this case a lower order non singular EKF can be derived assuming that the original R is actually singular (Anderson & Moore, 1979). In the present problem, assuming $\sigma_\theta^2=0$, the nullity of R is $m=1$ and the original singular EKF of order $n=3$ can be reduced to a non singular problem of order $n-m=2$, considering the third component $\theta(k)$ of the state vector $X(k)$ coinciding with the known deterministic signal $z_2(k)=\theta_g(k)$. Under this assumption, only $X_1(k)$ needs be estimated as a function of $z_1(\cdot)$. As the measures $z_1(\cdot)$ provided by the encoders are in terms of increments, it is convenient to define the following extended state $\bar{X}(k):=\left[X_1(k)^T,X_1(k-1)^T\right]^T$ in order to define a measure equation where the additive gaussian noise is white. The dynamic state-space equation for $\bar{X}(k)$ is directly derived from (22), taking into account that, by the assumption on $z_2(\cdot)$, in all vectors and matrices defined in (25)–(32), the term $\hat{\theta}(k,k)$ must be replaced by $\theta_g(k)$.

The following equation is obtained

$$\bar{X}(k+1)=\bar{A}(k)\bar{X}(k)+\bar{L}(k)U(k)+\bar{B}(k)\theta_g(k)+\bar{D}(k)+\bar{W}(k) \tag{33}$$

where

$$\bar{A}(k)=\begin{bmatrix}I_2 & 0_{2,2}\\ I_2 & 0_{2,2}\end{bmatrix},\ \bar{L}(K)=\begin{bmatrix}L_1(k)\\ 0_{2,2}\end{bmatrix},\ \bar{B}(k)=\begin{bmatrix}A_{1,2,j}(k)\\ 0_{2,1}\end{bmatrix} \tag{34}$$

$$\bar{D}(k)=\begin{bmatrix}D_1(k)\\ 0_{2,1}\end{bmatrix},\ \bar{W}(k)=\begin{bmatrix}W_1(k)\\ 0_{2,1}\end{bmatrix} \tag{35}$$

$0_{i,j}$ being the $(i\times j)$ null matrix.

Equations (6), (7) and (13) and the way the state vector $\bar{X}(k)$ is defined imply that the $z_1(k)\equiv y_v(t_k)$ can be indifferently expressed as

$$z_1(k) = \left[\alpha(k)^{-1}, 0, -\alpha(k)^{-1}, 0 \right] \bar{X}(k) + \eta_v(k) \tag{36}$$

or

$$z_1(k) = \left[0, \beta(k)^{-1}, 0, \beta(k)^{-1} \right] \bar{X}(k) + \eta_v(k) \tag{37}$$

where

$$\alpha(kT) := \frac{\sin \frac{\bar{\omega}(t_k)\Delta t_k}{2}}{\frac{\bar{\omega}(t_k)\Delta t_k}{2}} \cos\left(\theta(t_k) + \frac{\bar{\omega}(t_k)\Delta t_k}{2} \right) \tag{38}$$

$$\beta(kT) := \frac{\sin \frac{\bar{\omega}(t_k)\Delta t_k}{2}}{\frac{\bar{\omega}(t_k)\Delta t_k}{2}} \sin\left(\theta(t_k) + \frac{\bar{\omega}(t_k)\Delta t_k}{2} \right) \tag{39}$$

with $\bar{\omega}(t_k)\Delta t_k = \theta_g(t_{k+1}) - \theta_g(t_k)$ and $\theta(t_k) = \theta_g(t_k)$. The measure equations (36) and (37) can be combined to obtain a unique equation where $z_1(k)$ is expressed as a function both of $x(k+1) - x(k)$ and of $y(k+1) - y(k)$. As the amount of observation noise is the same, equations (36) and (37) are averaged, obtaining

$$z_1(k) = C_1(k) \bar{X}(k) + v_1(k) \tag{40}$$

where $C_1(k) = \left[\alpha(k)^{-1}/2, \beta(k)^{-1}/2, -\alpha(k)^{-1}/2, -\beta(k)^{-1}/2 \right]$ and $v_1(k) := \eta_v(k)$. Equations (33) and (40) represent the linearized, discretized state-space form to which the classical EKF algorithm has been applied.

3.4 Experimental results

The experimental tests have been performed on the TGR Explorer powered wheelchair (TGR Bologna, 2000) in an indoor environment. This mobile base has two driving wheels and a steering wheel. The odometric system is composed by two optical encoders connected to independent passive wheels aligned with the axes of the driving wheels, as shown in Fig. 5. A sampling time of $0.4\,s$ has been used.

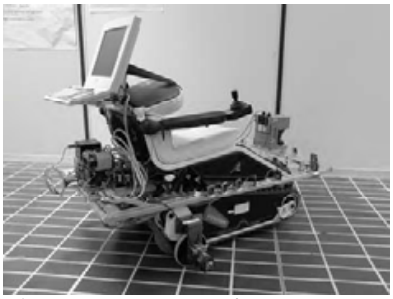


Fig. 5. TGR Explorer with data acquisition system for FOG sensor and incremental encoders.

The odometric data are the incremental measures that at each sampling interval are provided by the encoders attached to the right and left passive wheels. The incremental optical encoders SICOD mod. F3-1200-824-BZ-K-CV-01 have been used to collect the odometric data. Each encoder has 1200 *pulses/rev*. and a resolution of 0.0013 *rad*. These measures are directly acquired by the low level controller of the mobile base. The gyroscopic measures on the absolute orientation have been acquired in a digital form by a serial port on the on-board computer. The fiber optic gyroscope HITACHI mod. HOFG-1 was used for measuring the angle θ of the mobile robot. The main characteristics of this FOG are reported in the Table 1. While the used FOG measures the rotational rates with a very high accuracy, the internal integration of angular rates to derive the heading angle can suffer from drift (Barshan & Durrant-Whyte, 1995; Komoriya & Oyama, 1994). Because of the low rate integration drift of the used FOG (see Table 1), the drift is not accounted for in the proposed experiments where the robot task duration is on the order of several minutes. For longer task duration the rate integration drift can be compensated as proposed in (Ojeda et al., 2000) or can be periodically reset by a proper docking system or an absolute sensing mechanism (Barshan & Durrant-Whyte, 1995).

Rotation Rate	-1.0472 to +1.0472 rad/s
Angle Measurement Range	-6.2832 to +6.2832 *rad*
Random Walk	$\leq 0.0018 rad/\sqrt{h}$
Zero Drift (Rate Integration)	$\leq 0.00175 rad/h$
Non-linearity of Scale Factor	within $\pm$ 1.0%
Time Constant	Typ. 20 *ms*
Response Time	Typ. 20 *ms*
Data Output Interval	Min. 10 *ms*
Warm-up Time	Typ. 6.0 *s*

Table 1. Characteristics of the HITACHI gyroscope mod. HFOG - 1.

The navigation module developed for the considered mobile base interacts with the user in order to involve her/him in the guidance of the vehicle without limiting the functionality and the security of the system. The user sends commands to the navigation module through the user interface and the module translates the user commands in the low level command for the driving wheels. Two autonomy levels are developed to perform a simple filtering or to introduce some local corrections of the user commands on the basis of the environment information acquired by a set of sonar sensors (for more details see (Fioretti et al., 2000)). The navigation system is connected directly with the low level controller and with the Fiber Optic Gyroscope by analog and digital converters and serial port RS232, respectively.

All the experiments have been performed making the mobile base track relatively long trajectories. In the indoor environment of our Department a closed trajectory of 108 *m* length, characterized by a lot of orientation changes has been considered. The trajectory has been imposed by the user interface with the end configuration coincident with the start configuration. In order to quantify the accuracy of the proposed localization algorithms, six markers have been introduced along the trajectory. The covariance matrix R of the observation noise $V(\cdot)$ has been determined by an analysis of the sensor characteristics. The detected estimate errors in correspondence of the marker

configurations (the distance between the marker and the corresponding estimated configuration) of the mobile base with Algorithm 1 have been reported in the first row of Table 2. This algorithm fails to successfully localize the robot, because as it was predictable, the results exhibit a very large drift and the estimated trajectory is totally wrong after few meters of travel.
With reference to the same experimental path, the trajectory estimated by Algorithm 2 is more accurate with respect to that estimated by Algorithm 1. Algorithm 2 successfully removes the integration error present in the odometry. The goodness of the estimated trajectory is quantified by the numerical values of the estimation errors in correspondence of the markers. These values are reported in the second row of Table 2.
The experimental results obtained by Algorithm 3 are relatively close to those of Algorithm 2. The improvement introduced by Algorithm 3 can be evaluated looking at the numerical values reported in the third row of Table 2.

	Markers						
	Mk1	**Mk2**	**Mk3**	**Mk4**	**Mk5**	**Mk6**	**stop**
Algorithm 1	0.014	0.143	0.690	4.760	1.868	3.770	6.572
Algorithm 2	0.012	0.041	0.042	0.164	0.142	0.049	0.187
Algorithm 3	0.012	0.037	0.035	0.150	0.106	0.030	0.161

Table 2. Estimation errors (in meters) in correspondence of the marker configurations (distance between the marker and the corresponding estimated configuration).

3.5 Comments

The performed experimental tests show that the simple odometric localization is not satisfactory, making it necessary the introduction of another internal sensor. A fiber optic gyroscope showed to be a key tool for obtaining a significant improvement in the accuracy of the estimated trajectory. Algorithm 2 is very similar to Algorithm 1, the only difference is that Algorithm 2 exploits the gyroscopic measures. This is enough to produce a huge improvement of the estimated trajectory, thus confirming the validity of Equations (6), (7) provided that an accurate estimate of the robot orientation is available.
Algorithm 3 uses the same measures of Algorithm 2 but operates in the stochastic framework of the Kalman filtering theory. The novelty of the proposed EKF is that its formulation explicitly includes both the information carried by the model of the robot and the information carried by the observations. This introduces a further improvement with respect to Algorithm 2 and a very high degree of accuracy in the estimated trajectory is achieved. The main merit of Algorithm 3 is that it operates in a state-space form where sensor and model uncertainties are intrinsically taken into account. This makes the estimator more robust with respect to possible uncertain physical parameters and/or not exactly known initial conditions. Taking also into account its modest computational burden, Algorithm 3 appears to be the most appealing among the three localization procedures here proposed.

4. Absolute approaches for mobile robot localization

The purpose of this section is to propose and to experimentally evaluate a localization algorithm based on a measure apparatus composed of a set of internal and external sensors of a different nature and characterized by a highly different degree of accuracy. The sensor equipment includes odometric, gyroscopic and laser measures.
The main technical novelty of this section is the integration in a stochastic framework of

the new set of measures. Both the information carried by the kinematic model of the robot and that carried by the dynamic equations of the odometry are exploited. The nearly singular filtering problem arising from the very high accuracy of angular measure has been explicitly taken into account. An exteroceptive laser sensor is integrated for reducing the continuous growth in the integrated error affecting any relative localization algorithm, such as the Algorithm 3.

4.1 Algorithm 4

The algorithm operates in a stochastic framework as Algorithm 3, and is based on the ascertainment that the angular measure $y_\theta(t_k)$ provided by the FOG is much accurate than the other measures. This gives rise to a nearly singular filtering problem which can be solved by a lower order non singular Extended Kalman Filter, as described in subsection 3.3. The EKF is defined on the basis of a state equation derived from (1)–(3) and of a measure equation containing the incremental measures of the encoders $y_v(t_k)$ and the distance measures $y_{d_i^j}(t_k)$, $i=1,2,\ldots,n_s$, provided by the laser scanner from the P^j plane, $j\in\{1,2,\ldots,n_p\}$. The angular measure of the gyroscope $y_\theta(t_k)$ is assumed coincident to the third component $\theta(k)$ of the state vector $X(k)$.

Let $Z(k)$ be the measurement vector at time instant kT. Its dimension is not constant, depending on the number of sensory measures that are actually used at each time instant. The measure vector $Z(k)$ is composed by two subvectors $Z_1(k)=\left[z_1(k),z_2(k)\right]^T$ and $Z_2(k)=\left[z_3(k),z_4(k),\cdots,z_{2+n_s}(k)\right]^T$, where the elements of $Z_1(k)$ are: $z_1(k)\equiv y_v(k)$, $z_2(k)\equiv y_\theta(k)$, where $y_v(k)$ is the measure related to the increments provided by the encoders through equations (9) and (13), $y_\theta(k)$ is the angular measure provided by the FOG. The elements of $Z_2(k)$ are: $z_{2+i}(k)=d_i^j(k)+\eta_i(k)$, $i=1,2,\ldots,n_s$, $j\in\{1,2,\ldots,n_p\}$, with $d_i^j(k)$ given by (18) and $\eta_i(\cdot)\sim N(0,\sigma_i^2)$. The environment map provides the information needed to detect which is the plane P^j in front of the laser.

The observation noise $V(k)=\left[\eta_v(k),\eta_\theta(k),\eta_1(k),\ldots,\eta_{n_s}(k)\right]^T$, is a white sequence $\sim N(0,R)$ where $R:=\text{block diag}\left[R_1,R_2\right]$, with $R_1:=\text{diag}\left[\sigma_v^2,\sigma_\theta^2\right]$ and $R_2:=\text{diag}\left[\sigma_1^2,\sigma_2^2,\cdots,\sigma_{n_s}^2\right]$. The diagonal form of R follows by the independence of the encoder, FOG and laser scanner measures.

The components of the extended state vector $\bar{X}(k)$ and the last n_s components of vector $Z(k)$ are related by a non linear measure equation which depends on the environment geometric parameter vector $\prod$. The dimension $n_s(k)$ is not constant, depending on the number of laser scanner measures that are actually used at each time, this number depends on enviroment and robot configuration.

Linearization of the measure equation relating $Z_2(k)$ and $\bar{X}(k)$ about the current estimate

of $\bar{X}(k)$ results in:

$$\bar{Z}_2(k) = C_2(k)\,\bar{X}(k) + V_2(k) \tag{41}$$

where $V_2(k) = \left[\eta_1(k), \eta_2(k), \ldots, \eta_{n_s}(k)\right]$ is a white noise sequence $\sim N(0, R_2)$ and

$$C_2(k) := \left[c_1(k)^T, c_2(k)^T, \cdots, c_{n_s(k)}(k)^T\right]^T \tag{42}$$

with

$$c_i(k) = \frac{P_v^j}{\cos\theta_i^j}\left[-\cos P_n^j, -\sin P_n^j, 0, 0\right],\ i = 1,2,\ldots,n_s(k),\ j \in \{1,2,\ldots,n_p\} \tag{43}$$

and

$$\theta_i^j = P_n^j - \theta_g - \theta_i + \frac{\pi}{2} \tag{44}$$

Equations (33), (40) and (41) represent the linearized, discretized state-space form to which the classical EKF algorithm has been applied.

4.2 Laser scanner readings selection

To reduce the probability of an inadequate interpretation of erroneous sensor data, a method is introduced to deal with the undesired interferences produced by the presence of unknown obstacles on the environment or by incertitude on the sensor readings. Notice that for the problem handled here both the above events are equally distributed. A simple and efficient way to perform this preliminary measure selection is to compare the actual sensor readings with their expected values. Measures are discharged if the difference exceeds a time-varying threshold. This is here done in the following way: at each step, for each measure $z_{2+i}(k)$ of the laser scanner, the residual $\gamma_i(k) = z_{2+i}(k) - d_i^j(k)$ represents the difference between the actual sensor measure $z_{2+i}(k)$ and its expected value d_i^j, $i = 1,2,\ldots,n_s(k)$, $j = 1,2,\ldots,n_p$, which is computed by (18) on the basis of the current estimate of the vector state $X(k)$. As $\gamma_i(k) \sim N(0, s_i(k))$, the current value $z_{2+i}(k)$ is accepted if $|\gamma_i(k)| \le 2\sqrt{s_i(k)}$ (Jetto et al., 1999). Namely, the variable threshold is chosen as two times the standard deviation of the innovation process.

4.3 Experimental results

The experimental tests have been performed in an indoor environment using the same TGR Explorer powered wheelchair (TGR Bologna, 2000), described in Section 3.4.
The laser scanner measures have been acquired by the SICK LMS mod. 200 installed on the vehicle. The main characteristics of the LMS are reported in Table 3.

Aperture Angle	3.14 *rad*
Angular Resolution	0.0175/ 0.0088/ 0.0044 *rad*
Response Time	0.013/ 0.026/ 0.053 *s*
Resolution	0.010 *m*

Systematic Error	$\pm 0.015\ m$
Statistic Error (1 Sigma)	$0.005\ m$
Laser Class	1
Max. Distance	$80\ m$
Transfer Rate	9.6/ 19.2/ 38.4/ 500 *kBaud*

Table 3. Laser.

A characterization study of the Sick LMS 200 laser scanner has been performed as proposed in (Ye & Borenstein, 2002). Different experiments have been carried out to analyze the effects of data transfer rate, drift, optical properties of the target surfaces and incidence angle of the laser beam. Based on empirical data a mathematical model of the scanner errors has been obtained. This model has been used as a calibration function to reduce measurement errors.
The TGR Explorer powered wheelchair with data acquisition system for FOG sensor, incremental encoders, sonar sensors and laser scanner is shown in Fig. 5.

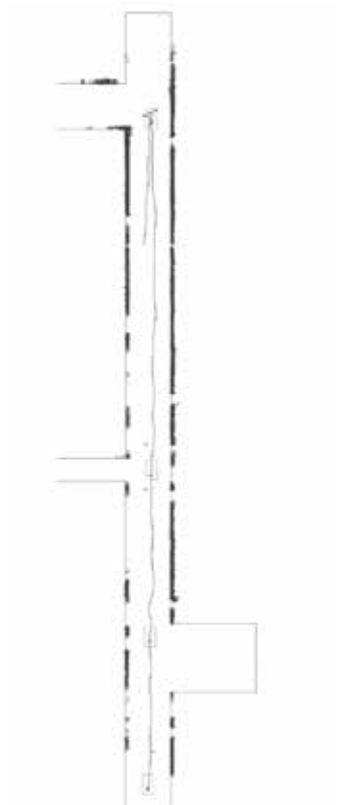


Fig. 6. Sample of the estimated trajectory. The dots are the actually used laser scanner measures.

A significative reduction of the wrong readings produced by the presence of unknown obstacles has been realized by the selection of the laser scanner measures using the procedure described in the previous subsection .

Different experiments have been performed making the mobile base track short and relatively long and closed trajectories. Fig. 6 illustrates a sample of the obtained results; the dots in the figure, are the actually used laser scanner measures. In the indoor environment of our Department, represented by a suitable set of planes orthogonal to the plane XY of the inertial system, a trajectory of 118 m length, characterized by orientation changes, has been imposed by the user interface. The starting and final positions have been measured, while six markers specify different middle positions; this permits to compute the distance and angle errors between the marker and the corresponding estimated configuration.

In these tests, the performances of Algorithm 4 have been compared with those ones of the Algorithm 3, which is the most appealing among the three relative procedures here analyzed. Table 4 summarizes the distance and angle errors between the marker and the corresponding configurations estimated by the two algorithms.

		Markers						
		Mk1	**Mk2**	**Mk3**	**Mk4**	**Mk5**	**Mk6**	**stop**
Alg 3	Error	0.1392	0.095	0.2553	0.1226	0.2004	0.0301	0.3595
	$\Delta\theta$	0.49	0.11	0.85	0.58	1.39	0.84	2.66
Alg 4	Error	0.0156	0.0899	0.0659	0.1788	0.0261	0.0601	0.0951
	$\Delta\theta$	0.59	0.05	0.45	0.07	0.72	0.12	1.55

Table 4. Estimation errors (in meters) in correspondence of the marker configurations (distance between the marker and the corresponding estimated configuration) and corresponding angular errors (in degrees).

Other significant sets of experiments have been performed inside a room, considering a short trajectory of 20 m characterized by different orientation changes (see Fig. 7).

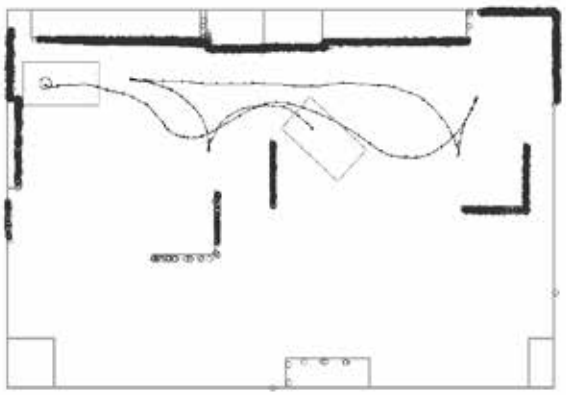


Fig. 7. Sample of the estimated trajectory inside the room, where dots indicate the laser measures.

The room has been modelled very carefully, permitting a precise evaluation of the distance and angle errors between the final position and the corresponding configuration estimated by the Algorithm 4; Table 5 resumes these results.

		final position
Alg 4	error	0.0061
	$\Delta\theta$	0.27

Table 5. Estimation distance errors (in meters) and corresponding angular errors (in degrees).

In order to investigate further the efficiency of the developed Algorithm 4 and to evaluate its correction performances, it has been imposed a wrong initial position (see Table 6 and Fig. 8).

	error of initial position	**error of final position**
error	0.2236	0.0152
$\Delta\theta$	1.5	0.73

Table 6. Distance (in meters) and angle (in degrees) errors introduced on the initial position and corresponding errors on the final position.

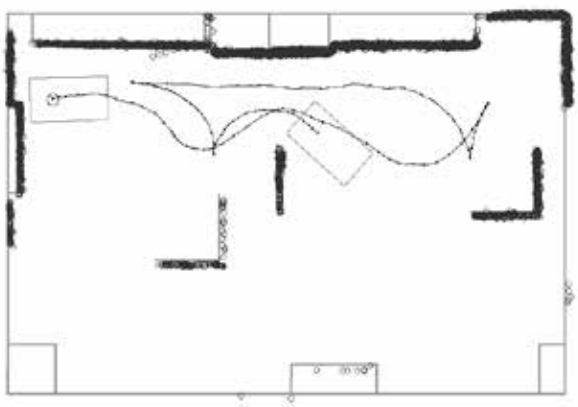


Fig. 8. Estimated trajectory with a wrong initial positioning.

As a result, it has been seen that the Algorithm 4 is able to correct possible errors on the initial positioning, as confirmed by the results reported in Table 6.

4.4 Comments

As shown by the developed experimental tests (see Table 4), Algorithm 4 permits to obtain a much more reliable and accurate positioning than that one obtained by Algorithm 3. Note that estimation errors on the final position of the Algorithm 3 are due to the angle drift introduced by the gyroscope.

Additionally, Algorithm 4 improves the positioning accuracy in spite of a wrong initial positioning. Table 6 shows as the possible errors introduced by a wrong initial pose, have been efficiently corrected by the Extended Kalman Filter.

5. Concluding remarks

This chapter has presented a concise look at the problems and methods relative to the mobile robot localization. Both the relative and absolute approaches have been discussed.
Relative localization has the main advantage of using a sensor equipment which is totally self-contained in the robot. It is relatively simple to be used and guarantees a high data rate. The main drawback is that the localization errors may considerably grow over time.
The three corresponding algorithms which have been proposed only use odometric and gyroscopic measures. The experimental tests relative to Algorithm 1 show that the incremental errors of the encoder readings heavily affect the orientation estimate, thus reducing the applicability of the algorithm to short trajectories. A significant improvement is introduced by Algorithm 2 where the odometric measures are integrated with the angular measures provided by a gyroscope.
Algorithm 3 uses the same measures of Algorithm 2 but operates in a stochastic framework. The localization problem is formulated as a state estimation problem and a very accurate estimate of the robot localization is obtained through a suitably defined EKF. A further notable improvement is provided by the fusion of the internal measures with absolute laser measures. This is clearly evidenced by Algorithm 4 where an EKF is again used.
A novelty of the EKF algorithms used here is that the relative state-space forms include all the available information, namely both the information carried by the vehicle dynamics and by the sensor readings. The appealing features of this approach are:

- The possibility of collecting all the available information and uncertainties of a different kind in the compact form of a meaningful state-space representation,
- The recursive structure of the solution,
- The modest computational effort.

Other previous, significant experimental tests have been performed at our Department using sonar measures instead of laser readings (Bonci et al., 2004; Ippoliti et al., 2004). Table 7 reports a comparison of the results obtained with Algorithm 3, Algorithm 4, and the algorithm (Algorithm 4(S)) based on an EKF fusing together odometric, gyroscopic and sonar measures. The comparative evaluation refers to the same relatively long trajectory used for Algorithm 4.

	Alg 3	**Alg 4**	**Alg 4(S)**
error	0.8079	0.0971	0.1408
$\Delta\theta$	2. 4637	0.7449	1. 4324

Table 7. Estimation errors (in meters) in correspondence of the final vehicle configuration (distance between the actual and the corresponding estimated configuration) and corresponding angular errors (in degrees).

Table 7 evidences that in spite of a higher cost with respect to the sonar system, the localization procedure based on odometric, inertial and laser measures does really seem to be an effective tool to deal with the mobile robot localization problem.
A very interesting and still open research field is the Simultaneous Localization and Map Building (SLAM) problem. It consists in defining a map of the unknown environment and simultaneously using this map to estimate the absolute location of the vehicle. An efficient solution of this problem appears to be of a dominant importance because it would definitely confer autonomy to the vehicle. The SLAM problem has been deeply investigated in (Leonard et al., 1990; Levitt & Lawton, 1990; Cox, 1991; Barshan & Durrant-Whyte, 1995; Kobayashi et al., 1995; Thrun et al., 1998; Sukkarieh et al., 1999; Roumeliotis & Bekey, 2000;

Antoniali & Orialo, 2001; Castellanos et al., 2001; Dissanayake et al., 2001a; Dissanayake et al., 2001b; Zunino & Christensen, 2001; Guivant et al., 2002; Williams et al., 2002; Zalama et al., 2002; Rekleitis et al., 2003)). The algorithms described in this chapter, represent a solid basis of theoretical background and practical experience from which the numerous questions raised by SLAM problem can be solved, as confirmed by the preliminary results in (Ippoliti et al., 2004; Ippoliti et al., 2005).

6. References

Anderson, B.D.O. & Moore, J.B. (1979). *Optimal Filtering*. Prentice-Hall, Inc, Englewood Cliffs

Antoniali, F.M. & Oriolo, G. (2001). Robot localization in nonsmooth environments: experiments with a new filtering technique, *Proceedings of the IEEE International Conference on Robotics and Automation (2001 ICRA)*, Vol. 2, pp. 1591–1596

Arras, K.O.; Tomatis, N. & Siegwart, R. (2000). Multisensor on-the-fly localization using laser and vision, *Proceedings of the 2000 IEEE/RSJ International Conference onIntelligent Robots and Systems, (IROS 2000)*, Vol. 1, pp. 462–467

Barshan, B. & Durrant-Whyte, H.F. (1995). Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 3, pp. 328–342

Bemporad, A.; Di Marco, M. & Tesi, A. (2000). Sonar-based wall-following control of mobile robots. *Journal of dynamic systems, measurement, and control*, Vol. 122, pp. 226–230

Bonci, A.; Ippoliti, G.; Jetto, L.; Leo, T. & Longhi, S. (2004). Methods and algorithms for sensor data fusion aimed at improving the autonomy of a mobile robot. In: *Advances in Control of Articulated and Mobile Robots*, B. Siciliano, A. De Luca , C. Melchiorri, and G. Casalino, Eds. Berlin, Heidelberg, Germany: STAR (Springer Tracts in Advanced Robotics), Springer-Verlag, Vol. 10, pp. 191–222.

Borenstein, J. & Feng, L. (1996). Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transaction on Robotics and Automation*, Vol. 12, No. 6, pp. 869–880

Borenstein, J.; Everett, H. R.; Feng, L. & Wehe, D. (1997). Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, Vol. 14, No. 4, pp. 231–249

Castellanos, J.A.; Neira, J. & Tardós, J.D. (2001). Multisensor fusion for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 6, pp. 908–914

Chung, H.; Ojeda, L. & Borenstein, J. (2001). Accurate mobile robot dead-reckoning with a precision-calibrated fiber-optic gyroscope. *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 1, pp. 80–84

Cox, I. (1991). Blanche – an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, pp. 193–204

De Cecco, M. (2003). Sensor fusion of inertial-odometric navigation as a function of the actual manoeuvres of autonomous guided vehicles. *Measurement Science and Tecnology*, Vol. 14, pp. 643–653

Dissanayake, M.; Newman, P.; Clark, S.; Durrant-Whyte, H.F. & Csorba, M. (2001a). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 3, pp. 229–241

Dissanayake, G.; Sukkarieh, S.; Nebot, E. & Durrant-Whyte, H.F. (2001b). The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. *IEEE Transactions on Robotics and Automation*, Vol. 17, pp. 731–747

Durrant-Whyte, H.F. (1988). Sensor models and multisensor integration. *International Journal of Robotics Research*, Vol. 7, No. 9, pp. 97–113

Fioretti, S.; Leo, T. & Longhi, S. (2000). A navigation system for increasing the autonomy and the security of powered wheelchairs. *IEEE Transactions on Rehabilitation Engineering*, Vol. 8, No. 4, pp. 490–498

Garcia, G.; Bonnifait, Ph. & Le Corre, J.-F. (1995). A multisensor fusion localization algorithm with self-calibration of error-corrupted mobile robot parameters, *Proceedings of the International Conference in Advanced Robotics, ICAR'95*, pp. 391–397, Barcelona, Spain

Gu, J.; Meng, M.; Cook, A. & Liu, P.X. (2002). Sensor fusion in mobile robot: some perspectives, *Proceedings of the 4th World Congress on Intelligent Control and Automation*, Vol. 2, pp. 1194–1199

Guivant, J.E.; Masson, F.R. & Nebot, E.M. (2002). Simultaneous localization and map building using natural features and absolute information. In: *Robotics and Autonomous Systems*, pp. 79–90

Ippoliti, G.; Jetto, L.; La Manna, A. & Longhi, S. (2004). Consistent on line estimation of environment features aimed at enhancing the efficiency of the localization procedure for a powered wheelchair, *Proceedings of the Tenth International Symposium on Robotics with Applications - World Automation Congress (ISORA-WAC 2004)*, Seville, Spain

Ippoliti, G.; Jetto, L.; La Manna, A. & Longhi, S. (2005). Improving the robustness properties of robot localization procedures with respect to environment features uncertainties, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005)*, Barcelona, Spain

Jarvis, R.A. (1992). Autonomous robot localisation by sensor fusion, *Proceedings of the IEEE International Workshop on Emerging Technologies and Factory Automation*, pp. 446–450

Jetto, L.; Longhi, S. & Venturini, G. (1999). Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots. *IEEE Transactions on Robotics and Automation*, Vol. 15, pp. 219–229

Killian, K. (1994). Pointing grade fiber optic gyroscope, *Proceedings of the IEEE Symposium on Position Location and Navigation*, pp. 165–169, Las Vegas, NV,USA

Kobayashi, K.; Cheok, K.C.; Watanabe, K. & Munekata, F. (1995). Accurate global positioning via fuzzy logic Kalman filter-based sensor fusion technique, *Proceedings of the 1995 IEEE IECON 21st International Conference on Industrial Electronics, Control, and Instrumentation*, Vol. 2, pp. 1136–1141, Orlando, FL ,USA

Komoriya, K. & Oyama, E. (1994). Position estimation of a mobile robot using optical fiber gyroscope (ofg), *Proceedings of the 1994 International Conference on Intelligent Robots and Systems (IROS'94)*, pp. 143–149, Munich, Germany

Leonard, J.; Durrant-Whyte, H.F. & Cox, I. (1990). Dynamic map building for autonomous mobile robot, *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems (IROS '90)*, Vol. 1, pp. 89–96, Ibaraki, Japan

Levitt, T.S. & Lawton, D.T. (1990). Qualitative navigation for mobile robots. *Artificial Intelligence Journal*, Vol. 44, No. 3, pp. 305–360

Mar, J. & Leu, J.-H. (1996). Simulations of the positioning accuracy of integrated vehicular navigation systems. *IEE Proceedings - Radar, Sonar and Navigation*, Vol. 2, No. 143, pp. 121–128

Martinelli, A. (2002). The odometry error of a mobile robot with a synchronous drive system. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 3, pp. 399–405

Ojeda, L.; Chung, H. & Borenstein, J. (2000). Precision-calibration of fiber-optics gyroscopes for mobile robot navigation, *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pp. 2064–2069, San Francisco, CA, USA

Panzieri, S.; Pascucci, F. & Ulivi, G. (2002). An outdoor navigation system using GPS and inertial platform. *IEEE/ASME Transactions on Mechatronics*, Vol. 7, No. 2, pp. 134–142

Rekleitis, I.; Dudek, G. & Milios, E. (2003). Probabilistic cooperative localization and mapping in practice, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03)*, Vol. 2, pp. 1907 – 1912

Roumeliotis, S.I. & Bekey, G.A. (2000). Bayesian estimation and Kalman filtering: a unified framework for mobile robot localization, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, Vol. 3, pp. 2985–2992, San Francisco, CA, USA

Sukkarieh, S.; Nebot, E.M. & Durrant-Whyte, H.F. (1999). A high integrity IMU GPS navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation*, Vol. 15, pp. 572–578

Talluri, R. & Aggarwal, J.K. (1992). Position estimation for an autonomous mobile robot in an outdoor environment. *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 5, pp. 573–584

TGR Bologna (2000). *TGR Explorer*. Italy [Online]. Available on-line: http://www.tgr.it.

Thrun, S.; Fox, D. & Burgard, W. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning Autonom. Robots*, Vol. 31, pp. 29–53. Kluwer Academic Publisher, Boston

Wang, C.M. (1988). Localization estimation and uncertainty analysis for mobile robots, *Proceedings of the Int. Conf. on Robotics and Automation*, pp. 1230–1235

Williams, S.B.; Dissanayake, G. & Durrant-Whyte, H.F. (2002). An efficient approach to the simultaneous localization and mapping problem, *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pp. 406–411, Washington DC, USA

Ye, C. & Borenstein, J. (2002). Characterization of a 2d laser scanner for mobile robot obstacle negotiation, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, Vol. 3, pp. 2512 – 2518, Washington DC, USA

Yi, Z.; Khing, H. Y.; Seng, C.C. & Wei, Z.X. (2000). Multi-ultrasonic sensor fusion for mobile robots, *Proceedings of the IEEE Intelligent Vehicles Symposium*, Vol. 1, pp. 387–391, Dearborn, MI, USA

Zalama, E.; Candela, G.; Gómez, J. & Thrun, S. (2002). Concurrent mapping and localization for mobile robots with segmented local maps, *Proceedings of the 2002 IEEE/RSJ Conference on Intelligent Robots and Systems*, pp. 546–551, Lausanne, Switzerland

Zhu, R.; Zhang, Y. & Bao, Q. (2000). A novel intelligent strategy for improving measurement precision of FOG. *IEEE Transactions on Instrumentation and Measurement*, Vol. 49, No. 6, pp. 1183–1188

Zhuang, W. & Tranquilla, J. (1995). Modeling and analysis for the GPS pseudo-range observable. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 31, No. 2, pp. 739–751

Zunino, G. & Christensen, H.I. (2001). Simultaneous localization and mapping in domestic environments, *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2001)*, pp. 67–72

12

Composite Models for Mobile Robot Offline Path Planning

Ellips Masehian, M. R. Amin-Naseri
Tarbiat Modares University
Iran

1. Introduction

As new technological achievements take place in the robotic hardware field, an increased level of intelligence is required as well. The most fundamental intelligent task for a mobile robot is the ability to plan a valid path from its initial to terminal configurations while avoiding all obstacles located on its way.

The robot motion planning problem came into existence in early 70's and evolved to a vast and active research discipline as it is today. Numerous solution methods have been developed for robot motion planning since then, many of them being variations of a few general approaches: Roadmap, Cell Decomposition, Potential Fields, mathematical programming, and heuristic methods. Most classes of motion planning problems can be solved using these approaches, which are broadly surveyed in (Latombe, 1991), (Hwang & Ahuja, 1992), and (Choset et al., 2005).

This chapter introduces two new offline path planning models which are founded on the Roadmap and Potential Fields classic motion planning approaches. These approaches have their unique characteristics and strategies for solving motion planning problems. In fact, each one has its own advantage that excels others in certain aspects. For instance, the Visibility Graph yields the shortest path; but its computational time exceeds other methods. Or, while the Voronoi Diagram plans the safest path and is easy to calculate in 2D, it often produces overly lengthy paths, and yields poor results in higher space dimensions. On the other hand, Potential Fields are easy to compute and are suitable for high dimensional problems, but they suffer from the local minima problem, and the oscillating paths generated near narrow passages of configuration space reduce their efficiency. A brief review on these underlying methods is given in this section.

In order to benefit from the strong aspects of these classic path planning methods and compensate their drawbacks, a policy of combining these basic approaches into single architectures is adopted. In devising the new planners it is intended to aggregate the superiorities of these methods and work out efficient and reliable composite algorithms for robot motion planning.

1.1 Roadmap Methods

The Roadmap approach involves retracting or reducing the robot's free Configuration space (C_{free}) onto a network of one-dimensional lines (i.e. a graph). Motion planning is then reduced to a graph-searching problem. At first, two paths are constructed from the start and

goal positions to the roadmap, one for each. Then a path is planned between these points on the roadmap. The correctness of the solution strongly depends on the connectivity of the roadmap representing the entire C-space. If the roadmap does not represent the entire C-space, a solution path may be missed.

The *Visibility Graph* is the collection of lines in the free space that connects a feature of an object to that of another. In its principal form, these features are vertices of polygonal obstacles, and there are $O(n^2)$ edges in the visibility graph, which can be constructed in $O(n^2)$ time and space in 2D, where n is the number of features (Hwang & Ahuja, 1992).

The *Reduced Generalized Visibility Graph* can be constructed in $O(n^3)$ time and its search performed in $O(n^2)$ time. The shortest path can be found in $O(n^2 \log n)$ time using the A* algorithm with the Euclidean distance to the goal as the heuristic function (Latombe, 1991). Works such as (Oommen et al., 1987) and (Yeung & Bekey, 1987) have employed this approach for path planning.

The *Voronoi Diagram* is defined as the set of points that are equidistant from two or more object features. Let the set of input features be denoted as $s_1, s_2, \ldots, s_n$. For each feature s_i, a distance function $D_i(x) = \text{Dist}(s_i, x)$ is defined. Then the *Voronoi region* of s_i is the set $V_i = \{x \mid D_i(x) \leq D_j(x)\ \forall\ j \neq i\}$. The Voronoi diagram partitions the space into such regions. When the edges of convex obstacles are taken as features and the C-space is in $\Re^2$, The Voronoi diagram of the C_{free} consists of a finite collection of straight line segments and parabolic curve segments, referred to as *Medial Axis*, or more often, *Generalized Voronoi Diagram* (*GVD*).

In an $\Re^k$ space, the *k-equidistant face* is the set of points equidistant to objects $C_1, \ldots, C_k$ such that each point is closer to objects $C_1, \ldots, C_k$ than any other object. The *Generalized Voronoi Graph* (*GVG*) is the collection of m-equidistant faces (i.e. generalized Voronoi edges) and m+1-equidistant faces (i.e. generalized Voronoi vertices, or, *meet points*). The GVD is the locus of points equidistant to *two* obstacles, whereas the GVG is the locus of points equidistant to m obstacles. Therefore, in $\Re^m$, the GVD is m–1-dimensional, and the GVG, 1-dimensional. In planar case, the GVG and GVD coincide (Aurenhammer & Klein, 2000).

The Voronoi diagram is attractive in two respects: there are only $O(n)$ edges in the Voronoi diagram, and it can be efficiently constructed in $\Omega(n \log n)$ time, where n is the number of features. The Voronoi diagram can be searched for the shortest path in $O(n^2)$ time by using the Dijkstra's method. Another advantage of Voronoi method is the fact that the object's initial connectedness is directly transferred to the diagram (Hwang & Ahuja, 1992). In (Canny, 1985) and (Choset & Burdick, 2000) the Voronoi diagram is used for planning robot paths.

For higher-dimensional spaces than 2D, both the Visibility graph and the Voronoi diagram have higher complexities, and it is not obvious what to select for the features. For example, the Voronoi diagram among polyhedra is a collection of 2D faces, which is not a 1D roadmap (Agarwal et al., 1998).

The *Silhouette* method has been developed at early stages of the motion planning discipline, and is complex to implement. Its time complexity is in $O(2^m)$, where m is the dimension of the C-space, and is mostly used in theoretical algorithms analyzing complexity, rather than developing practical algorithms. A path found from the silhouette curves makes the robot slide along obstacle boundaries (Canny, 1988).

Probabilistic Roadmaps use randomization to construct a graph in C-space. Roadmap nodes correspond to collision-free configurations of the robot. Two nodes are connected by an

edge if a path between the two corresponding configurations can be found by a 'local planning' method. Queries are processed by connecting the initial and goal configurations to the roadmap, and then finding a path in the roadmap between these two connection points (Kavraki et al., 1996).

1.2 The Potential Fields Method

A robot in Potential Fields method is treated as a point represented in configuration space, and as a particle under the influence of an artificial potential field U whose local variations reflect the 'structure' of the free space (Khatib, 1986). In order to make the robot attracted toward its goal configuration while being repulsed from the obstacles, U is constructed as the sum of two elementary potential functions; attractive potential associated with the goal configuration q_{goal} and repulsive potential associated with the C-obstacle region. Motion planning is performed in an iterative fashion. At each iteration, the artificial force induced by the potential function at the current configuration is regarded as the most appropriate direction of motion, and path planning proceeds along this direction by some increment.
The most serious problem with the Potential Fields method is the presence of local minima caused by the interaction of attractive and repulsive potentials, which results in a cyclic motion. The routine method for getting free is to take a random step outwards the minimum well. Other drawbacks are (Koren & Borenstein, 1991):

- No passage between closely spaced obstacles.
- Oscillations in the presence of obstacles or in narrow passages.
- Non-smooth movements of the robot when trying to extricate from a local minimum.
- Overlapping of different obstacles' repulsive potentials when they are adjacent to each other.
- Difficulty in defining potential parameters properly.

Nevertheless, the Potential Fields method remains as a major path-planning approach, especially when high degrees of freedoms are involved. This approach has improved later through a number of works such as (Sato, 1993), (Brook & Khatib, 1999) and (Alvarez et al., 2003) to overcome the problem of getting trapped in local minima.
The next sections of this chapter introduce two new composite models for robot path planning, called *V-P Hybrid*, and *V-V-P Compound*. They are apt to cover the shortcomings of their original methods and are efficient both in time complexity and path quality. Although originally devised for two-dimensional workspaces, they can be extended straightforwardly to 3D spaces. Experiments have shown their strength in solving a wide variety of problems.

2. The V-P Hybrid Model

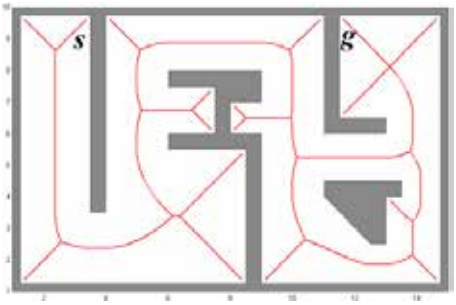
In this section we present a new algorithm, called *V-P Hybrid*, where the concepts of Voronoi diagram and Potential fields are combined to integrate the advantages of each. In this approach, the initial path planning problem is decomposed to a number of smaller tasks, having intermediate milestones as temporary start and goal points. Through this iterative process the global path is incrementally constructed.
For the path planning task, a number of assumptions are made: (*i*) the map of workspace is known a priori, (*ii*) the obstacles are static, and (*iii*) the robot is considered a point. For real world applications, the latter assumption can be attained by expanding the obstacles using the Minkowski Set Difference method.

The algorithm's major steps are:
(1) *Preprocessing Phase*; consisted of constructing a *Pruned Generalized Voronoi Graph* of the workspace, and then applying a Potential Field to it. This operation yields a network of Voronoi valleys (Sec. 2.1).
(2) *Search Phase*; consisted of implementing a bidirectional steepest descent – mildest ascent search method to navigate through the network of Voronoi valleys. The search phase is designed to progressively build up a start-to-goal path (Sec. 2.2).
Before explaining the details of the composite model, a mathematical representation of some variables is given:

- n : Total number of obstacles' vertices.
- s : The Start configuration.
- g : The Goal configuration.
- $G = (V, E)$: The Generalized Voronoi Graph (GVG) of the C_{free} with the set of vertices (nodes) $V(G)$ and edges $E(G)$.
- $E(v, w)$: The edge connecting vertices v and w , $\forall\, v, w \in V(G)$.
- $N(v) = \{w \mid E(v, w) \neq \varnothing\}$: Neighboring vertices of the vertex v.
- $E(v)$: The set of all edges at vertex v.
- $d(v) = |E(v)|$: The degree of vertex v, equal to the number of passing edges.

2.1 Preprocessing Phase

The V-P Hybrid model starts solving the problem by constructing the Generalized Voronoi Graph (GVG) of the C-space. The Start and Goal configurations are then connected to the main Voronoi graph through shortest lines which are also included in the diagram. Fig. 1(a) provides an example of GVG.



Procedure **PRUNE**(***G, s, g***)
  P={ $v \mid v \in V(G) \setminus \{s, g\}, d(v) = 1$ }
  if ($P = \varnothing$) then Stop
  $V(G) \leftarrow V(G) \setminus P$
  $E(G) \leftarrow E(G) \setminus E(v, N(v)), v \in P$
  PRUNE(G, s, g)
end

Fig. 1. (a) Generalized Voronoi Graph (GVG). (b) Algorithm for pruning the GVG.

The main reason for incorporating the Voronoi concept in the Hybrid algorithm is its property of lying on the maximum clearance from the obstacles. This property helps the robot to navigate at a safe distance from obstacles, making it less prone to be trapped in local minimum wells.
The next step is to exclude redundant or unpromising edges from the GVG. This is done through the *pruning* operation, where the Voronoi edges which either touch obstacle boundaries or have vertices with a degree ($d(v)$) equal to 1 are iteratively truncated. The pruning procedure is explained in Fig. 1(b). Also, the result of this operation performed on

the example of Fig. 1(a) is portrayed in Fig. 2. The resulting subgraph is called *Pruned Generalized Voronoi Graph*, or simply PGVG.
Note that the hypersensitivity of Voronoi diagram to minor inaccuracies in workspace definition which may lead to redundant edges (as in the lower-right disjoint obstacle in Fig. 2(a)) is resolved after running the pruning procedure.
The pruning operation is an important stage in the Hybrid algorithm since it truncates all paths toward collision with obstacles and dead-end traps, and therefore reduces the search space drastically. The resulting graph is a 'lean' network of interconnected Voronoi vertices, including the Start and Goal nodes.

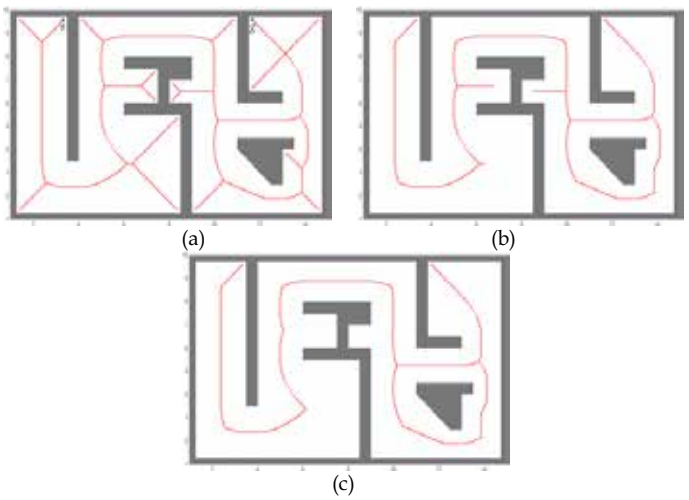


Fig. 2. The construction of the Pruned Generalized Voronoi Graph in two iterations.

The last step of the preprocessing phase is constructing a potential field for guiding the robot toward its goal. Unlike the conventional Potential Fields concept where there are two kinds of attractive and repulsive potentials associated with goal and obstacles respectively, the V-P hybrid algorithm makes use of only attractive potentials, related to the goal and the PGVG. By this, we avoid some known problems of the standard Potential Fields method concerning the calculation of repulsive forces for each obstacle and their integration into a single function, which usually gives rise to complexities due to overlapping and parameter setting (Koren & Bornstein, 1991). This reduces the computational time and memory significantly. Moreover, the problem of narrow corridors, where most Potential Field algorithms give way is fixed in this version.
To apply these potentials, we graduate the configuration space into a grid of fine-enough resolution. For every grid point (x_i, y_i) the potential can then be numerically calculated in a very short time.
As mentioned, the path planning process is decomposed into intermediate stages. So, each stage has its own temporary goal point, g_{temp}. The attractive potential of the goal is exerted through a paraboloid function with a nadir at the temporary goal by:

$$U_{g_{temp}}(x,y) = \xi\left[\left(x - x_{g_{temp}}\right)^2 + \left(y - y_{g_{temp}}\right)^2\right], \tag{1}$$

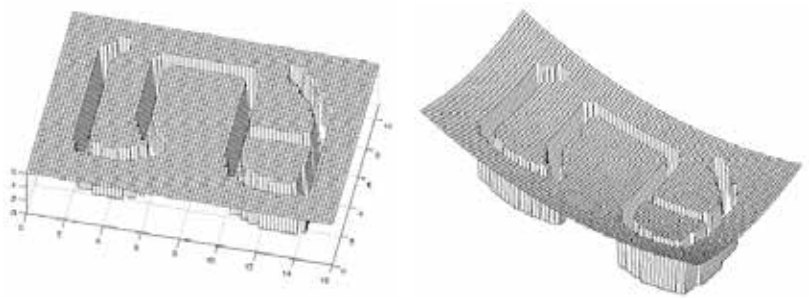
where ξ is a scaling factor.
The next attractive potential applies to the PGVG. Because the GVG keeps a safe distance from obstacles the robot will hardly collide with them. Besides, since we prune the GVG such that all Voronoi edges toward obstacles (mainly leading to dead-ends) are eliminated from the graph, the possibility of the robot to get trapped in local minima reduces drastically. So we try to "encourage" the robot to move along the edges of PGVG. This is done by associating an attractive potential with the points on PGVG, which generates a network of deep "valleys" located at the maximum distance from obstacles, with a width of one gridpoint (Fig. 3(a)). The (virtual) robot will safely navigate at the bottom of PGVG valleys. The following function gives the desired result, in which *s* is the depth of valley:

$$U_{PGVG}(x_i, y_i) = \begin{cases} -s & \text{if } (x_i, y_i) \in \text{PGVG} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The U_{PGVG} field is calculated only once and remains constant till the end of the path planning. Instead, the attractive potential of the (temporary) goal is calculated at each iteration and is added to the U_{PGVG} to yield the total potential used for the Search phase by

$$U_{Total} = U_g + U_{PGVG} \tag{3}$$

The resulting manifold is depicted in Fig. 3(b) for a typical temporary goal point. Note that due to the numerical nature of the model, working with these complex functions is extremely easy, and just a simple addition of corresponding grid values is sufficient.



(a) (b)

Fig. 3. (a) PGVG potential valleys for the sample problem (here the width of canals are intentionally aggrandized for a better view). (b) the total potential manifold as the sum of PGVG valleys and goal attractive potentials.

Since the PGVG is a connected roadmap, a path connecting the Start and Goal points (which are located at the bottom of PGVG valleys) certainly exists.
This combination of potentials provides a straightforward and guaranteed attraction from start to goal point. The potential associated with the goal absorbs every point to itself, as the gradient direction at every configuration points to the goal. Note that repulsive potentials are not calculated and consequently all the problems related to them are avoided.

The parameters of the functions such as the valley depth and concavity of the paraboloid should be selected carefully to make sure that the robot will not "escape" from valleys and surmount the obstacles, which are implicitly defined by their high potentials compared to the deeper valleys.
It should be mentioned that the obtained total potential field may still have local minima (e.g. the V-shaped channel left to the center in Fig. 3(b)), but due to the applied search method they are resolved.

2.2 Search Phase

To search within the potential manifold, a bidirectional approach is adopted. First, two trajectory sets, *Traj*(*s*) and *Traj*(*g*), spanned from the Start (*s*) and Goal (*g*) points respectively, are initialized to keep the track of planned paths. Then through an iterative process, the PGVG valleys are being navigated alternately by *Traj*(*s*) and *Traj*(*g*). At each iteration first *Traj*(*s*) and then *Traj*(*g*) extend toward the endpoints of each other. Whenever a trajectory reaches a junction (i.e. a Voronoi vertex) it stops extending more, and the expansion is shifted to the other trajectory. The trajectories meet on the halfway and are concatenated into a single start-to-goal trajectory.
The bidirectional nature of the search requires that for each iteration, the PGVG manifold be numerically added to a paraboloid centered on an intermediate goal point. For instance, when extending *Traj*(*s*), the temporary goal is to reach the endpoint of *Traj*(*g*), which is located on a junction of PGVG valleys.
To maintain the movement of the robot in each iteration, the method of descent search is employed, which is the simplest and fastest searching method in numerical contexts.
The neighborhood of each cell is defined to be 2-neighbors, that is, the points lying in the range of (x±1, y±1) for the point (x, y). The number of neighbors of a cell is thus $3^2 - 1 = 8$. For a k-dimensional space, it would be $3^k - 1$.
The searching begins at Start point, with examining all its neighboring gridpoints. The descent search selects a neighboring cell with the lowest potential among all neighbors as the next configuration. The simple steepest descent method, however, is prone to stop at a local minimum. To cope with this problem, taking ascending steps (or, "hill climbing") is devised for exiting from local minimums. The amount of ascension is kept minimal. Therefore, the concept used here is a "steepest descent, mildest ascent" motion. The hill climbing movement is comparable to the random walk in the randomized planning (Barraquand et al., 1992). Upon reaching a junction, the next edge to navigate is the one having the lowest potential value at that point.
In order to prevent the robot from looping (i.e. infinitely fluctuating between two neighboring cells), we assign to all visited grid cells a relatively higher potential, but still lower than the potentials of points not on the PGVG. Therefore, the robot will not return immediately to a local minimum after it has been once there, simply because it is not a local minimum anymore. The height to which a visited point is elevated is suggested to be less than 1/3 of the valley depth (Fig. 4). This will allow traversing an edge for three times (as in correcting a wrong route) without diverting from the PGVG edges.
The process of the steepest descent - mildest ascent search applied to the example in Fig. 2(c) is shown in Figs. 5(a)-(d). Fig. 5(b) shows iteration 1, navigating from Start toward Goal. The *Traj*(*s*) stops at the first encountered junction (or Voronoi vertex). Fig. 5(c) shows iteration 1, navigating from the Goal point towards the temporary goal, which is now the endpoint of *Traj*(*s*). The *Traj*(*g*) stops at the first encountered junction, which becomes the new

temporary goal. Fig. 5(d) illustrates iteration 2, navigating from endpoint of *Traj*(*s*) toward the temporary goal. The two trajectories *Traj*(*s*) and *Traj*(*g*) are now get connected, and the Search phase is completed. Note the changes in depth of valleys as they are being filled.

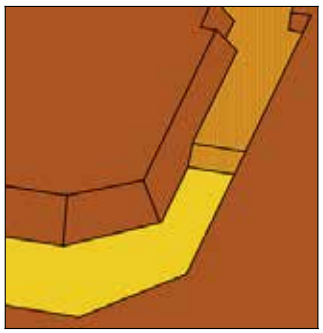


Fig. 4. Valley-filling operation: the potential valley is being filled as the trajectory proceeds.

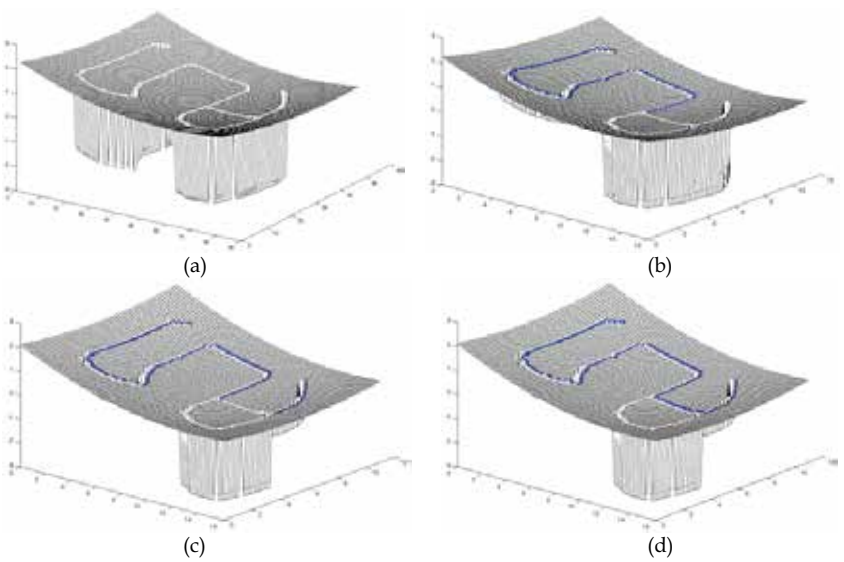


Fig. 5. The process of searching in the V-P Hybrid model is completed in two iterations.

2.3 Experiments

In order to test and evaluate the V-P Hybrid algorithm, 20 problems with obstacles differing in number and shape (including convex, concave, and maze-like problems) were designed and solved by three different methods: the V-P Hybrid, the classical Potential Fields, and the A* Search. Experiments were run on a PC with a 1.4 GHz processor using MATLAB.

Table 1 shows the average values of path lengths (in equal units), CPU time (in seconds) and the number of evaluated grid points computed for the test problems via different approaches. The average length of optimal paths was 27.46 units.

Parameter \ Model	Potential Field Algorithm	A* Search Algorithm	V-P Hybrid Algorithm
Path Length	33.49	30.67	33.62
Search CPU Time	1.0375	19.40	0.0715
Total Examined Grid points	2513	2664.2	331.8

Table 1. Experimental results.

An advantage of the V-P Hybrid algorithm over the classical Potential Fields method is its completeness. While the Potential Fields approach is not guaranteed to generate a valid path (Latombe, 1991), the V-P algorithm is *exact*, i.e. it finds a path if one exists. Since the Goal should be connected to the PGVG at the Preprocessing phase, the algorithm will report any failure in this stage, and so is *complete*.

The V-P Hybrid algorithm has also resolved a number of problems inherent in the conventional Potential Fields method. The local minimum problem is settled by implementing the steepest descent – mildest ascent search method and utilizing the PGVG. Problems due to obstacle potentials and narrow passages are totally fixed.

The Voronoi diagram-Potential Field Hybrid algorithm averagely spent much less time for searching the C-space than the Potential Field method (around 15 times faster). Also the number of examined grid-points was reduced about 7.5 times for the Hybrid algorithm. We ascribe these results to the efficient abstraction of workspace due to the pruning procedure where most local minimum wells are excluded from the search space. The number of Voronoi vertices is also reduced effectively. The pruning procedure together with the fast searching of Voronoi valleys made the V-P model successful in solving complex and labyrinthine, maze-like workspaces. In sparse environments the Potential Fields found slightly shorter paths, but for maze-like problems the Hybrid algorithm outperformed.

The time complexity of A* search is $O(n^2)$ (Latombe, 1991). A* is complete and optimal, but its space complexity is still prohibitive. The A* search employs a heuristic function for estimating the cost to reach the goal. For our experimentation a Euclidean straight-line distance was used as the heuristic. The Hybrid algorithm searched the grid space very much faster than A* search (270 times), examining around 8 times less points than it. This is because of the lower time complexity order of the Hybrid method compared to the $O(n^2)$ of A*. However, the quality of the path generated by A* is better than the Hybrid model by %10. The Hybrid algorithm also outperforms the Dijkstra's algorithm which has an $O(n^2)$ time complexity. The time complexity of the V-P Hybrid algorithm is discussed below.

2.4 Time Complexity Analysis

For a time complexity analysis of the V-P Hybrid algorithm, its two phases must be analyzed separately. Time complexities of constructing and pruning the Voronoi graph, as

well as the potential field calculation determine the computational burden of the Preprocessing phase. To evaluate this, we first need to study the problem's size. The following two lemmas deal with this issue:

Lemma 1. The Voronoi diagram has $O(n)$ many edges and vertices, in which n is the number of Voronoi sites.

Lemma 2. The average number of edges in the boundary of a Voronoi region is bounded by 6.

Proofs to these lemmas are provided in (Aurenhammer & Klein, 2000). The proofs are originally developed for the case of points or convex objects taken as Voronoi sites. However, since due to the pruning procedure any non-convex obstacle is located in a unique connected Voronoi region, the above lemmas hold true for non-convex cases as well.

The direct consequence of the Lemma 1 is that the Hybrid algorithm must perform $O(n)$ neighborhood checks for pruning the Voronoi Graph. Therefore, considering that the construction of the Generalized Voronoi Diagram takes $O(n\log n)$ time, we conclude that the Pruned Generalized Voronoi Diagram is built in $O(n\log n)$ time.

For the potential field calculation, since we do not need to calculate the potential values for *all* gridpoints, save for those located on the PGVG, it is essential to have an estimate for the number of gridpoints on the PGVG.

Assuming that after graduating the C-space the PGVG edges are divided into small intervals of size λ, each PGVG edge with vertices v and w will have grid points equal to $e = \frac{|E(v,w)|}{\lambda}$. Considering the $O(n)$ edges of the C-space, the number of all grid points would be $O(e \times n) \equiv O(n)$, which also gives the complexity of potential field calculation.

For obtaining an average-space complexity, the average length of the PGVG edges should be computed. Let m be the total number of configuration gridpoints, o the number of configuration gridpoints occupied by obstacles, and b the number of obstacles. Then the average number of C-points around an obstacle (Voronoi region) is $(m–o)/b$. Since the average number of edges around each obstacle is bounded by 6 (Lemma 2), we will assume that the typical shape of the region is hexagonal, with the surface area of $S = 3\sqrt{3}a^2/2$, where a is the edge of the hexagon (Fig. 6). By setting this surface area equal to the average number of C-points in a Voronoi region, we get

$$a = \frac{1}{3}\sqrt{\frac{2\sqrt{3}(m-o)}{b}} \cong 0.62\sqrt{\frac{(m-o)}{b}}. \tag{4}$$

Since $o < m$ in (4), we conclude that the average length of a Voronoi edge in terms of its number of gridpoints is in $O(\sqrt{m})$. This means that the number of points whose potentials are to be computed is in $O(\sqrt{m})$, where m is the total number of gridpoints.

The above space complexity can also be used for calculating the time complexity of the Search phase. Since only the gridpoints on the PGVG need to be searched, and the average number of these points is $O(\sqrt{m})$, the Search phase averagely will take $O(\sqrt{m})$ time to navigate the PGVG and accomplish the search. This result is superior to the conventional Potential Field's search which contains a neighborhood checking operation and is carried on in $O(m)$, m being the number of C-points (Latombe, 1991).

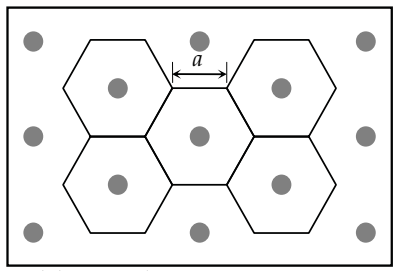


Fig. 6. A typical problem with hexagonal Voronoi regions.

To conclude, the Preprocessing phase of the algorithm takes $O(n\log n)$ time (n being the total number of obstacle vertices), which is due to construction of the GVG. The remaining components of the algorithm, i.e. the pruning, potential calculation, and potential search procedures all have linear or sub-linear time complexities. Since these components are executed sequentially, the most time-consuming operation will be bound to $O(n\log n)$ time, which is the total time complexity.

3. The V-V-P Compound Model

Since the paths generated by the V-P Hybrid model are a subset of the Generalized Voronoi Graph of the workspace, they have lengths identical to the ones generated by the Voronoi Diagram method. The Voronoi paths are longer than the optimal Visibility Graph-based paths, especially in sparse environments. Aiming to improve the quality of generated paths, another composite algorithm is proposed (Masehian & Amin-Naseri, 2004) where three methods of Voronoi Diagram, Visibility graph, and Potential Fields are integrated in a single architecture, called *V-V-P Compound* model.

The Compound model provides a parametric tradeoff between the safest and shortest paths and generally yields shorter paths than the Voronoi and Potential field methods, and faster than the Visibility graph. In the proposed model, positive attributes of these three path planning techniques have been combined in order to benefit from the advantages of each. To accomplish this, they are tailored and associated with a number of complementary procedures to generate a valid and high quality path. Hence, the Compound algorithm borrows its name, V-V-P, from these basic techniques, although the outcome is a new and different model as a whole.

An overview of the model is as follows: after constructing the PGVG, a network of broad freeways is developed through a new concept based on medial axis, named *MID*. A potential function is then assigned to the freeways to form an obstacle-free network of valleys. Afterwards we take advantage of a bidirectional search, where the Visibility Graph and Potential Field modules execute alternately from both Start and Goal configurations. A steepest descent – mildest ascent search technique is used for local planning and avoiding local minima. The assumptions on which the model is principally developed are the same as for the V-P Hybrid model; that is, the workspace is considered two-dimensional, and the map of workspace is known a priori. Similar to the Hybrid model, the Compound model has also two major stages: the Preprocessing phase and the Search phase. The Search phase contains two modules: *Visibility*, and *Potential Field*, which are executed alternately, as illustrated in Fig. 7.

The main differences between the V-V-P Compound and V-P Hybrid models are the width of the potential valleys and their filling technique. Additionally, the V-V-P model employs a Visibility module to obtain shorter paths than the V-P model. The description of algorithm's phases is presented in the next subsections.

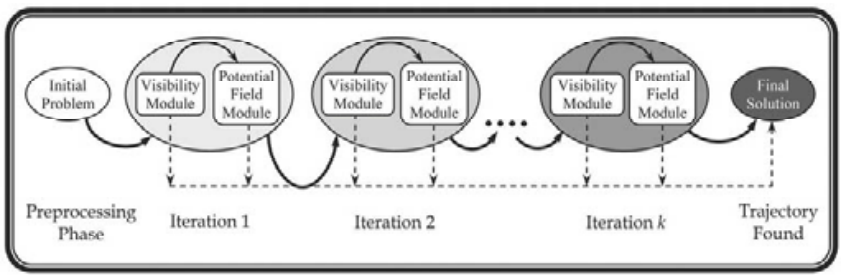


Fig. 7. The overall process of problem solving in the V-V-P path planning model. Each iteration in search phase is comprised of two sequentially executed modules, Visibility and Potential Field. The gradually darkening shades imply the completion of a solution.

3.1 Preprocessing Phase

This phase establishes an obstacle-free area for robot navigation. The main steps are:

P1) Constructing the PGVG of the workspace (as described in Sec. 2.1).

P2) Forming an obstacle-free C-space region based on PGVG points.

P3) Associating an attractive (negative) potential to that region. The result is an obstacle-free network of valleys as the robot's navigation area.

As noted in Sec. 1.1, the Generalized Voronoi Graph is also known as *Medial Axis* (*MA*). Voronoi diagram lies on the maximum clearance of objects. Although this property offers some advantages regarding to path safety, it makes the path longer, especially in workspaces where the obstacles are located quite far from each other. Besides, the generated path usually has sharp angles at Voronoi vertices, making it ineffective for robots with nonholonomic or rotational constraints.

In order to compensate these shortcomings, unlike the 1-pixel-wide valleys in the V-P model, a network of "wider" channels is built based on PGVG. These channels are "dilated" Voronoi edges that provide sufficient space for the robot to plan shorter paths and maneuver freely. Due to the varying sizes of inter-obstacle free spaces, the widths of these channels must vary from region to region.

For constructing this obstacle-free network of channels the *Maximal Inscribed Disc* (*MID*) concept is incorporated. First some definitions are presented:

A *Locally Maximal Disc* (*LMD*) of the point $x \in C_{free}$ is the set of points such that:

$$LMD(x) = \left\{ q \;\middle|\; \|x-q\| \le \mathrm{Min}\,\|x - \partial C_{free}\|,\ q \in C_{free} \right\}, \tag{5}$$

and denotes a disc centered at x and tangent to the nearest obstacle boundary (∂C_{free}).

The *Maximal Inscribed Disc* (*MID*) is defined as:

$$MID(x)=\left\{LMD(x)\middle|\, r_{LMD(x)}>r_{LMD(y)},\ x\in MA\wedge y\in N(x)\right\}, \tag{6}$$

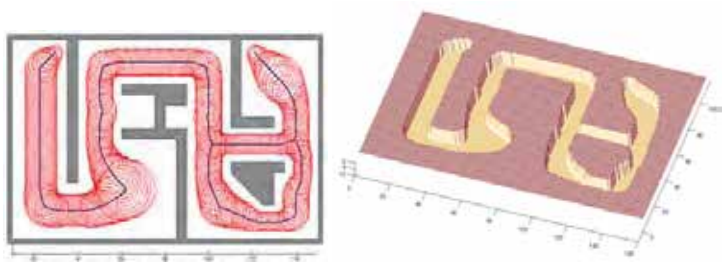
in which the $r_{LMD(x)}$ is the radius of the $LMD(x)$, and $N(x)$ is the neighborhood of x.
For the Compound model, only the radii of all *MID*s centered on PGVG points are calculated. Also, in order to maintain a safe distance from obstacle borders, *MID*s radii are multiplied by a lessening factor α (α ∈ [0, 1]), to produce α*MID*s defined as:

$$\alpha MID(x)=\left\{LMD(x)\middle|\, r_{LMD(x)}=\alpha\times r_{MID(x)},\ x\in MA, 0\le\alpha\le 1\right\} \tag{7}$$

All α*MID*s are integrated in a connected region called *Region*(α*MID*). The *Region*(α*MID*) of a C-space is the union of all α*MID*s centered on the medial axis:

$$Region(\alpha MID)=\left\{\bigcup_{\forall x\in MA}\alpha MID(x)\right\} \tag{8}$$

The *Region*(α*MID*) is obstacle-free and non-convex, and reflects the topology of the C_{free}. An interesting property of the α is that it offers a balance between the Roadmap and full C_{free} concepts. If we set α=0, the *Region*(α*MID*) will turn into the medial axis roadmap. For α =1, the region's borders will be tangent to obstacles. Based on experiments, we recommend α ∈ [0.5, 0.8].
The *Region*(α*MID*) for the workspace of Fig. 2(c) is calculated and depicted in Fig. 8(a). Fig. 8 also indicates the property of *Region*(α
MID) in smoothening the Voronoi roadmap's sharp corners and local irregularities.



(a) (b)

Fig. 8. (a) The *Region*(α*MID*) is comprised of α*MID*s centered on the medial axis. Here the α is set to 0.6. (b) Attractive potentials associated with the *Region*(α*MID*).

Similar to the Hybrid model, the Compound model also creates a network of navigable valleys. It assigns attractive potentials to the points lying in *Region*(α*MID*):

$$U(x_i,y_i)=\begin{cases}-s & \text{if } (x_i,y_i)\in Region(\alpha MID)\\ 0 & \text{otherwise.}\end{cases} \tag{9}$$

The preprocessing phase terminates with the construction of potential valleys.

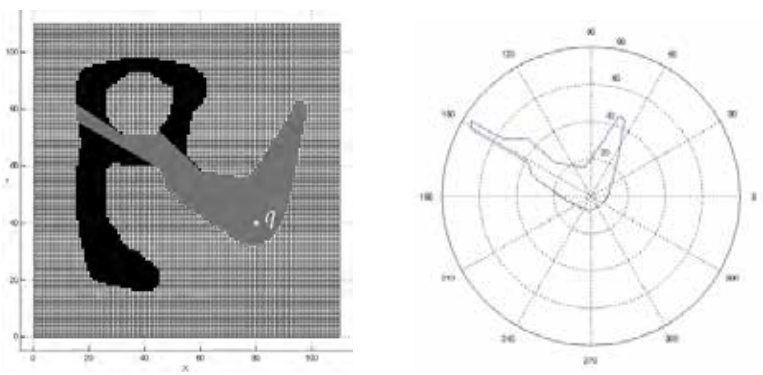
3.2 Search Phase

This phase is designed to progressively build up a Start-to-Goal path. The initial problem is decomposed to a number of smaller path planning tasks, having intermediate milestones as

temporary start and goal points. Through this iterative process the solution path is incrementally constructed, and the algorithm becomes capable to resolve more complex problems.
Similar to the V-P Hybrid, the global search process is performed bidirectionally. Again we initialize the two trajectories *Traj*(*s*) and *Traj*(*g*), and set *Traj*(*s*) = {*s*} and *Traj*(*g*) = {*g*} for the beginning.
The main modules included in this phase are *Visibility* and *Potential Field*, which are executed iteratively until the construction of the final path. The termination condition is satisfied when *Traj*(*s*) and *Traj*(*g*) are either being seen or get in touch with each other. We characterize 'being seen' as being able to draw a straight line in free space to connect the two trajectories' endpoints.
The following subsections describe the Visibility and Potential Field modules.

3.2.1 Visibility Module

Each iteration of the Search phase starts with a *Visibility scan* performed concurrently for both endpoints of *Traj*(*s*) and *Traj*(*g*). For this purpose, a "ray sweeping" technique is used to collect information about the surrounding valley borders and probably the opposite trajectory.
The aim of this procedure is to determine whether the opposite trajectory is visible from the current point or not. If it is visible, then the Search phase is over. If not, we have to find the boundary vertices as seen from the current point, as described below.
By applying a polar coordinate system with the origin defined on the vantage point (e.g. endpoint of *Traj*(*s*)), the radial Euclidean distances to valley borders (∂C_{free}) are calculated for [0, 2π] and integrated in an array (i.e. *Visibility Polygon*). Fig. 9(a) shows the C_{free} valleys and the point (*q*) considered for visibility scan in a sample problem. Fig. 9(b) shows the distance of that point from its surroundings.



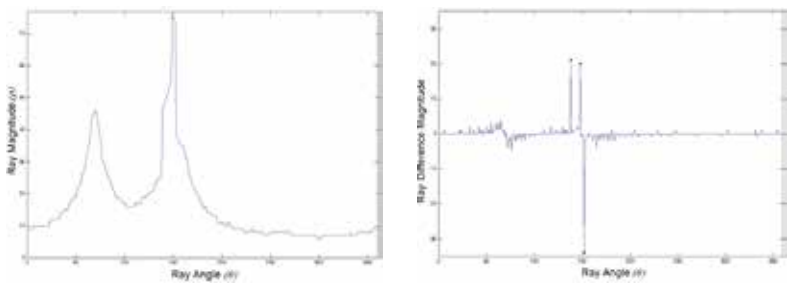
(a) (b)

Fig. 9. (a) Visible configurations (visibility polygon) as a result of a visibility scan performed for the point *q*. (b) The polar representation of radial distances (i.e. ray magnitudes) of the point *q* from C_{free} boundary (∂C_{free}).
Subsequent to the calculation of distances (ρ) between the vantage point and ∂C_{free} for any angle ($\theta \in [0, 2\pi]$), this data is mapped into Cartesian coordinates (Fig. 10(a)).

Since the C_{free} boundary generally has a complex geometrical shape and lacks definite vertices as in polygonal objects, we take advantage of the ray sweeping data to determine the boundary points being tangent to any ray emanated from the vision source point. A ray is *tangent* to ∂C_{free} if in the neighborhood of their contact point the interior of C_{free} lies entirely on a single side of it.

In order to find the tangent rays and their touching boundary points, we apply a difference function for successive adjacent rays. We define the *Ray Difference* variables as $\Delta\rho_\theta = \rho_{\theta+1} - \rho_\theta$ for $\theta \in [0, 2\pi]$ and collect them in an array plotted in Fig. 10(b). By applying a notch filter, the peaks of the Ray Difference array are determined. These peaks imply abrupt and large differences in successive ray magnitudes and therefore indicate the points where sweeping rays leave (positive peaks) or meet (negative peaks) a convex contour on ∂C_{free}, based on anticlockwise rotation of rays.

The boundary points corresponding to the tangent rays are treated as boundary vertices visible from the vantage point, q. These points are called *Critical points* and form the set $R(q)$ (see step S1(d)). The tangent rays and critical points are shown in Fig. 11.



(a) (b)

Fig. 10. (a) The Cartesian representation for the ray magnitudes of Fig. 9. (b) Magnitude difference of sweeping rays for successive angles. The three peaks show tangent rays.

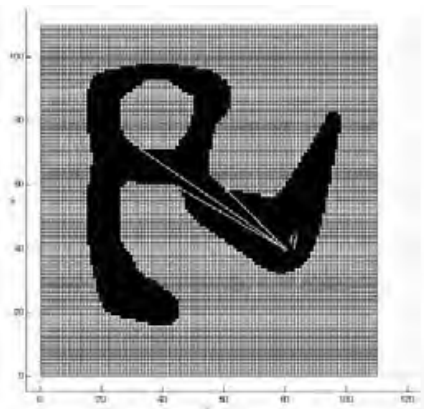
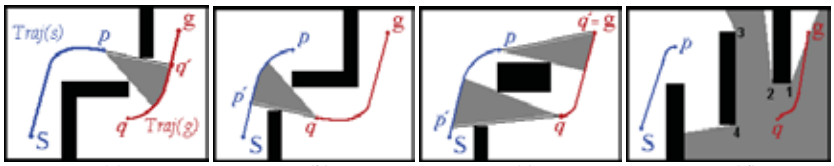


Fig. 11. The tangent rays and their corresponding boundary vertices (critical points).

By concurrently implementing the visibility scan for both ends of *Traj*(*s*) and *Traj*(*g*), we discover that either there exists a line which connects the two trajectories (and lies entirely in C_{free}), or none of them is within the scope of the other's endpoint. If the first case holds then the search phase terminates. For the latter case, critical points of the two sets *R*(*p*) and *R*(*q*) are calculated and matched to find the closest pair, one point from each. These points determine the two positions which the two trajectories must extend toward.
The following steps are taken for the Visibility module:

S1) Performing *Visibility scan*. The scan is concurrently implemented for the endpoints of both *Traj*(*s*) and *Traj*(*g*).

Suppose that the visibility scan operation is performed from *p* and *q*, the endpoints of *Traj*(*s*) and *Traj*(*g*), respectively. Consequently, four incidences may occur (Fig. 12):



(a) (b) (c) (d)

Fig. 12. Four different combinations of *Traj*(*s*) and *Traj*(*g*) in visibility scan. The visibility envelope is shown in grey.

(*a*) A subset of points in *Traj*(*g*) is visible from *p*, but no point from *Traj*(*s*) is visible from *q* (Fig. 12(a)). In this case, by a straight line, connect *p* to a visible point in *Traj*(*g*), say *q*', which is nearest to the Goal (i.e. has the smallest ordinal rank in *Traj*(*g*) among the visible points), and truncate all elements in *Traj*(*g*) located after *q*'. Note that the Goal point might be visible itself, which in that case point *p* is directly connected to the *g* (Fig. 12(c)).

(*b*) A subset of points in *Traj*(*s*) is visible from *q*, but no point from *Traj*(*g*) is visible from *p* (Fig. 12(b)). This is the reverse of the previous case, so act similarly, but swap *p* and *q*, and also *Traj*(*s*) and *Traj*(*g*).

(*c*) Subsets of points in both *Traj*(*g*) and *Traj*(*s*) are visible from *p* and *q*, respectively (Fig. 12(c)). In this case, define the following criterion *C* as:

$$\begin{aligned} C &= \mathrm{Min}\left\{\left|spq'g\right|, \left|gqp's\right|\right\} \\ &= \mathrm{Min}\left\{\left|Traj(s)\right| + \left\|p - q'\right\| + \left|q' \in Traj(g)\right|, \left|Traj(g)\right| + \left\|q - p'\right\| + \left|p' \in Traj(s)\right|\right\} \end{aligned} \tag{10}$$

where $|Traj(s)|$ means the cardinality (or length) of *Traj*(*s*), $\|p - q'\|$ is the Euclidean distance of *p* and *q*', and $|q' \in Traj(g)|$ indicates the ordinal position of *q*' in *Traj*(*g*) (i.e. the distance of *q*' to *g* via the *Traj*(*g*)). Among *pq*' and *qp*', the line providing the minimum value for the above criterion will be selected to connect *Traj*(*s*) and *Traj*(*g*). Again truncate the elements of the trajectory located after the connection point *p*' or *q*', according to the drawn line.

(*d*) If none of the *Traj*(*s*) and *Traj*(*g*) are visible to each other's endpoints, then for both *p* and *q*, determine those rays that are tangent to visible C_{obs} boundary. Note that this boundary is at a safe distance from actual obstacles' edges. The intersection of these rays and the free space's boundary produces two sets of *Critical Points*, *R*(*p*)

and $R(q)$. Fig. 12(d) shows the result of visibility scan from q, which consequently renders 4 visible obstacle vertices in $R(q) = \{1, 2, 3, 4\}$.
Now among all combinations of the elements of $R(p)$ and $R(q)$, select the closest x and y pair meeting the following condition:

$$\{(x, y) \mid \forall\, x, u \in R(p); y, v \in R(q); \left\| x - y \right\| \leq \left\| u - v \right\|\}, \tag{11}$$

where $\left\| \bullet \right\|$ shows Euclidean distance. The total number of combinations to be evaluated is $|R(p)| \times |R(q)|$, where $|\bullet|$ is the cardinality of sets. This operation determines the mutually best points that $Traj(s)$ and $Traj(g)$ must extend toward via two straight lines.

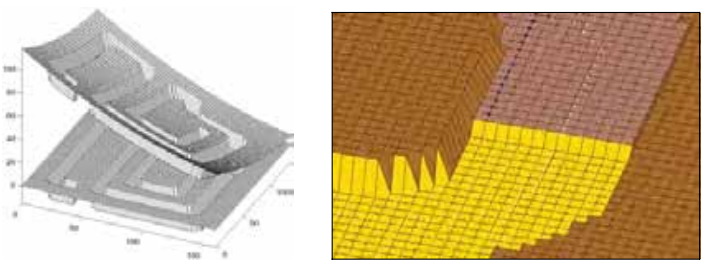
S2) Map the line segment(s) found in step S1 to the configuration space grid. Through a fine-enough discretizing operation, new points are added to $Traj(s)$ and/or $Traj(g)$.

If any of the cases (*a*), (*b*), or (*c*) in step S1 holds, then terminate the Search phase and go to step S10 (Sec. 3.2.2). For the case (*d*) continue with the next step.

S3) Since all the points in $Traj(s)$ and $Traj(g)$ lie on the bottom of roadmap valleys, in order to mark the valleys as traversed, increase the potentials of trajectory points and their surroundings to 'fill' the width of valleys (Sec. 3.2.2). This is an effective operation for preventing the planner from searching the C_{free} exhaustively.

3.2.2 Potential Field Module

The bidirectional nature of the V-V-P algorithm requires that for each iteration, the valley potentials manifold be numerically added to a paraboloid with a nadir on a temporary goal point (see step S4). For instance, when extending $Traj(s)$, the temporary goal is the endpoint of $Traj(g)$, and vice versa. To apply the paraboloid potential, we graduate the configuration space in a fine-enough resolution, then assigning every grid cell as (x_i, y_i), the potential is calculated numerically. Fig. 13(a) shows the Potential Field manifold superimposed on the 'flat' valley potentials manifold.
As soon as new points are appended to the trajectories, the navigated valleys must be distinguished by 'elevating' their potentials in order to prevent the robot to re-traverse them later (Fig. 13(b)).

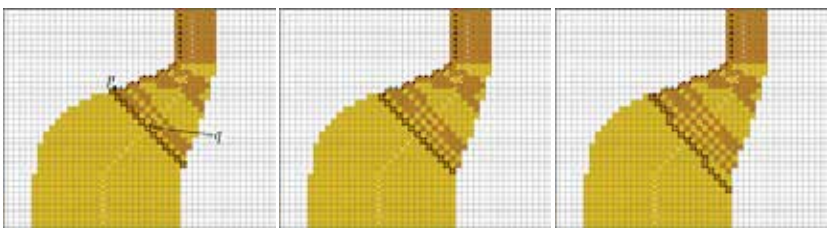


(a) (b)

Fig. 13. (a) The Potential Field manifold (upper object) is constructed by numerically adding a paraboloid function defined in (1) to the valley potentials manifold (lower object). (b) A scene from an intermediate iteration in potential search. Trajectory points are shown black and the medial axis points are in white.

The valley filling technique is somehow a "micro-visibility" process; it marks the neighboring configurations as 'seen', and excludes them from the search space. This process is analogous to walking in a long corridor while trying to get out by reaching an open door or a junction. Naturally one does not consider the tiles across the corridor and near his feet as promising cells leading to a desired destination. Rather, he deems those points as traversed (though physically not indeed), and continues his wall-following motion. This is done in filling technique by 'elevating' the potentials of those cells, making them less attractive. Since in a steepest descent context the robot occupies the cell with the least potential value across the valley, the filling procedure does not affect the path length adversely.

The filling procedure is applied immediately after a new point is appended to a trajectory. So it is performed in a layer-by-layer manner. Suppose that a point p is just being added to an existing trajectory array (Fig. 14(a)). In order to 'mark' and elevate the potentials of visited cells across the C_{free} valley, we must find a line passing from p and perpendicular to the local direction of the channel. To do this, the point p must be connected to its nearest point q on the medial axis (skeleton) of the valley. By interpolation and extrapolation, the cells along this line are found and increased in potential. The amount of this increase is proposed to be about 1/3 of the valley depth (i.e. s in (9)). Fig. 14 shows three consecutive iterations of filling operation.



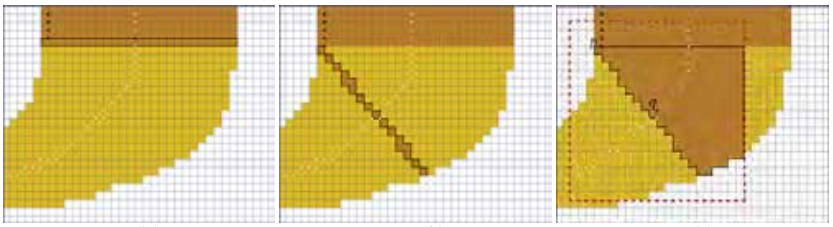
(a) (b) (c)

Fig. 14. Three iterations from the valley filling process. As new points (black dots) are appended to the trajectory, the cells across the channel are elevated in potential, so that the planner is encouraged to move along the valley's main direction. Points on the medial axis are shown white, except for the point q which is nearest to trajectory's endpoint p (shown in black). The elevated rack is highlighted in each iteration.

For a better understanding of the role of this process, imagine that an attractive potential (i.e. a local minimum) is located in the upper-end of the narrow channel in Fig. 14(a). Then according to the *steepest descent* search, the trajectory points should move towards it, which is of course hopeless. However, the elevated barrier created in each iteration blocks this motion, and forces the planner to take a *mildest ascent* step and run off the fatal situation.

For channels of uniform width this method fills the cells thoroughly and compactly, but it may cause porosities in curved and bent valleys, or leave unfilled areas behind, as in Figs. 14 or 15(b). The case in Fig. 15(b) arises from the fact that for two successive trajectory points their respective nearest medial axis points are not adjacent. Although this does not cause a serious problem most of the time, we will present a variation to this procedure to overcome such conditions:

First a square (or rectangular) frame with a symmetrical center on the medial point q is defined (the dashed line in Fig. 15(c)). This frame is partitioned into two hyper-planes by the connecting line pq. The hyper-plane that contains the penultimate trajectory point is therefore the 'backward' region which may contain some unfilled cells. Then, the potentials of the cells confined within the frame and valley border are elevated. The magnitude of this frame can be set such that all the unfilled cells can be covered. However, a size equal to the valley width in that point suffices. The still unfilled area at the right of Fig. 15(c) will not cause any problem since it is far from trajectory points.



(a) (b) (c)

Fig. 15. An unfilled area is originated from the fact that for two successive trajectory points, their respective nearest medial axis points are not adjacent. To resolve this problem, a frame is defined around the medial point q (drawn by dashed line), and the unfilled area confined within this frame is elevated in potential.

The implemented valley filling routine provides some advantages for the model:

(1) It reduces the potential searching time significantly by discarding the configurations in C_{free} which have normal vectors pointing toward a local minimum, and so obviates the random or 'Brownian' movements.

(2) This technique enables the planner to perform a 'hill climbing' operation for coping with the attraction of a nearby local minimum, and as such, is a subtle way to avoid exhaustively filling up dead-end or saddle point regions and the consequent path smoothing operations (Barraquand et al., 1992).

For more clarification, suppose that the planner incrementally builds up a search tree and adopts a 'best-first' strategy to find the goal point. This task becomes time-consuming when the tree has many branches. The valley filling process curtails most of the non-promising branches and directs the planner along an effective branch leading to another valley. In other words, this technique converts a 'breadth-first' or 'best-first' search into a 'depth-first' search.

Experiments showed that the valley filling process aids the robot considerably especially in departing from deep local minimum wells.

Now the Potential Field module is executed according to the following steps. It is applied in two directions: first the $Traj(s)$ is extended (steps S4 to S6), then $Traj(g)$ is stretched out (step S7 to S9).

S4) Setting the endpoint of $Traj(g)$ as the temporary goal (g_{temp}), construct an attractive field by the paraboloid function introduced in (1). Then add this potential to the potential of $Region(\alpha MID)$ calculated in step P3 (Sec. 3.1).

S5) Now the steepest descent – mildest ascent search is performed with setting the endpoint of $Traj(s)$ as temporary start and the endpoint of $Traj(g)$ as temporary goal

point. This step contains a gradient search for selecting the next gridcell to proceed. New points are appended to *Traj*(*s*). Also, in order to provide a mechanism for escaping from local minima, perform the valley filling procedure.

S6) Repeat the step S5 until one of the following situations take place:
- **(*a*)** If before the occurrence of case (*b*) below, the endpoint of *Traj*(*s*) meets any point in opposite trajectory *Traj*(*g*), the search phase is completed. First truncate the elements of *Traj*(*g*) located after the connection point, then go to step S10.
- **(*b*)** The gridcell wavefront distance between the endpoint of *Traj*(*s*) and the free space boundary, ∂C_{free}, exceeds a certain limit, i.e. $|\text{END}(Traj(s)) - \partial C_{free}| > d$. Through experimentations $d = 3$ was found appropriate.

The steepest descent search for *Traj*(*s*) is now terminated and the searching process is shifted to steps S7 to S9, where *Traj*(*g*) is being extended towards *Traj*(*s*).

S7) This step is similar to step S4, except that the paraboloid which is added to the *Region*(α*MID*) valleys has a minimum on the endpoint of *Traj*(*s*).

S8) Setting the endpoints of *Traj*(*g*) and *Traj*(*s*) as temporary start and goal points respectively, perform a steepest descent – mildest ascent search, as well as the valley filling procedure, as described in step S5.

S9) Repeat the step S8 until either of the following cases happen:
- **(*a*)** If before the occurrence of case (*b*) below, the endpoint of *Traj*(*g*) meets any point in *Traj*(*s*), the search phase is completed. Truncate the elements of *Traj*(*s*) located after the connection point, then go to step S10.
- **(*b*)** If the gridcell wavefront distance between the endpoint of *Traj*(*g*) and the ∂C_{free} exceeds a certain limit, i.e. $|\text{END}(Traj(g)) - \partial C_{free}| > 3$, terminate the Potential Field module and start the next iteration from step S1, the Visibility module.

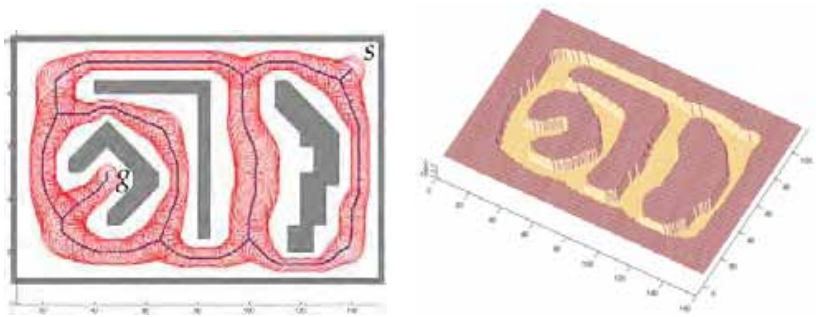
S10) Reverse the order of elements in *Traj*(*g*) and concatenate it to the endpoint of *Traj*(*s*). As a result, a single start-to-goal trajectory is achieved which is the final output of the V-V-P algorithm.

3.3 An Example

Now the algorithm's path planning technique is demonstrated through solving a problem illustrated in Fig. 16(a).

After preparing the valley potentials (Fig. 16(b)), the Search phase is accomplished in 3 iterations. The bidirectional progression of trajectories is clearly shown in Figs. 17(a)-(c). The C_{free} region is light-colored, and the 'filled' area has a darker shade. Fig. 17(a) indicates the development of *Traj*(*s*) (upper-right), and *Traj*(*g*) (lower-left) trajectories in iteration 1, by first performing a visibility scan, then a Potential Field search. The visibility scan matches with case S1(*d*), where none of the two trajectories is in the scope of another. Hence, 6 possible pairs of critical points ((2 for *g*) × (3 for *s*)) are evaluated and the closest pair is selected as the destination of trajectories. The filling procedure is then implemented for the drawn lines (darker area in C_{free}) according to step S3.

The Potential Field module now starts with performing a steepest descent – mildest ascent search from the endpoint of *Traj*(*s*) toward the endpoint of *Traj*(*g*), the temporary goal. This requires a superimposition of a paraboloid function with a minimum on END(*Traj*(*g*)) on the 'flat' potential manifold in Fig. 16(b) (as described in step S4). This search generates points directed to the temporary goal, elevates the potentials across the current valley, and stops after a few repetitions upon detaching enough from the ∂C_{free} (case S6(*b*)). These points are appended to *Traj*(*s*).



(a) (b)

Fig. 16. (a) The PGVG and *Region*(α*MID*) (Step P2). (b) Obstacle-free network of valley potentials (Step P3).

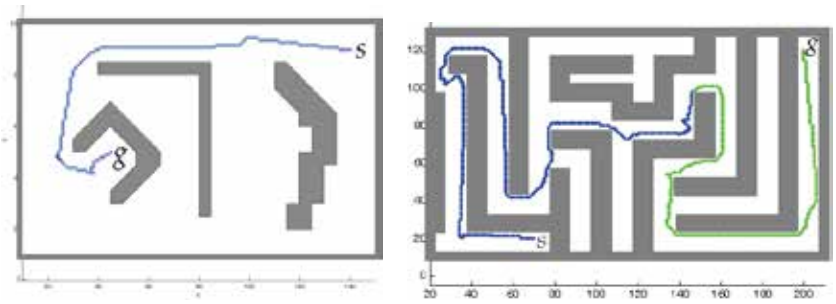
(a) (b) (c)

Fig. 17. The first, second and third iterations of the Search phase. The black lines show tangent rays for visibility scan, and white points are generated by potential search.

The same operation is carried on from END(*Traj*(*g*)) to the new endpoint of *Traj*(*s*), which now includes recently added potential search points. Note that in Fig. 17(a), due to the filling operations executed before and during the Potential Field module, the steepest descent search does not fill the nearby minimum well, and thus avoids entrapment in the local minimum around the Goal point. Rather, it utilizes the *mildest ascent* concept, and exhibits a hill climbing behavior. This case shows the importance and effectiveness of the filling procedure, which helps the planner substantially through the whole process. Fig 17(b) illustrates the second iteration, which is performed in the same fashion as the first iteration. Note the wall-following function of the potential module before detachment from C_{free} border.

Fig. 17(c) displays the case S1(*c*) occurred in the third iteration, where both trajectories are being seen by each other's endpoints. By applying the criterion (10) it becomes evident that the endpoint of *Traj*(*s*) must be connected to a visible point in *Traj*(*g*) closest to *g*. The remaining points to the end of *Traj*(*g*) are truncated afterwards. Eventually the reversely-ordered *Traj*(*g*) is concatenated to the *Traj*(*s*) and yields the final path from Start to Goal (Fig. 18(a)).

Another example is presented in Fig. 18(b) to display the shape of the generated path for a maze-like problem. The meeting point of the approaching trajectories is shown by a color contrast. The search took 7 seconds and five iterations.



(a) (b)

Fig. 18. (a) The final start-to-goal path. (b) Maze-like problem solved by the V-V-P algorithm.

3.4 Time Complexity

As discussed in the Sec. 2.4, the time complexity of constructing the PGVG is $O(n\log n)$. The time required for establishing the *Region*(□*MID*) depends on the total length of PGVG edges, which is in $O(n)$. The time required to calculate the valley potentials is constant for each gridpoint lying in *Region*(□*MID*). Hence, the total time complexity for the preprocessing phase is in the order of $O(n\log n)$.

The Search phase has the Visibility and Potential Field modules which are executed for k iterations. In the worst-case, k is bounded by half the number of all edges, which is in $O(n/2) \equiv O(n)$. During the Search phase, the visibility radial sweep operation has constant time complexity and depends on the number of radial rays. The number of potential valleys is in $O(n)$, which is affected by the $O(n)$ number of Voronoi edges, n being the total number of obstacle vertices. The time complexity for the Potential Field searching operation is $O(m)$ in the total number of gridpoints (m), and is independent of the number and shape of the obstacles (Latombe 1991). Therefore, the time complexity of the Search phase is in the order of $O(m)$.

3.5 Comparisons

In order to compare the V-V-P model with the Visibility Graph, Voronoi diagram, and Potential Fields methods, we solved the 20 problems mentioned in Sec. 2.3 by these methods and calculated the lengths of produced paths. Path lengths were normalized via a uniform scale to set up a proper benchmark for comparison. The value of □ in V-V-P algorithm was set to 0.7. The Preprocessing phase of the Compound model took about 9 seconds averagely, and the Search phase finished within 6 seconds on average. The experiments were run in MATLAB using a 1.4 GHz processor. A comparison of path lengths, as well as time complexities of the preprocessing and search procedures of all tested methods is provided in Table 2.

The results show that the V-V-P Compound takes advantage of the superiorities of its parent methods; that is, low construction time from the GVG, low search time from the PF, and short paths from the VG. It provides an effective balance between computational speed and path quality. The extent of this tradeoff is determined by selecting different values for $\alpha \in (0, 1)$, after which the V-V-P method assumes the properties of either the Visibility, or Voronoi methods, or an intermediate state.

Path planning method	Preprocessing Time complexity	Searching		Relative path length
		Time complexity	Search method	
Voronoi Diagrams	$O(n\log n)$	$O(n^2)$	Dijkstra on graph nodes	125.7
Potential Fields	$O(n)$	$O(m)$	Improved numerical navigation function	128.0 [a]
Visibility Graph	$O(n^2)$	$O(n^2)$	A* on graph nodes	100.0
V-P Hybrid	$O(n\log n)$	$O(\sqrt{m})$	Steepest descent – mildest ascent	126.8
V-V-P Compound	$O(n\log n)$	$O(m)$	Steepest descent – mildest ascent	114.3

[a] After post-processing and path smoothing

Table 2. Time complexity and path quality comparison for five path planning approaches.

It is worth noting that similar to the V-P Hybrid method (Sec. 2.4), the V-V-P Compound algorithm has the property of completeness.

3.6 Extension to Higher Spaces

The V-V-P algorithm has the potential to be extended to three and higher dimensional spaces. Though the full *n*-dimensional implementation of the algorithm is among our future research, we will briefly discuss here the possibility of its extension to 3D.

Recall that the Generalized Voronoi Graph (GVG) in *n*-D space is the locus of points being equidistant from *n* or more obstacle features. Figs. 19(a)-(b) demonstrate a 3D environment and its GVG. The GVG is constructed incrementally using an algorithm which is the 3D version of our work presented in (Masehian et al., 2003).

Due to the one-dimensional nature of the GVG roadmap, the pruning procedure is still applicable to 3D context. Fig. 19(c) depicts the result of pruning the GVG in Fig. 19(a), after fixing Start and Goal positions. Similar to the 2D case, the pruning procedure reduces the search space considerably in 3D.

The Maximal Inscribed Discs can easily be generalized to 3D space, resulting in *Maximal Inscribed Balls* (*MIB*s), which are spheres centered on the GVG and tangent to 3 or more obstacle boundaries. In the same manner, we can extend the concept of αMID to αMIB, and the concept of *Region*(αMID) to *Region*(αMIB). The *Region*(αMIB) is a network of "tube-like" obstacle-free navigable channels. Fig. 19(d) illustrates the *Region*(□*MIB*) with $\alpha = 0.5$. Greater values for α cause "fatter" tubes, and freer space for robot's maneuvering.

The visibility scan in 3D can be applied via "sweep surfaces" instead of sweep rays in the 2D method. The robot should scan the space inside the *Region*(αMIB) to find "tangent surfaces". The Potential calculations for gridpoints is still tractable in 3D workspace, and the

search phase can be performed similar to the 2D V-V-P method; the Visibility and Potential Field modules will execute alternately, and the valley filling procedure will change to "tube filling". Therefore, the V-V-P and V-P models are extendable to at least 3D C-spaces.

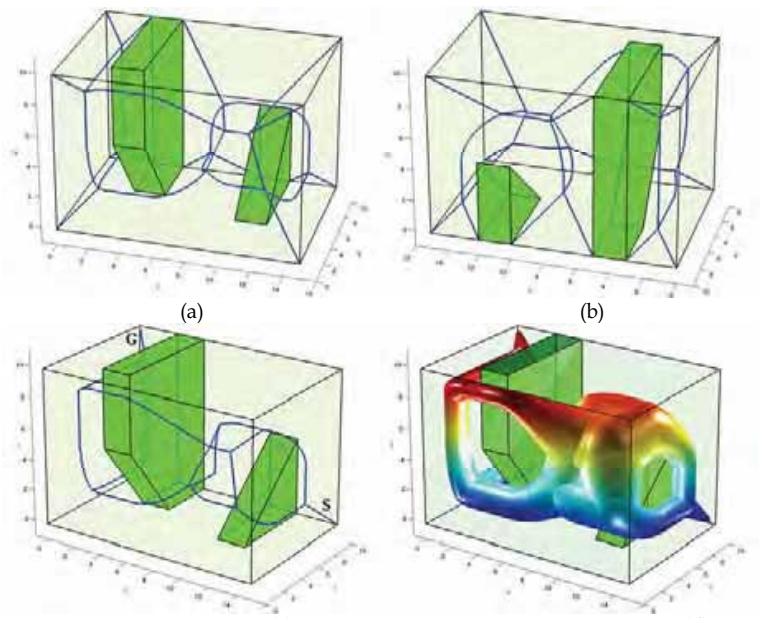


Fig. 19. (a) Front view, and, (b) Back view of the medial axis (GVG) of a 3D workspace. (c) The PGVG of the same workspace. (d) The *Region*(α*MIB*).

4. Summary and Future work

This chapter introduces two new offline path planning models which are based on the Roadmap and Potential Fields classic motion planning approaches. It is shown that how some relatively old methods can combine and yield new models.

The first path planning model is established based on two traditional methods: the Voronoi Diagrams and Potential Fields, and so is called V-P Hybrid model. The model integrates the advantages of Voronoi diagram's safest distance and Potential Fields' search simplicity properties. After constructing the Generalized Voronoi Graph roadmap for the workspace, it is reduced to the Pruned Generalized Voronoi Graph (PGVG) through a pruning procedure. The PGVG decreases the search time effectively. An attractive potential is then applied to the resulting roadmap, which yields a new version of Potential Fields method, since it implicitly models the obstacles by attractive potentials rather than repulsive ones. The search technique developed for finding the trajectory is a bidirectional steepest descent – mildest ascent stage-by-stage method, which is complete, and performs much faster than the classical Potential Fields or Dijkstra's methods.

The second model is a generalization of the V-P Hybrid model: it integrates three main approaches: Voronoi Diagrams, Visibility Graph, and Potential Fields, and is called V-V-P Compound path planner. After constructing the PGVG roadmap, a broad freeway net (called *Region*(□*MID*)) is developed based on the Maximal Inscribed Discs concept. A potential function is then assigned to this net to form an obstacle-free network of valleys. Afterwards, a bidirectional search technique is used where the Visibility Graph and Potential Fields modules execute alternately from both start and goal configurations. The steepest descent – mildest ascent search method is used for valley filling and local planning to avoid local minima. This Compound model provides a parametric tradeoff between safest and shortest paths, and generally yields shorter paths than the Voronoi and Potential Fields methods, and faster solutions than the Visibility Graph.

Different implementations of the presented algorithms exhibited these models' competence in solving path planning problems in complex and maze-like environments. Comparisons with classical Potential Fields and A* methods showed that composite methods usually perform faster and explore far less grid-points.

The developed composite path planning models can however be extended in numerous directions to accommodate more general assumptions. Here we mention two possible extensions which are achievable in the future versions of the models:

(1) Both methods are basically developed for point robots. This assumption is not realistic and requires an extra preprocessing step for obstacle expanding through the Minkowski Set Difference technique. Moreover, the robot is bound to have mere translational movements, and not rotational. The models can be modified to accommodate arbitrary-shaped robots with rotational ability.

(2) The developed models handle single-robot problems. The potential valleys in both V-P and V-V-P models may provide a framework for multiple robots motion planning. Especially, the Visibility component of the Compound model can be readily applied to mobile robots teams with vision capabilities.

5. References

Agarwal, P.K.; de Berg, M.; Matousek, J. & Schwarzkopf, O. (1998). Constructing levels in arrangements and higher order Voronoi diagrams, *SIAM Journal on Computing*, Vol. 27, 1998, pp. 654-667.

Alvarez, D.; Alvarez, J.C.; & González, R.C. (2003). Online motion planning using Laplace potential fields, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2003)*, pp. 3347-3352, Taiwan, September 2003, IEEE, Taipei.

Aurenhammer, F. & Klein, R. (2000). Voronoi diagrams, In: *Handbook of Computational Geometry*, Sack, J. & Urrutia, G. (Eds.), pp. 201-290, Elsevier/North-Holland, ISBN: 0444825371, Amsterdam.

Barraquand, J.; Langlois, B. & Latombe, J.C. (1992). Numerical potential field techniques for robot path planning, *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 22, No. 2, (March/April 1992), pp. 224-241.

Brock, O. & Khatib, O. (1999). High-speed navigation using the global dynamic window approach, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '99)*.

Canny, J.F. (1985). A voronoi method for the piano movers' problem, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '85)*.

Canny, J.F. (1998). *The Complexity of Robot Motion Planning*, MIT Press, ISBN: 0262031361, Cambridge, Mass.

Choset, H. & Burdick, J. (2000). Sensor-based exploration: The hierarchical generalized Voronoi graph, *International Journal of Robotics Research*, Vol. 19, No. 2, 2000, pp. 96-125.

Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.E. & Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, ISBN: 0262033275, Boston.

Hwang, Y.K. & Ahuja, N. (1992). Gross motion planning-A survey, *ACM Computing Surveys*, Vol. 24, No. 3, (September 1992), pp. 219-291.

Kavraki, L.E.; Svestka, P.; Latombe, J.C. & Overmars, M. (1996). Probabilistic roadmaps for path planning in high dimensional configuration spaces, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, (August 1996), pp. 566-580.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots, *International Journal of Robotics Research*, Vol. 5, No. 1, pp. 90-98.

Koren, Y. & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '91)*, April 1991, pp. 1398-1404.

Latombe, J.C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers, ISBN: 0792391292, Boston.

Masehian, E.; Amin-Naseri, M.R. & Khadem, S.E. (2003). Online motion planning using incremental construction of medial axis, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2003)*, (September 2003), Taiwan, pp. 2928-2933.

Masehian, E. & Amin-Naseri, M.R. (2004). A Voronoi diagram – visibility graph – potential fields compound algorithm for robot path planning, *Journal of Robotic Systems*, Vol. 21, No. 6, (June 2004), pp. 275-300.

Oommen, J.B.; Iyengar, S.S.; Rao, N.S.V. & Kashyap, R.L. (1987). Robot navigation in unknown terrains using visibility graphs: Part I: The disjoint convex obstacle case, *IEEE Journal of Robotics and Automation, Vol. RA-3*, (1987), pp. 672-681.

Sato, K. (1993). Deadlock-free motion planning using the Laplace potential field, *Advanced Robotics*, Vol. 7, No. 5, 1993, pp. 449-462.

Yeung, D.Y. & Bekey, G.A. (1987). A decentralized approach to the motion planning problem for multiple mobile robots, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '87)*, pp. 1779-1784.

13

Global Navigation of Assistant Robots using Partially Observable Markov Decision Processes

María Elena López, Rafael Barea, Luis Miguel Bergasa,
Manuel Ocaña & María Soledad Escudero
Electronics Department. University of Alcalá
Spain

1. Introduction

In the last years, one of the applications of service robots with a greater social impact has been the assistance to elderly or disabled people. In these applications, assistant robots must robustly navigate in structured indoor environments such as hospitals, nursing homes or houses, heading from room to room to carry out different nursing or service tasks.

Although the state of the art in navigation systems is very wide, there are not systems that simultaneously satisfy all the requirements of this application. Firstly, it must be a very robust navigation system, because it is going to work in highly dynamic environments and to interact with non-expert users. In second place, and to ensure the future commercial viability of this kind of prototypes, it must be a system very easy to export to new working domains, not requiring a previous preparation of the environment or a long, hard and tedious configuration process. Most of the actual navigation systems propose "ad-hoc" solutions that only can be applied in very specific conditions and environments. Besides, they usually require an artificial preparation of the environment and are not capable of automatically recover general localization failures.

In order to contribute to this research field, the Electronics Department of the University of Alcalá has been working on a robotic assistant called SIRA, within the projects SIRAPEM (Spanish acronym of Robotic System for Elderly Assistance) and SIMCA (Cooperative multi-robot assistance system). The main goal of these projects is the development of robotic aids that serve primary functions of tele-presence, tele-medicine, intelligent reminding, safeguarding, mobility assistance and social interaction. Figure 1 shows a simplified diagram of the SIRAPEM global architecture, based on a commercial platform (the PeopleBot robot of ActivMedia Robotics) endowed with a differential drive system, encoders, bumpers, two sonar rings (high and low), loudspeakers, microphone and on-board PC. The robot has been also provided with a PTZ color camera, a tactile screen and wireless Ethernet link. The system architecture includes several human-machine interaction systems, such as voice (synthesis and recognition speech) and touch screen for simple command selection.

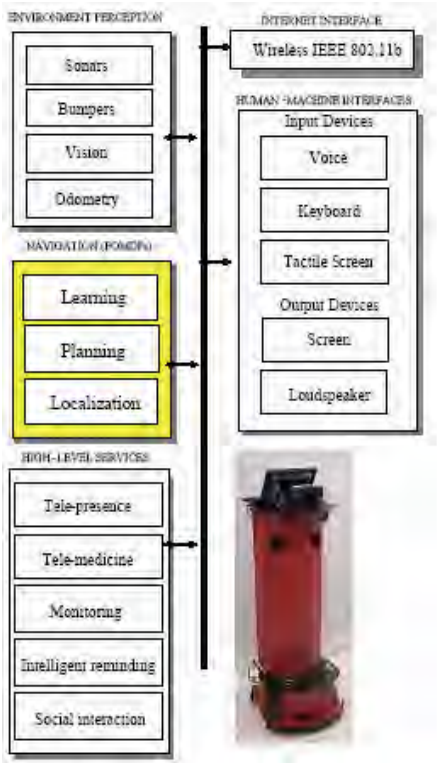


Fig. 1. Global architecture of the SIRAPEM System

This chapter describes the navigation module of the SIRAPEM project, including localization, planning and learning systems. A suitable framework to cope with all the requirements of this application is Partially Observable Markov Decision Processes (POMDPs). These models use probabilistic reasoning to deal with uncertainties, and a topological representation of the environment to reduce memory and time requirements of the algorithms. For the proposed global navigation system, in which the objective is the guidance to a goal room and some low-level behaviors perform local navigation, a topological discretization is appropriate to facilitate the planning and learning tasks.
POMDP models provide solutions to localization, planning and learning in the robotics context, and have been used as probabilistic reasoning method in the three modules of the navigation system of SIRA. The main contributions of the navigation architecture of SIRA, regarding other similar ones (that we'll be referenced in next section), are the following:

- Addition of visual information to the Markov model, not only as observation, but also for improving state transition detection. This visual information reduces typical perceptual aliasing of proximity sensors, accelerating the process of global localization when initial pose is unknown.

- Development of a new planning architecture that selects actions to combine several objectives, such as guidance to a goal room, localization to reduce uncertainty, and environment exploration.
- Development of a new exploration and learning strategy that takes advantage of human-machine interaction to robustly and quickly fast learn new working environments.

The chapter is organized as follows. Section 2 places this work within the context of previous similar ones. A brief overview of POMDPs foundations is presented as background in section 3. Section 4 describes the proposed Markov model while section 5 shows the global architecture of the navigation system. The localization module is described in section 6, the two layers of the planning system are shown in section 7 and the learning and exploration module are explained in section 8. Finally, we show some experimental results (section 9), whereas a final discussion and conclusion summarizes the chapter (sections 10 and 11).

2. Related Previous Work

Markov models, and particularly POMDPs, have already been widely used in robotics, and especially in robot navigation. The robots DERVISH (Nourbakhsh et al., 1995), developed in the Stanford University, and Xavier (Koenig & Simmons, 1998), in the Carnegie Mellon University, were the first robots successfully using this kind of navigation strategies for localization and action planning. Other successful robots guided with POMDPs are those proposed by (Zanichelli, 1999) or (Asoh et al., 1996). In the nursing applications field, in which robots interact with people and uncertainty is pervasive, robots such as Flo (Roy et al., 2000) or Pearl (Montemerlo et al., 2002) use POMDPs at all levels of decision making, and not only in low-level navigation routines.

However, in all these successful navigation systems, only proximity sensors are used to perceive the environment. Due to the typical high perceptual aliasing of these sensors in office environments, using only proximity sensors makes the Markov model highly non-observable, and the initial global localization stage is rather slow.

On the other hand, there are quite a lot of recent works using appearance-based methods for robot navigation with visual information. Some of these works, such as (Gechter et al., 2001) and (Regini et al., 2002), incorporate POMDP models as a method for taking into account previous state of the robot to evaluate its new pose, avoiding the teleportation phenomena. However, these works are focused on visual algorithms, and very slightly integrate them into a complete robot navigation architecture. So, the above referenced systems don't combine any other sensorial system, and use the POMDP only for localizing the robot, and not for planning or exploring.

This work is a convergence point between these two research lines, proposing a complete navigation architecture that adds visual information to proximity sensors to improve previous navigation results, making more robust and faster the global localization task. Furthermore, a new Markov model is proposed that better adapts to environment topology, being completely integrated with a planning system that simultaneously contemplates several navigation objectives.

Regarding the learning system, most of the related works need a previous "hand-made" introduction of the Markov model of a new environment. Learning a POMDP involves two main issues: (1) obtaining its topology (structure), and (2) adjusting the parameters (probabilities) of the

model. The majority of the works deals with the last problem, using the well-known EM algorithm to learn the parameters of a Markov model whose structure is known (Thrun et al., 1998; Koenig and Simmons, 1996). However, because computational complexity of the learning process increases exponentially as the number of states increases, these methods are still time consuming and its working ability is limited to learn reduced environments. In this work, the POMDP model can be easily obtained for new environments by means of human-robot cooperation, being an optimal solution for assistant robots endowed with human-machine interfaces. The topological representation of the environment is intuitive enough to be easily defined by the designer. The uncertainties and observations that constitute the parameters of the Markov model are learned by the robot using a modification of the EM algorithm that exploits slight user supervision and topology constraints to highly reduce memory requirements and computational cost of the standard EM algorithm.

3. POMDPs Review

Although there is a wide literature about POMDPs theory (Papadimitriou & Tsitsiklis, 1987; Puterman, 1994; Kaelbling et al., 1996) in this section some terminology and main foundations are briefly introduced as theoretical background of the proposed work.

A Markov Decision Process (MDP) is a model for sequential decision making, formally defined as a tuple $\{\boldsymbol{S},\boldsymbol{A},\boldsymbol{T},\boldsymbol{R}\}$, where,

- $\boldsymbol{S}$ is a finite set of states ($s \in \boldsymbol{S}$).
- $\boldsymbol{A}$ is a finite set of actions ($a \in \boldsymbol{A}$).
- $\boldsymbol{T}=\{p(s' \mid s,a) \quad \forall (s,s' \in \boldsymbol{S} \quad a \in \boldsymbol{A})\}$ is a state transition model which specifies a conditional probability distribution of posterior state s' given prior state s and action executed a.
- $\boldsymbol{R}=\{r(s,a) \quad \forall (s \in \boldsymbol{S} \quad a \in \boldsymbol{A})\}$ is the reward function, that determines the immediate utility (as a function of an objective) of executing action a at state s.

A MDP assumes the Markov property, which establishes that actual state and action are the only information needed to predict next state:

$$p(s_{t+1} \mid s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = p(s_{t+1} \mid s_t, a_t) \tag{1}$$

In a MDP, the actual state s is always known without uncertainty. So, planning in a MDP is the problem of action selection as a function of the actual state (Howard, 1960). A MDP solution is a policy $a=\pi(s)$, which maps states into actions and so determines which action must be executed at each state. An optimal policy $a=\pi^*(s)$ is that one that maximizes future rewards. Finding optimal policies for MDPs is a well known problem in the artificial intelligent field, to which several exact and approximate solutions (such as the "value iteration" algorithm) have been proposed (Howard, 1960; Puterman, 1994).

Partially Observable Markov Decision Processes (POMDPs) are used under domains where there is not certainty about the actual state of the system. Instead, the agent can do observations and use them to compute a probabilistic distribution over all possible states. So, a POMDP adds:

- $\boldsymbol{O}$, a finite set of observations ($o \in \boldsymbol{O}$)
- $\boldsymbol{\vartheta}=\{p(o \mid s) \quad \forall o \in \boldsymbol{O},\ s \in \boldsymbol{S}\}$ is an observation model which specifies a conditional probability distribution over observations given the actual state s.

Because in this case the agent has not direct access to the current state, it uses actions and observations to maintain a probability distribution over all possible states, known as the

"belief distribution", *Bel(**S**)*. A POMDP is still a markovian process in terms of this probability distribution, which only depends on the prior belief, prior action, and current observation. This belief must be updated whenever a new action or perception is carried out. When an action *a* is executed, the new probabilities become:

$$\mathrm{Bel}_{\mathrm{posterior}}(\mathbf{S}=s') = K \cdot \sum_{s \in S} p(s' \mid s,a) \cdot \mathrm{Bel}_{\mathrm{prior}}(s) \qquad \forall s' \in \mathbf{S} \tag{2}$$

where K is a normalization factor to ensure that the probabilities all sum one. When a sensor report *o* is received, the probabilities become:

$$\mathrm{Bel}_{\mathrm{posterior}}(\mathbf{S}=s) = K \cdot p(o \mid s) \cdot \mathrm{Bel}_{\mathrm{prior}}(s) \qquad \forall s \in \mathbf{S} \tag{3}$$

In a POMDP, a policy $a=\pi(\mathbf{Bel})$ maps beliefs into actions. However, what in a MDP was a discrete state space problem, now is a high-dimensional continuous space. Although there are numerous studies about finding optimal policies in POMDPs (Cassandra, 1994; Kaelbling et al.,1998), the size of state spaces and real-time constraints make them infeasible to solve navigation problems in robotic contexts. This work uses an alternative approximate solution for planning in POMDP-based navigation contexts, dividing the problem into two layers and applying some heuristic strategies for action selection.

In the context of robot navigation, the states of the Markov model are the locations (or nodes) of a topological representation of the environment. Actions are local navigation behaviors that the robot can execute to move from one state to another, and observations are perceptions of the environment that the robot can extract from its sensors. In this case, the Markov model is partially observable because the robot may never know exactly which state it is in.

4. Markov Model for Global Navigation

A POMDP model for robot navigation is constructed from two sources of information: the topology of the environment, and some experimental or learned information about action and sensor errors and uncertainties.

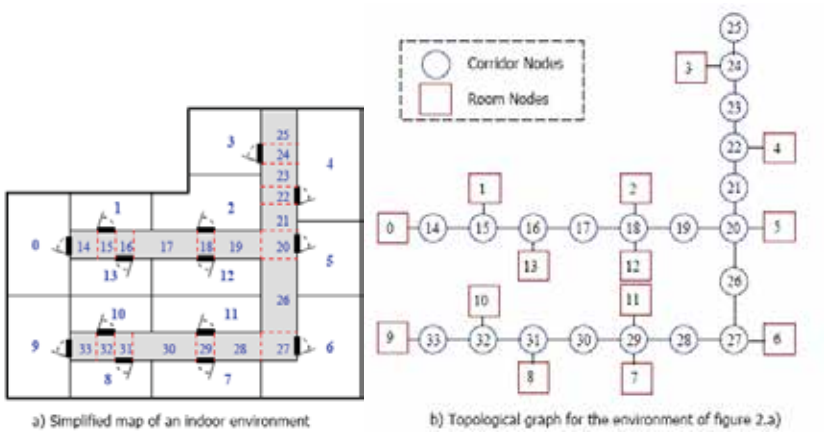


Fig. 2. Topological graph of an environment map.

Taking into account that the final objective of the SIRAPEM navigation system is to direct the robot from one room to another to perform guiding or service tasks, we discretize the environment into coarse-grained regions (nodes) of variable size in accordance with the topology of the environment, in order to make easier the planning task. As it's shown in figure 2 for a virtual environment, only one node is assigned to each room, while the corridor is discretized into thinner regions. The limits of these regions correspond to any change in lateral features of the corridor (such as a new door, opening or piece of wall). This is a suitable discretization method in this type of structured environments, since nodes are directly related to topological locations in which the planning module may need to change the commanded action.

4.1. The elements of the Markov model: states, actions and observations

States (***S***) of the Markov model are directly related to the nodes of the topological graph. A single state corresponds to each room node, while four states are assigned to each corridor node, one for each of the four orientations the robot can adopt.

The actions (***A***) selected to produce transitions from one state to another correspond to local navigation behaviors of the robot. We assume imperfect actions, so the effect of an action can be different of the expected one (this will be modeled by the transition model **T**). These actions are:

(1) *"Go out room"* (a_O): to traverse door using sonar an visual information in room states,
(2) *"Enter room"* (a_E): only defined in corridor states oriented to a door,
(3) *"Turn right"* (a_R): to turn 90° to the right,
(4) *"Turn Left"* (a_L): to turn 90° to the left,
(5) *"Follow Corridor"* (a_F): to continue through the corridor to the next state, and
(6) *"No Operation"* (a_{NO}): used as a directive in the goal state.

Finally, the observations (***O***) in our model come from the two sensorial systems of the robot: sonar and vision. Markov models provide a natural way to combine multisensorial information, as it will be shown in section 4.2.1. In each state, the robot makes three kind of observations:

(1) *"Abstract Sonar Observation"* (o_{ASO}). Each of the three nominal directions around the robot (left, front and right) is classified as "free" or "occupied" using sonar information, and an abstract observation is constructed from the combination of the percepts in each direction (thus, there are eight possible abstract sonar observations, as it's shown in figure 3.a).
(2) *"Landmark Visual Observation"* (o_{LVO}). Doors are considered as natural visual landmarks, because they exist in all indoor environments and can be easily segmented from the image using color (previously trained) and some geometrical restrictions. This observation is the number of doors (in lateral walls of the corridor) extracted from the image (see figure 3.b), and it reduces the perceptual aliasing of sonar by distinguishing states at the beginning from states at the end of a corridor. However, in long corridors, doors far away from the robot can't be easily segmented from the image (this is the case of image 2 of figure 3.b), and this is the reason because we introduce a third visual observation.
(3) *"Depth Visual Observation"* (o_{DVO}). As human-interaction robots have tall bodies with the camera on the top, it's possible to detect the vanishing ceiling lines, and

classify its length into a set of discrete values (in this case, we use four quantification levels, as it's shown in figure 3.b). This is a less sensitive to noise observation than using floor vanishing lines (mainly to occlusions due to people walking through the corridor), and provides complementary information to o_{LVO}.

Figure 3.b shows two scenes of the same corridor from different positions, and their corresponding o_{LVO} and o_{DVO} observations. It's shown that these are obtained by means of very simple image processing techniques (color segmentation for o_{LVO} and edge detection for o_{DVO}), and have the advantage, regarding correlation techniques used in (Gechter et al., 2001) or (Regini et al., 2002), that they are less sensitive to slight pose deviations within the same node.

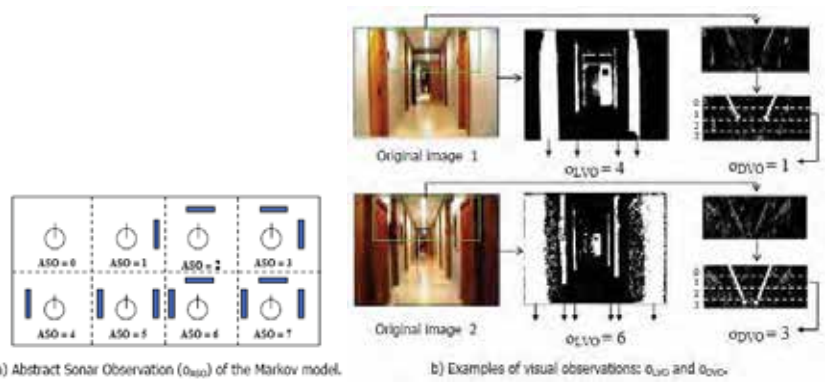


Fig. 3. Observations of the proposed Markov model.

4.2. Visual information utility and improvements

Visual observations increase the robustness of the localization system by reducing perceptual aliasing. On the other hand, visual information also improves state transition detection, as it's shown in the following subsections.

4.2.1. Sensor fusion to improve observability

Using only sonar to perceive the environment makes the Markov model highly non-observable due to perceptual aliasing. Furthermore, the "Abstract Sonar Observation" is highly dependent on doors state (opened or closed). The addition of the visual observations proposed in this work augments the observability of states. For example, corridor states with an opened door on the left and a wall on the right produces the same abstract sonar observation (o_{ASO}=1) independently if they are at the beginning or at the end of the corridor. However, the number of doors seen from the current state (o_{LVO}) allows to distinguish between these states.

POMDPs provide a natural way for using multisensorial fusion in their observation models ($p(o \mid s)$ probabilities). In this case, **o** is a vector composed by the three observations proposed in the former subsection. Because these are independent observations, the observation model can be simplified in the following way:

$$p(o \mid s) = p(o_{ASO}, o_{LVO}, o_{DVO} \mid s) = p(o_{ASO} \mid s)\, p(o_{LVO} \mid s)\, p(o_{DVO} \mid s) \tag{4}$$

4.2.2. Visual information to improve state transition detection

To ensure that when the robot is in a corridor, it only adopts the four allowed directions without large errors, it's necessary that, during the execution of a "Follow Corridor" action, the robot becomes aligned with the corridor longitudinal axis. So, when the robot stands up to a new corridor, it aligns itself with a subtask that uses visual vanishing points, and during corridor following, it uses sonar buffers to detect the walls and construct a local model of the corridor. Besides, an individual "Follow Corridor" action terminates when the robot eaches a new state of the corridor. Detecting these transitions only with sonar readings is very critical when doors are closed.

To solve this problem, we add visual information to detect door frames as natural landmarks of state transitions (using color segmentation and some geometrical restrictions). The advantage of this method is that the image processing step is fast and easy, being only necessary to process two lateral windows of the image as it's shown in figure 4.

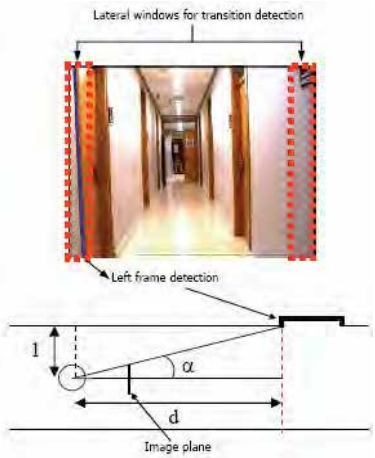


Fig. 4. State transition detection by means of visual information.

Whenever a vertical transition from wall to door color (or vice versa) is detected in a lateral window, the distance to travel as far as that new state is obtained from the following formula, using a pin-hole model of the camera (see figure 4):

$$d = \frac{l}{tg(\alpha)} = K \cdot l \tag{5}$$

where l is the distance of the robot to the wall in the same side as the detected door frame (obtained from sonar readings) and α is the visual angle of the door frame. As the detected frame is always in the edge of the image, the visual angle α only depends on the focal distance of the camera, that is constant for a fixed zoom (and known from camera specifications). After covering distance d (measured with relative odometry readings), the robot reaches the new state. This transition can be confirmed (fused) with sonar if the door is opened. Another advantage of this transition detection approach is that no assumptions are made about doors or corridor widths.

4.3. Action and observation uncertainties

Besides the topology of the environment, it's necessary to define some action and observation uncertainties to generate the final POMDP model (transition and observation matrixes). A first way of defining these uncertainties is by introducing some experimental "hand-made" rules (this method is used in (Koenig & Simmons, 1998) and (Zanichelli, 1999)). For example, if a "Follow" action (a_F) is commanded, the expected probability of making a state transition (F) is 70%, while there is a 10% probability of remaining in the same state (N=no action), a 10% probability of making two successive state transitions (FF), and a 10% probability of making three state transitions (FFF). Experience with this method has shown it to produce reliable navigation. However, a limitation of this method is that some uncertainties or parameters of the transition and observation models are not intuitive for being estimated by the user. Besides, results are better when probabilities are learned to more closely reflect the actual environment of the robot. So, our proposed learning module adjusts observation and transition probabilities with real data during an initial exploration stage, and maintains these parameters updated when the robot is performing another guiding or service tasks. This module, that also makes easier the installation of the system in new environments, is described in detail in section 8.

5. Navigation System Architecture

The problem of acting in partially observable environments can be decomposed into two components: a state estimator, which takes as input the last belief state, the most recent action and the most recent observation, and returns an updated belief state, and a policy, which maps belief states into actions. In robotics context, the first component is robot localization and the last one is task planning.

Figure 5 shows the global navigation architecture of the SIRAPEM project, formulated as a POMDP model. At each process step, the planning module selects a new action as a command for the local navigation module, that implements the actions of the POMDP as local navigation behaviors. As a result, the robot modifies its state (location), and receives a new observation from its sensorial systems. The last action executed, besides the new observation perceived, are used by the localization module to update the belief distribution *Bel(S)*.

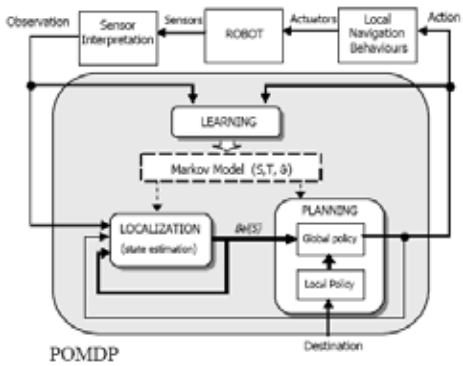


Fig. 5. Global architecture of the navigation system.

After each state transition, and once updated the belief, the planning module chooses the next action to execute. Instead of using an optimal POMDP policy (that involves high computational times), this selection is simplified by dividing the planning module into two layers:

- A local policy, that assigns an optimal action to each individual state (as in the MDP case). This assignment depends on the planning context. Three possible contexts have been considered: (1) guiding (the objective is to reach a goal room selected by the user to perform a service or guiding task), (2) localizing (the objective is to reduce location uncertainty) and (3) exploring (the objective is to learn or adjust observations and uncertainties of the Markov model).
- A global policy, that using the current belief and the local policy, selects the best action by means of different heuristic strategies proposed by (Kaelbling et al., 1996).

This proposed two-layered planning architecture is able to combine several contexts of the local policy to simultaneously integrate different planning objectives, as will be shown in subsequent sections.

Finally, the learning module (López et al., 2004) uses action and observation data to learn and adjust the observations and uncertainties of the Markov model.

6. Localization and Uncertainty Evaluation

The localization module updates the belief distribution after each state transition, using the well known Markov localization equations (2) and (3).

In the first execution step, the belief distribution can be initialized in one of the two following ways: (a) If initial state of the robot is known, that state is assigned probability 1 and the rest 0, (b) If initial state is unknown, a uniform distribution is calculated over all states.

Although the planning system chooses the action based on the entire belief distribution, in some cases it´s necessary to evaluate the degree of uncertainty of that distribution (this is, the locational uncertainty). A typical measure of discrete distributions uncertainty is the entropy. The normalized entropy (ranging from 0 to 1) of the belief distribution is:

$$\mathrm{H}(\mathbf{Bel}) = -\frac{\sum_{s \in \mathbf{S}} Bel(s) \cdot \log(Bel(s))}{\log(n_s)} \tag{6}$$

where n_s is the number of states of the Markov model. The lower the value, the more certain the distribution. This measure has been used in all previous robotic applications for characterizing locational uncertainty (Kaelbling, 1996; Zanichelli, 1999).

However, this measure is not appropriate for detecting situations in which there are a few maximums of similar value, being the rest of the elements zero, because it's detected as a low entropy distribution. In fact, even being only two maximums, that is a not good result for the localization module, because they can correspond to far locations in the environment. A more suitable choice should be to use a least square measure respect to ideal delta distribution, that better detects the convergence of the distribution to a unique maximum (and so, that the robot is globally localized). However, we propose another approximate measure that, providing similar results to least squares, is faster calculated by using only the two first maximum values of the distribution (it's also less sensitive when uncertainty is high, and more sensitive to secondary maximums during the tracking stage). This is the normalized divergence factor, calculated in the following way:

$$\mathrm{D}(\boldsymbol{Bel}) = 1 - \frac{n_s \left(d_{\max} + p_{\max}\right) - 1}{2 \cdot n_s - 1} \tag{7}$$

where d_{max} is the difference between first and second maximum values of the distribution, and p_{max} the absolute value of the first maximum. Again, a high value indicates that the distribution converges to a unique maximum. In the results section we'll show that this new measure provides much better results when planning in some kind of environments.

7. Planning under Uncertainty

A POMDP model is a MDP model with probabilistic observations. Finding optimal policies in the MDP case (that is a discrete space model) is easy and quickly for even very large models. However, in the POMDP case, finding optimal control strategies is computationally intractable for all but the simplest environments, because the beliefs space is continuous and high-dimensional.
There are several recent works that use a hierarchical representation of the environment, with different levels of resolution, to reduce the number of states that take part in the planning algorithms (Theocharous & Mahadevan, 2002; Pineau & Thrun, 2002). However, these methods need more complex perception algorithms to distinguish states at different levels of abstraction, and so they need more prior knowledge about the environment and more complex learning algorithms. On the other hand, there are also several recent approximate methods for solving POMDPs, such as those that use a compressed belief distribution to accelerate algorithms (Roy, 2003) or the 'point-based value iteration algorithm' (Pineau et al., 2003) in which planning is performed only on a sampled set of reachable belief points.
The solution adopted in this work is to divide the planning problem into two steps: the first one finds an optimal local policy for the underlying MDP ($a^*=\pi^*(s)$, or to simplify notation, $a^*(s)$), and the second one uses a number of simple heuristic strategies to select a final action ($a^*(\boldsymbol{Bel})$) as a function of the local policy and the belief. This structure is shown in figure 6 and described in subsequent subsections.

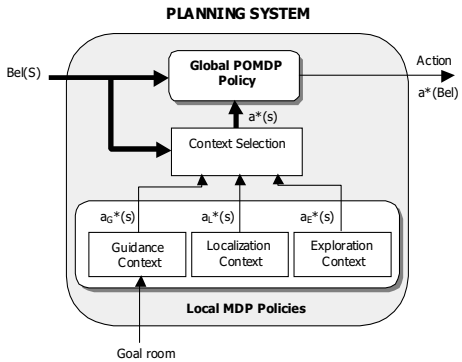


Fig. 6. Planning system architecture, consisting of two layers: (1) Global POMDP policy and (2) Local MDP policies.

7.1. Contexts and local policies

The objective of the local policy is to assign an optimal action ($a^*(s)$) to each individual state s. This assignment depends on the planning context. The use of several contexts allows the

robot to simultaneously achieve several planning objectives. The localization and guidance contexts try to simulate the optimal policy of a POMDP, which seamlessly integrates the two concerns of acting in order to reduce uncertainty and to achieve a goal. The exploration context is to select actions for learning the parameters of the Markov model.
In this subsection we show the three contexts separately. Later, they will be automatically selected or combined by the 'context selection' and 'global policy' modules (figure 6).

7.1.1. Guidance Context

This local policy is calculated whenever a new goal room is selected by the user. Its main objective is to assign to each individual state *s*, an optimal action ($a_G^*(s)$) to guide the robot to the goal.
One of the most well known algorithms for finding optimal policies in MDPs is 'value iteration' (Bellman, 1957). This algorithm assigns an optimal action to each state when the reward function $r(s,a)$ is available. In this application, the information about the utility of actions for reaching the destination room is contained in the graph. So, a simple path searching algorithm can effectively solve the underlying MDP, without any intermediate reward function.
So, a modification of the A* search algorithm (Winston, 1984) is used to assign a preferred heading to each node of the topological graph, based on minimizing the expected total number of nodes to traverse (shorter distance criterion cannot be used because the graph has not metric information). The modification of the algorithm consists of inverting the search direction, because in this application there is not an initial node (only a destination node). Figure 7 shows the resulting node directions for goal room 2 on the graph of environment of figure 2.

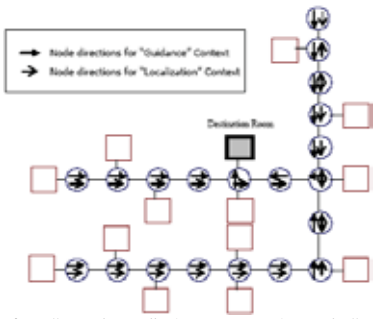


Fig. 7. Node directions for "Guidence" (to room 2) and "Localization" contexts for environment of figure 2.

Later, an optimal action is assigned to the four states of each node in the following way: a "follow" (a_F) action is assigned to the state whose orientation is the same as the preferred heading of the node, while the remaining states are assigned actions that will turn the robot towards that heading (a_L or a_R). Finally, a "no operation" action (a_{NO}) is assigned to the goal room state.
Besides optimal actions, when a new goal room is selected, Q(s,a) values are assigned to each (s,a) pair. In the MDPs theory, Q-values (Lovejoi, 1991) characterize the utility of executing each action at each state, and will be used by one of the global heuristic policies shown in next subsection. To simplify Q-values calculation, the following criterion has been used: Q(s,a)=1 if action a is optimal at state s, Q(s,a)=-1 (negative utility) if actions a is not

defined at state s, and Q(s,a)=-0.5 for the remaining cases (actions that disaligns the robot from the preferred heading).

7.1.2. Localization Context

This policy is used to guide the robot to "Sensorial Relevant States" (SRSs) that reduce positional uncertainty, even if that requires moving it away from the goal temporarily. This planning objective was not considered in previous similar robots, such as DERVISH (Nourbakhsh et al., 1995) or Xavier (Koenig & Simmons, 1998), or was implemented by means of fixed sequences of movements (Cassandra, 1994) that don't contemplate environment relevant places to reduce uncertainty.

In an indoor environment, it's usual to find different zones that produce not only the same observations, but also the same sequence of observations as the robot traverses them by executing the same actions (for example, symmetric corridors). SRSs are states that break a sequence of observations that can be found in another zone of the graph.

Because a state can be reached from different paths and so, with different histories of observations, SRSs are not characteristic states of the graph, but they depend on the starting state of the robot. This means that each starting state has its own SRS. To simplify the calculation of SRSs, and taking into account that the more informative states are those aligned with corridors, it has been supposed that in the localization context the robot is going to execute sequences of "follow corridor" actions. So, the moving direction along the corridor to reach a SRS as soon as possible must be calculated for each state of each corridor. To do this, the "Composed Observations" (COs) of these states are calculated from the graph and the current observation model ϑ in the following way:

$$CO(s) = 100 \cdot o_{DVO}(s) + 10 \cdot o_{LVO}(s) + o_{ASO}(s) \tag{8}$$

$$\text{with} \quad o_{DVO}(s) = \arg\max_{O_{DVO}} \left(p(o_{DVO} \mid s)\right)$$

$$o_{LVO}(s) = \arg\max_{O_{LVO}} \left(p(o_{LVO} \mid s)\right)$$

$$o_{ASO}(s) = \arg\max_{O_{ASO}} \left(p(o_{ASO} \mid s)\right)$$

Later, the nearest SRS for each node is calculated by studying the sequence of COs obtained while moving in both corridor directions. Then, a preferred heading (among them that align the robot with any connected corridor) is assigned to each node. This heading points at the corridor direction that, by a sequence of "Follow Corridor" actions, directs the robot to the nearest SRS (figure 7 shows the node directions obtained for environment of figure 2). And finally, an optimal action is assigned to the four states of each corridor node to align the robot with this preferred heading (as it was described in the guidance context section). The optimal action assigned to room states is always "Go out room" (a_O).

So, this policy ($a^*_L(s)$) is only environment dependent and is automatically calculated from the connections of the graph and the ideal observations of each state.

7.1.3. Exploration Context

The objective of this local policy is to select actions during the exploration stage, in order to learn transition and observation probabilities. As in this stage the Markov model is unknown (the belief can't be calculated), there is not distinction between local and global policies, whose common function is to select actions in a reactive way to explore the

environment. As this context is strongly connected with the learning module, it will be explained in section 8.

7.2. Global heuristic policies

The global policy combines the probabilities of each state to be the current state (belief distribution Bel(***S***)) with the best action assigned to each state (local policy $a^*(s)$) to select the final action to execute, $a^*(\boldsymbol{Bel})$. Once selected the local policy context (for example guidance context, $a^*(s)=a_G^*(s)$), some heuristic strategies proposed by (Kaelbling et al., 1996) can be used to do this final selection.
The simpler one is the "Most Likely State" (MLS) global policy that finds the state with the highest probability and directly executes its local policy:

$$a^*_{MLS}(\boldsymbol{Bel}) = a^*\left(\arg\max_s\left(Bel(s)\right)\right) \tag{9}$$

The "Voting" global policy first computes the "probability mass" of each action ($V(a)$) (probability of action a being optimal) according to the belief distribution, and then selects the action that is most likely to be optimal (the one with highest probability mass):

$$\begin{aligned} V(a) &= \sum_{s\,|_{a^*(s)=a}} Bel(s) \qquad \forall a \in \boldsymbol{A} \\ a^*_{vot}(Bel) &= \arg\max_a\left(V(a)\right) \end{aligned} \tag{10}$$

This method is less sensitive to locational uncertainty, because it takes into account all states, not only the most probable one.
Finally, the Q_{MDP} global policy is a more refined version of the voting policy, in which the votes of each state are apportioned among all actions according to their Q-values:

$$\begin{aligned} V(a) &= \sum_{s\in S} Bel(s)\cdot Q^a(s) \quad \forall a \in \boldsymbol{A} \\ a^*_{Q_{MDP}}(Bel) &= \arg\max_a\left(V(a)\right) \end{aligned} \tag{11}$$

This is in contrast to the "winner take all" behavior of the voting method, taking into account negative effect of actions.
Although there is some variability between these methods, for the most part all of them do well when initial state of the robot is known, and only the tracking problem is present. If initial state is unknown, the performance of the methods highly depends on particular configuration of starting states. However, MLS or Q_{MDP} global policies may cycle through the same set of actions without progressing to the goal when only guidance context is used. Properly combination of guidance and localization context highly improves the performance of these methods during global localization stage.

7.3. Automatic context selection or combination

Apart from the exploration context, this section considers the automatic context selection (see figure 6) as a function of the locational uncertainty. When uncertainty is high, localization context is useful to gather information, while with low uncertainty, guidance context is the appropriate one. In some cases, however, there is benign high uncertainty in the belief state; that is, there is confusion among states that requires the same action. In these cases, it's not necessary to commute to localization context. So, an appropriate measure of

uncertainty is the "normalized divergence factor" of the probability mass distribution, D(V(*a*)), (see eq. 7).

The "thresholding-method" for context selection uses a threshold ϕ for the divergence factor D. Only when divergence is over that threshold (high uncertainty), localization context is used as local policy:

$$a^*(s) = \begin{cases} a_G^*(s) & if \quad D < \phi \\ a_L^*(s) & si \quad D \geq \phi \end{cases} \tag{12}$$

However, the "weighting-method" combines both contexts using divergence as weighting factor. To do this, probability mass distributions for guidance and localization contexts ($V_G(a)$ and $V_L(a)$) are computed separately, and the weighted combined to obtain the final probability mass V(*a*). As in the voting method, the action selected is the one with highest probability mass:

$$\begin{aligned} V(a) &= (1-D)\cdot V_G(a) + D \cdot V_L(s) \\ a^*(Bel) &= \arg\max_a (V(a)) \end{aligned} \tag{13}$$

8. Learning the Markov Model of a New Environment

The POMDP model of a new environment is constructed from two sources of information:

- The topology of the environment, represented as a graph with nodes and connections. This graph fixes the states ($s \in \mathbf{S}$) of the model, and establishes the ideal transitions among them by means of logical connectivity rules.
- An uncertainty model, that characterizes the errors or ambiguities of actions and observations, and together with the graph, makes possible to generate the transition T and observation $\hat{\mathbf{0}}$ matrixes of the POMDP.

Taking into account that a reliable graph is crucial for the localization and planning systems to work properly, and the topological representation proposed in this work is very close to human environment perception, we propose a manual introduction of the graph. To do this, the SIRAPEM system incorporates an application to help the user to introduce the graph of the environment (this step is needed only once when the robot is installed in a new working domain, because the graph is a static representation of the environment).

```
GRAPH DEFINITION
****************
Total number of nodes:      32
Number of rooms:            14
Labels of rooms:    ROOM 0 (node 0) :    "dinning room A"
                    ROOM 1 (node 1) :    "gymnasium"
                    ................
                    ROOM 13(node 13):    "bedroom 101"

Connections (c=corridor, w=wall, r=room)
        NODE 14:    Right:  c15
                    Up:     w
                    Left:   r0
                    Down:   w
       ................
        NODE 33:    Right:  c32
                    Up:     w
                    Left:   r9
                    Down:   w
```

Fig. 8. Example of graph definition for the environment of Fig. 2.

After numbering the nodes of the graph (the only condition to do this is to assign the lower numbers to room nodes, starting with 0), the connections in the four directions of each corridor node must be indicated. Figure 8 shows an example of the "Graph definition" application (for the environment of figure 2), that also allows to associate a label to each room. These labels will be identified by the voice recognition interface and used as user commands to indicate goal rooms.
Once defined the graph, the objective of the learning module is to adjust the parameters of the POMDP model (entries of transition and observation matrixes). Figure 9 shows the steps involved in the POMDP generation of a new working environment. The graph introduced by the designer, together with some predefined initial uncertainty rules are used to generate an initial POMDP. This initial POMDP, described in next subsection, provides enough information for corridor navigation during an exploration stage, whose objective is to collect data in an optimum manner to adjust the settable parameters with minimum memory requirements and ensuring a reliable convergence of the model to fit real environment data (this is the "active learning" stage). Besides, during normal working of the navigation system (performing guiding tasks), the learning module carries on working ("passive learning" stage), collecting actions and observations to maintain the parameters updated in the face of possible changes.

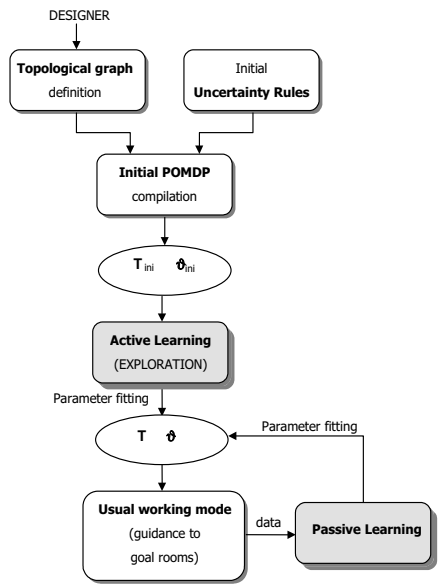


Fig.9. Steps for the introduction and learning of the Markov model of a new environment.

8.1. Settable parameters and initial POMDP compilation

A method used to reduce the amount of training data needed for convergence of the EM algorithm is to limit the number of model parameters to be learned. There are two reasons because some parameters can be excluded off the training process:

- Some parameters are only robot dependent, and don't change from one environment to another. Examples of this case are the errors in turn actions (that are nearly deterministic due to the accuracy of odometry sensors in short turns), or errors of sonars detecting "free" when "occupied" or vice versa.
- Other parameters directly depend on the graph and some general uncertainty rules, being possible to learn the general rules instead of its individual entries in the model matrixes. This means that the learning method constrains some probabilities to be identical, and updates a probability using all the information that applies to any probability in its class. For example, the probability of losing a transition while following a corridor can be supposed to be identical for all states in the corridor, being possible to learn the general probability instead of the particular ones.

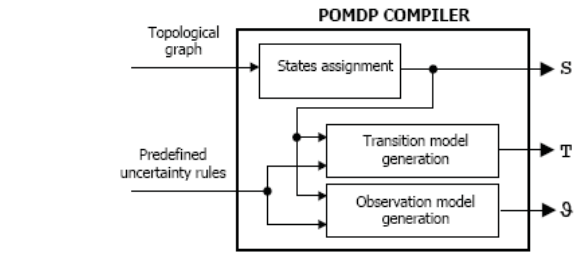
Taking these properties into account, table 1 shows the uncertainty rules used to generate the initial POMDP in the SIRAPEM system.

TRANSITION MODEL UNCERTAINTIES (F=Follow, L=Left, R=Right, O=Out, E=Enter,N=No action)				
Command	**Effect of Command (% probabilities)**			
a_F	**N** = 10	**F** = 70	**FF** = 10	**FFF**= 10
a_L	**N** = 5	**L** = 90	**LL** = 5	
a_R	**N** = 5	**R** = 90	**RR** = 5	
a_O	**N** = 5	**O** = 85	**OF** = 10	
a_E	**N** = 10	**E** = 90		
OBSERVATION MODEL UNCERTAINTIES				
ASO Model				
Open door probability (for all doors)	50 %			
Prob. of detecting something being nothing	10 %			
Prob. of detecting nothing being something	5 %			
LVO Model				
Room states	Uniform distribution over o_{LVO}={0,1,2}			
Corridor states perpendicular to corridor direction and oriented to a door	Uniform distribution over o_{LVO}={0,1,2}			
Corridor states perpendicular to corridor direction and oriented to a wall	Uniform distribution over o_{LVO}={0,1}			
Corridor states aligned with corridor direction	Uniform distribution over o_{LVO}			
DVO Model				
Room states	Delta distribution in o_{DVO}=0			
Corridor states perpendicular to corridor direction	Delta distribution in o_{DVO}=0			
Corridor states aligned with corridor direction	Uniform distribution over o_{DVO}			

Table 1. Predefined uncertainty rules for constructing the initial POMDP model.

Figure 10 shows the process of initial POMDP compilation. Firstly, the compiler automatically assigns a number (n_s) to each state of the graph as a function of the number of the node to which it belongs (n) and its orientation within the node (head={0(right), 1(up), 2(left), 3(down)}) in the following way (n_rooms being the number of room nodes):

Room states: n_s=n
Corridor states: n_s=n_rooms+(n-n_rooms)*4+head



a_e=Go Out Room

S \ S'	0	1	2	3	4	5	6	7	8	...
0	5	0	0	0	0	0	0	0	0	
1	0	5	0	0	0	0	0	0	0	
2	0	0	5	0	0	0	0	0	0	
3	0	0	0	5	0	0	0	0	0	
4	0	0	0	0	5	0	0	0	0	...
5	0	0	0	0	0	5	0	0	0	
6	0	0	0	0	0	0	5	0	0	
7	0	0	0	0	0	0	0	5	0	
8	0	0	0	0	0	0	0	0	5	
...				...						

a_E=Enter Room

S \ S'	0	1	2	3	4	5	6	7	8	...
...					...					
16	90	0	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	
19	0	90	0	0	0	0	0	0	0	...
20	0	0	0	0	0	0	0	0	0	
21	0	0	0	0	0	0	0	0	0	
22	0	0	0	0	0	0	0	0	0	
23	0	0	0	0	0	0	0	0	0	
...				...						

a_F=Follow Corridor

S \ S'	...	22	23	24	25	26	27	28	29	...
...						...				
22		10	0	0	0	70	0	0	0	
23		0	0	0	0	0	0	0	0	
24		0	0	10	0	0	0	0	0	
25	...	0	0	0	0	0	0	0	0	...
26		0	0	0	0	10	0	0	0	
27		0	0	0	0	0	0	0	0	
28		0	0	70	0	0	0	10	0	
29		0	0	0	0	0	0	0	0	
...						...				

a_L=Turn Left

S \ S'	...	14	15	16	17	18	19	20	21	...
...					...					
14		5	90	5	0	0	0	0	0	
15		0	5	90	5	0	0	0	0	
16		5	0	5	90	0	0	0	0	
17	...	90	5	0	5	0	0	0	0	...
18		0	0	0	0	5	90	5	0	
19		0	0	0	0	0	5	90	5	
20		0	0	0	0	5	0	5	90	
21		0	0	0	0	90	5	0	5	
...					...					

a_R=Turn Right

S \ S'	...	14	15	16	17	18	19	20	21	...
...					...					
14		5	0	5	90	0	0	0	0	
15		90	5	0	5	0	0	0	0	
16		5	90	5	0	0	0	0	0	
17	...	0	5	90	5	0	0	0	0	...
18		0	0	0	0	5	0	5	90	
19		0	0	0	0	90	5	0	5	
20		0	0	0	0	5	90	5	0	
21		0	0	0	0	0	5	90	5	
...					...					

$p(o_{ASO}|s)$

S \ ASO	0	1	2	3	4	5	6	7
...				...				
14	0	4	0	0	4	83	0	9
15	2	0	42	4	2	0	46	4
16	0	2	0	2	2	43	2	49
17	2	2	42	46	0	0	4	4
18	2	42	0	4	2	46	0	4
19	40	4	44	4	4	0	4	0
20	2	2	0	0	42	46	4	4
21	4	0	78	9	0	0	9	0
...				...				

$p(o_{LVO}|s)$

S \ LVO	0	1	2	3	4	5	6
...				...			
14	16	16	16	16	16	16	160
15	50	50	0	0	0	0	0
16	16	16	16	16	16	16	160
17	33	33	33	0	0	0	0
18	16	16	16	16	16	16	160
19	330	33	33	0	0	0	0
20	16	16	16	16	16	16	160
21	50	50	0	0	0	0	0
...				...			

$p(o_{DVO}|s)$

S \ DVO	0	1	2	3
...				
14	25	25	25	25
15	100	0	0	0
16	25	25	25	25
17	100	0	0	0
18	25	25	25	25
19	100	0	0	0
20	25	25	25	25
21	100	0	0	0
...				

Fig. 10. Initial POMDP compilation, and structure of the resulting transition and observation matrixes. Parameters over gray background will be adjusted by the learning system.

Finally, the compiler generates the initial transition and observation matrixes using the predefined uncertainty rules. Settable parameters are shown over gray background in figure 10, while the rest of them will be excluded of the training process. The choice of settable parameters is justified in the following way:

- Transition probabilities. Uncertainties for actions "Turn Left" (a_L), "Turn Right" (a_R), "Go out room" (a_O) and "Enter room" (a_E) depends on odometry and the developed algorithms, and can be considered environment independent. However, the "Follow corridor" (a_F) action highly depends on the ability of the vision system to segment doors color, that can change from one environment to another. As a pessimistic initialization rule, we use a 70% probability of making the ideal "follow" transition (F), and 10% probabilities for autotransition (N), and two (FF) or three (FFF) successive transitions, while the rest of possibilities are 0. However, these probabilities will be adjusted by the learning system to better fit real environment conditions. In this case, instead of learning each individual transition probability, the general rule (values for N, F, FF and FFF) will be trained (so, transitions that initially are 0 will be kept unchanged). The new learned values are used to recompile the rows of the transition matrix corresponding to corridor states aligned with corridor directions (the only ones in which the "Follow Corridor" action is defined).
- Observation probabilities. The Abstract Sonar Observation can be derived from the graph, the state of doors, and a model of the sonar sensor characterizing its probability of perceiving "occupied" when "free" or vice versa. The last one is no environment dependent, and the state of doors can change with high frequency. So, the initial model contemplates a 50% probability for states "closed" and "opened" of all doors. During the learning process, states containing doors will be updated to provide the system with some memory about past state of doors. Regarding the visual observations, it's obvious that they are not intuitive for being predefined by the user or deduced from the graph. So, in corridor states aligned with corridor direction, the initial model for both visual observations consists of a uniform distribution, and the probabilities will be later learned from robot experience during corridor following in the exploration stage.

As a resume, the parameters to be adjusted by the learning system are:

- The general rules N, F, FF and FFF for the "Follow Corridor" action. Their initial values are shown in table I.
- the probabilities for the Abstract Sonar Observation of corridor states in which there is a door in left, right or front directions (to endow the system with some memory about past door states, improving the localization system results). Initially, it's supposed a 50% probability for "opened" and "closed" states. In this case, the adjustment will use a low gain because the state of doors can change with high frequency.
- The probabilities for the Landmark Visual Observation and Deep Visual Observation of corridor states aligned with corridor direction, that are initialized as uniform distributions.

8.2. Training data collection

Learning Markov models of partially observable environments is a hard problem, because it involves inferring the hidden state at each step from observations, as well as estimating the transition and observation models, while these two procedures are mutually dependent.

The EM algorithm (in Hidden Markov Models context known as Baum-Welch algorithm) is an expectation-maximization algorithm for learning the parameters (entries of the transition and observation probabilistic models) of a POMDP from observations (Bilmes, 1997). The input for applying this method is an execution trace, containing the sequence of actions-observations executed and collected by the robot at each execution step t=1...T (T is the total number of steps of the execution trace):

$$\mathbf{trace} = [o_1, a_1, o_2, a_2, \ldots, o_t, a_t, \ldots, o_{T-1}, a_{T-1}, o_T] \tag{14}$$

The EM algorithm is a hill-climbing process that iteratively alternates two steps to converge to a POMDP that locally best fits the trace. In the E-Step (expectation step), probabilistic estimates for the robot states (locations) at the various time steps are estimated based on the currently available POMDP parameters (in the first iteration, they can be uniform matrixes). In the M-Step (maximization step), the maximum likelihood parameters are estimated based on the states computed in the E-step. Iterative application of both steps leads to a refinement of both, state estimation, and POMDP parameters.

The limitations of the standard EM algorithm are well known. One of them is that it converges to a local optimum, and so, the initial POMDP parameters have some influence on the final learned POMDP. But the main disadvantage of this algorithm is that it requires a large amount of training data. As the degrees of freedom (settable parameters) increase, so does the need for training data. Besides, in order to the algorithm to converge properly, and taking into account that EM is in essence a frequency-counting method, the robot needs to traverse several times de whole environment to obtain the training data. Given the relative slow speed at which mobile robots can move, it's desirable that the learning method learns good POMDP models with as few corridor traversals as possible. There are some works proposing alternative approximations of the algorithm to lighten this problem, such as (Koening & Simmons, 1996) or (Liu et al., 2001). We propose a new method that takes advantage of human-robot interfaces of assistant robots and the specific structure of the POMDP model to reduce the amount of data needed for convergence.

To reduce the memory requirements, we take advantage of the strong topological restrictions of our POMDP model in two ways:

- All the parameters to be learned (justified in the last subsection) can be obtained during corridor following by sequences of "Follow Corridor" actions. So, it's not necessary to alternate other actions in the execution traces, apart from turn actions needed to start the exploration of a new corridor (that in any case will be excluded off the execution trace).
- States corresponding to different corridors (and different directions within the same corridor) can be broken up from the global POMDP to obtain reduced sub-POMDPs. So, a different execution trace will be obtained for each corridor and each direction, and only the sub-POMDP corresponding to the involved states will be used to calculate de EM algorithm, reducing in this way the memory requirements.

As it was shown in figure 9, there are two learning modes, that also differ in the way in which data is collected: the active learning mode during an initial exploration stage, and the passive learning mode during normal working of the navigation system.

8.2.1. Supervised active learning. Corridors exploration

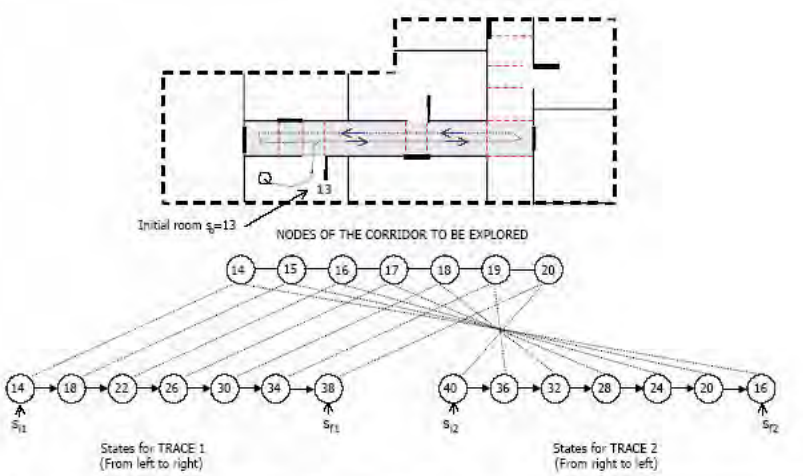
The objective of this exploration stage is to obtain training data in an optimized way to facilitate the initial adjustment of POMDP parameters, reducing the amount of data of execution traces, and the number of corridor traversals needed for convergence. The distinctive features of this exploration process are:

- The objective of the robot is to explore (active learning), and so, it independently moves up and down each corridor, collecting a different execution trace for each direction. Each corridor is traversed the number of times needed for the proper convergence of the EM algorithm (in the results section it will be demonstrated that the number of needed traversals ranges from 3 to 5).
- We introduce some user supervision in this stage, to ensure and accelerate convergence with a low number of corridor traversals. This supervision can be carried out by a non expert user, because it consists in answering some questions the robot formulates during corridor exploration, using the speech system of the robot. To start the exploration, the robot must be placed in any room of the corridor to be explored, whose label must be indicated with a talk as the following:

Robot: I'm going to start the exploration. ¿Which is the initial room?
Supervisor: dinning room A
Robot: Am I in dinning room A?
Supervisor: yes

With this information, the robot initializes its belief Bel(**S**) as a delta distribution centered in the known initial state. As the initial room is known, states corresponding to the corridor to be explored can be extracted from the graph, and broken up from the general POMDP as it's shown in figure 11. After executing an "Out Room" action, the robot starts going up and down the corridor, collecting the sequences of observations for each direction in two independent traces (trace 1 and trace 2 of figure 11). Taking advantage of the speech system, some "certainty points" (CPs) are introduced in the traces, corresponding to initial and final states of each corridor direction. To obtain these CPs, the robot asks the user *"Is this the end state of the corridor?"* when the belief of that final state is higher than a threshold (we use a value of 0.4). If the answer is "*yes*", a CP is introduced in the trace (flag cp=1 in figure 11), the robot executes two successive turns to change direction, and introduces a new CP corresponding to the initial state of the opposite direction. If the answer is "no", the robot continues executing "Follow Corridor" actions. This process is repeated until traversing the corridor a predefined number of times.

Figure 11 shows an example of exploration of the upper horizontal corridor of the environment of figure 2, with the robot initially in room 13. As it's shown, an independent trace is stored for each corridor direction, containing a header with the number of real states contained in the corridor, its numeration in the global POMDP, and the total number execution steps of the trace. The trace stores, for each execution step, the reading values of ASO, LVO and DVO, the "cp" flag indicating CPs, and their corresponding "known states". These traces are the inputs for the EM-CBP algorithm shown in the next subsection.



TRACE 1	(from left to right in horizontal corridors, and from botton to top in vertical corridors)		
Header	Nº of real states	7	
	Real states in global POMDP	{14, 18, 22, 26, 30, 34, 38}	
	Nº of data in the trace (T)	37	
Data	ASO	{ 4, 5, 1, 1, 5, 2, 5, 4, 5, 1, 5, 2, 5, 5, 4, 5, 1, 1, 1, 5, 2, ... }	
	LVO	{ 3, 3, 1, 1, 1, 1, 4, 3, 3, 1, 1, 1, 4, 4, 3, 3, 1, 1, 1, 1, 1, ... }	
	DVO	{ 2, 2, 1, 1, 0, 0, 3, 2, 2, 1, 1, 0, 3, 3, 2, 2, 1, 1, 1, 0, 0, ... }	
	cp	{ 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, ... }	
	known-state	{ X, X, X, X, X, 38,14, X, X, X, X,38, 14, X, X, X, X, X, X, X,38, ... }	
TRACE 2	(from right to left in horizontal corridors, and from top to botton in vertical corridors)		
Header	Nº of real states	7	
	Real states in global POMDP	{40, 36, 32, 28, 24, 20, 16}	
	Nº of data in the trace (T)	34	
Data	ASO	{ 0, 5, 5, 4, 5, 1, 1, 5, 5, 0, 5, 4, 5, 1, 5, 0, 5, 4, 4, 5, 5, 5, ... }	
	LVO	{ 5, 4, 4, 3, 3, 2, 2, 1, 1, 5, 4, 3, 3, 2, 1, 5, 4, 3, 3, 3, 1, 1, ... }	
	DVO	{ 3, 3, 3, 2, 2, 1, 1, 1, 0, 3, 3, 2, 2, 1, 0, 3, 3, 3, 2, 2, 2, 0, 0, ... }	
	cp	{ 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1 , 0, 0, 0, 0, 0, 1, ... }	
	known-state	{40, X, X, X, X, X, X, X, X, 16, 40, X, X, X, X, 16,40, X, X, X, X, X, 16, ... }	

Fig. 11. Example of exploration of one of the corridors of the environment of figure 2 (involved nodes, states of the two execution traces, and stored data).

8.2.2. Unsupervised passive learning

The objective of the passive learning is to keep POMDP parameters updated during the normal working of the navigation system. These parameters can change, mainly the state of doors (that affects the Abstract Sonar Observation), or the lighting conditions (that can modify the visual observations or the uncertainties of "Follow Corridor" actions). Because during the normal working of the system (passive learning), actions are not selected to optimize execution traces (but to guide the robot to goal rooms), the standard EM algorithm must be applied. Execution traces are obtained by storing sequences of actions and observations during the navigation from one room to another. Because they usually correspond to only one traversal of the route, sensitivity of the learning algorithm must be lower in this passive stage, as it's explained in the next subsection.

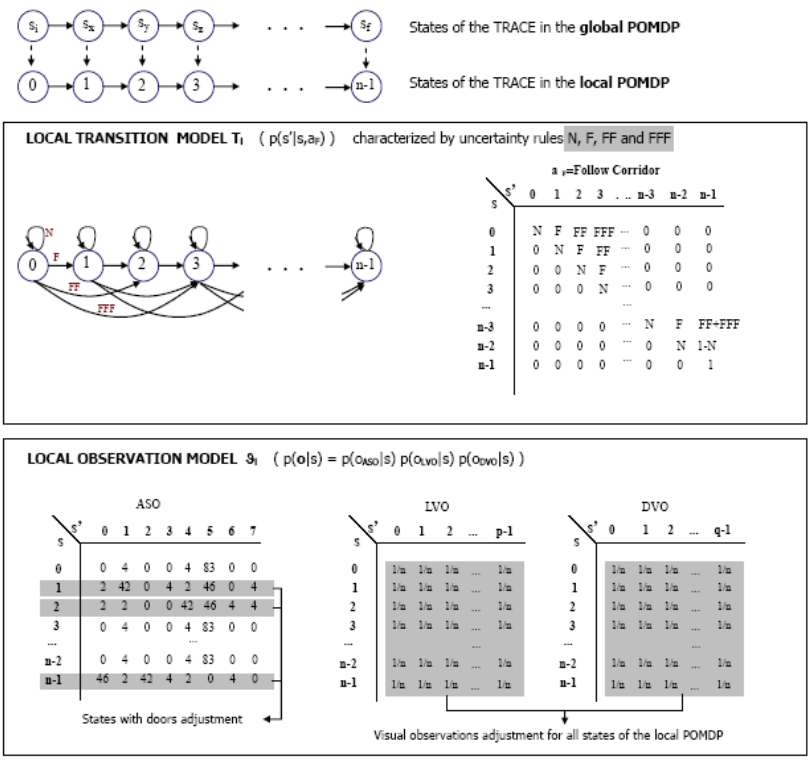


Fig. 12. Extraction of the local POMDP corresponding to one direction of the corridor to be explored.

8.3. The EM-CBP Algorithm

The EM with Certainty Break Points (EM-CBP) algorithm proposed in this section can be applied only in the active exploration stage, with the optimized execution traces. In this learning mode, an execution trace corresponds to one of the directions of a corridor, and involves only "Follow Corridor" actions.

The first step to apply the EM-CBP to a trace is to extract the local POMDP corresponding to the corridor direction from the global POMDP, as it's shown in figure 12. To do this, states are renumbering from 0 to n-1 (n being the number of real states of the local POMDP). The local transition model T_l contains only the matrix corresponding to the "Follow Corridor" action (probabilities $p(s'|s,a_F)$, whose size for the local POMDP is (n-1)x(n-1), and can be constructed from the current values of N, F, FF and FFF uncertainty rules (see figure 12). The local observation model ϑ_l also contains only the involved states, extracted from the global POMDP, as it's shown in figure 12.

The main distinguishing feature of the EM with Certainty Break Points algorithm is that it inserts delta distributions in *alfa* and *beta* (and so, *gamma*) distributions of the standard EM algorithm, corresponding to time steps with certainty points. This makes the algorithm to converge in a more reliable and fast way with shorter execution traces (and so, less corridor traversals) than the standard EM algorithm, as will be demonstrate in the results section.
Figure 13 shows the pseudocode of the EM-CBP algorithm. The expectation and maximization steps are iterated until convergence of the estimated parameters. The stopping criteria is that all the settable parameters remain stable between iterations (with probability changes lower than 0.05 in our experiments).
The update equations shown in figure 13 (items 2.4 and 2.5) differ from the standard EM in that they use Baye's rule (Dirichlet distributions) instead of frequencies. This is because, although both methods produce asymptotically the same results for long execution traces, frequency-based estimates are not very reliable if the sample size is small. So, we use the factor K (K>0) to indicate the confidence in the initial probabilities (the higher the value, the higher the confidence, and so, the lower the variations in the parameters). The original re-estimation formulas are a special case with K=0. Similarly, leaving the transition probabilities unchanged is a special case with K→∞.

EM-CBP algorithm
(this code must be applied to each execution trace)

1. Extraction of the local POMDP (T_1 and S_1)

2. **Repeat** until convergence of parameters of (2.4) and (2.5)
   2.1. Calculate alfa values (forward propagation)
   For t=0 **to** t=T-1
   If cp(t)=1 //Certainty Point
   $\alpha(t, known\_state(t))=1$, and the rest columns of $\alpha(t)$ are 0
   else
   $\alpha(t, s') = p(o(t) \mid s') \cdot \sum_{s \in S_1} p(s' \mid s, a_t) \cdot \alpha(t-1, s) \quad \forall s' \in S_1$
   Normalize $\alpha(t)$
   2.2. Calculate beta values (backward propagation)
   For t=T-1 **to** t=0
   If cp(t)=1 //Certainty Point
   $\beta(t, known\_state(t))=1$ and the rest columns of $\beta(t)$ are 0
   else
   $\beta(t, s) = \sum_{s' \in S_1} p(s' \mid s, a_t) \cdot p(o(t+1) \mid s') \cdot \beta(t+1, s') \quad \forall s \in S_1$
   Normalize $\beta(t)$
   2.3. Calculate gamma values
   For t=0 **to** t=T-1
   $\gamma(t, s)=\alpha(t, s)\cdot\beta(t, s) \quad \forall s \in S_1$
   Normalize $\gamma(t)$
   2.4. Update observation parameters (apply independently to each observation o_i)

$$p(o_i \mid s) = \left(K \cdot p(o_i \mid s) + \sum_{t=0 \,\wedge\, o(t)=o_i}^{T-1} \gamma(t, s) \right) \Big/ \left(K + \sum_{t=0}^{T-1} \gamma(t, s) \right) \quad \forall s \in S_1 \quad \text{and} \quad \forall o_i$$

   2.5. Update uncertainty rules of "Follow Corridor" action
   n = (K · n + n° times(max(γ(t)) − max(γ(t − 1)) = 0))/(K + T)
   f = (K · f + n° times(max(γ(t)) − max(γ(t − 1)) = 1))/(K + T)
   ff = (K · ff + n° times(max(γ(t)) − max(γ(t − 1)) = 2))/(K + T)
   fff = (K · fff + n° times(max(γ(t)) − max(γ(t − 1)) = 3))/(K + T)

3. Return the adjusted parameters to the corresponding rows of the global POMDP

Fig. 13. Pseudocode of the EM-CBP algorithm.

In practice, we use different values of K for the different settable parameters. For example, as visual observations are uniformly initialized, we use K=0 (or low values) to allow convergence with a low number of iterations. However, the adjustment of Abstract Sonar Observations corresponding to states with doors must be less sensitive (we use K=100), because the state of doors can easily change, and all the probabilities must be contemplated with relative high probability. During passive learning we also use a high value of K (K=500), because in this case the execution traces contain only one traversal of the route, and some confidence about previous values must be admitted.
The final step of the EM-CBP algorithm is to return the adjusted parameters from the local POMDP to the global one. This is carried out by simple replacing the involved rows of the global POMDP with their corresponding rows of the learned local POMDP.

9. Results

To validate the proposed navigation system and test the effect of the different involved parameters, some experimental results are shown. Because some statistics must be extracted and it's also necessary to validate the methods in real robotic platforms and environments, two kind of experiments are shown. Firstly, we show some results obtained with a simulator of the robot in the virtual environment of figure 2, in order to extract some statistics without making long tests with the real robotic platform. Finally, we´ll show some experiments carried out with the real robot of the SIRAPEM project in one of the corridors of the Electronics Department.

9.1. Simulation results

The simulation platform used in these experiments (figure 14) is based on "Saphira" commercial software (Konolige & Myers, 1998) provided by ActivMedia Robotics, that includes a very realistic robot simulator, that very closely reproduces real robot movements and ultrasound noisy measures on a user defined map. A visual 3D simulator using OpenGL software has been added to incorporate visual observations. Besides, to test the algorithms in extreme situations, we have incorporated to the simulator some methods to increase the non-ideal effect of actions, and noise in observations (indeed, these are higher that in real environment tests). So, simulation results can be reliably extrapolated to extract realistic conclusions about the system.

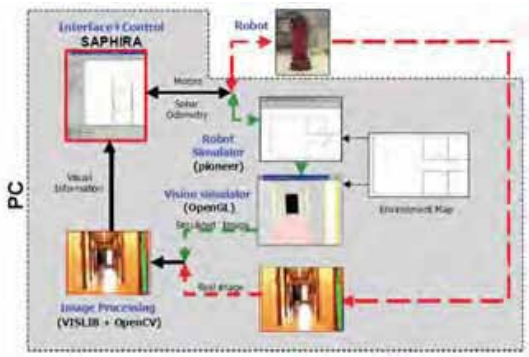


Fig. 14. Diagram of test platforms: real robot and simulator.

There are some things that make one world more difficult to navigate than another. One of them is its degree of perceptual aliasing, that substantially affects the agent's ability for localization and planning. The localization and two-layered planning architecture proposed in this work improves the robustness of the system in typical "aliased" environments, by properly combining two planning contexts: guidance and localization. As an example to demonstrate this, we use the virtual aliased environment shown in figure 2, in which there are two identical corridors. Firstly, we show some results about the learning system, then some results concerning only the localization system are shown and finally we include the planning module in some guidance experiments to compare the different planning strategies.

9.1.1. Learning results

The objective of the first simulation experiment is to learn the Markov model of the sub-POMDP corresponding to the upper horizontal corridor of the environment of figure 2, going from left to right (so, using only the trace 1 of the corridor). Although the global graph yields a POMDP with 94 states, the local POMDP corresponding to states for one direction of that corridor has 7 states (renumbered from 0 to 6), and so, the sizes of the local matrixes are: 7x7 for the transition matrix $p(s'|s,a_F)$, 7x4 for the Deep Visual Observation matrix $p(o_{DVO}|s)$, and 7x8 for the Abstract Sonar Observation matrix $p(o_{ASO}|s)$. The Landmark Visual Observation has been excluded off the simulation experiments to avoid overloading the results, providing similar results to the Deep Visual Observation. In all cases, the initial POMDP was obtained using the predefined uncertainty rules of table 1. The simulator establishes that the "ideal" model (the learned model should converge to it) is that shown in table 2. It shows the "ideal" D.V.O. and A.S.O. for each local state (A.S.O. depends on doors states), and the simulated non-ideal effect of "Follow Corridor" action, determined by uncertainty rules N=10%, F=80%, FF=5% and FFF=5%.

Local states	0 →	1 →	2 →	3 →	4 →	5 →	6
DVO:	4	4	3	3	2	1	1
ASO:	5	5,1	5,4	5	5,1,4,0	5	2,0
Follow Corridor rules: N=10%, F=80%, FF=5%, FFF=5%							

Table 2. Ideal local model to be learned for upper horizontal corridor of figure 2.

In the first experiment, we use the proposed EM-CBP algorithm to simultaneously learn the "follow corridor" transition rules, D.V.O. observations, and A.S.O. observations (all doors were closed in this experiment, being the worst case, because the A.S.O doesn't provide information for localization during corridor following). The corridor was traversed 5 times to obtain the execution trace, that contains a CP at each initial and final state of the corridor, obtained by user supervision. Figure 15 shows the learned model, that properly fits the ideal parameters of table 2. Because K is large for A.S.O. probabilities adjustment, the learned model still contemplates the probability of doors being opened. The graph on the right of figure 15 shows a comparison between the real states that the robot traversed to obtain the execution trace, and the estimated states using the learned model, showing that the model properly fits the execution trace.
Figure 16 shows the same results using the standard EM algorithm, without certainty points. All the conditions are identical to the last experiment, but the execution trace was

obtained by traversing the corridor 5 times with different and unknown initial and final positions. It's shown that the learned model is much worse, and its ability to describe the execution trace is much lower.

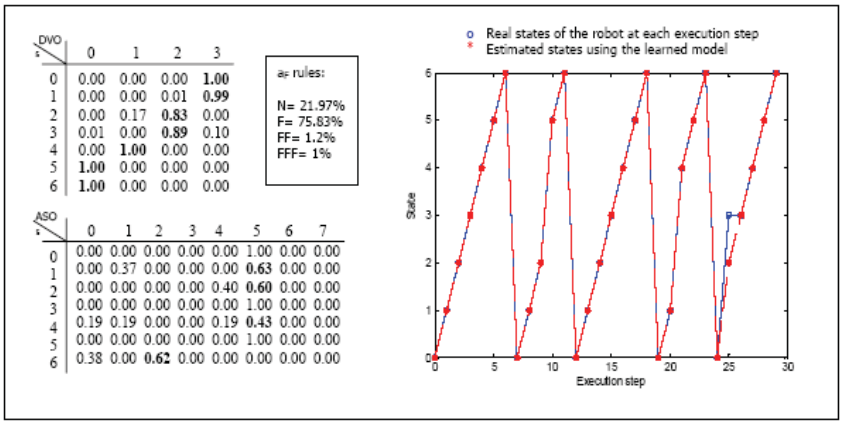


Fig. 15. Learned model for upper corridor of figure 2 using the EM-CBP algorithm.

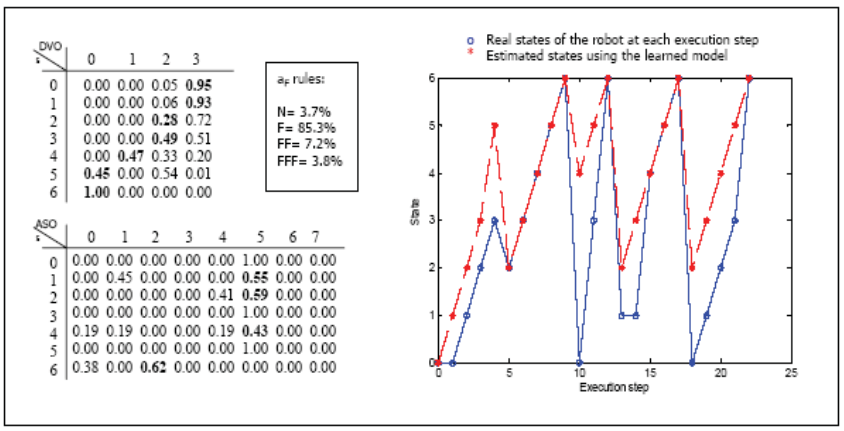


Fig. 16. Learned model for upper corridor of figure 2 using the standard EM algorithm.

Table 3 shows some statistical results (each experiment was repeated ten times) about the effect of the number of corridor traversals contained in the execution trace, and the state of doors, using the EM-CBP and the standard EM algorithms. Although there are several measures to determine how well the learning method converges, in this table we show the percentage of faults in estimating the states of the execution trace. Opened doors clearly improve the learned model, because they provide very useful information to estimate states in the expectation step of the algorithm (so, it's a good choice to open all doors during the active exploration stage). As it's shown, using the EM-CBP method with all doors opened

provides very good models even with only one corridor traversal. With closed doors, the EM-CBP needs between 3 and 5 traversals to obtain good models, while standard EM needs around 10 to produce similar results. In our experiments, we tested that the number of iterations for convergence of the algorithm is independent of all these conditions (number of corridor traversals, state of doors, etc.), ranging from 7 to 12.

	EM-CBP		**Standard EM**	
Nº of corridor traversals	**All doors closed**	**All doors opened**	**All doors closed**	**All doors opened**
1	37.6 %	6.0 %	57.0 %	10.2 %
2	19.5 %	0.6 %	33.8 %	5.2 %
3	13.7 %	0.5 %	32.2 %	0.8 %
5	12.9 %	0.0 %	23.9 %	0.1 %
10	6.7 %	0.0 %	12.6 %	0.0 %

Table 3. Statistical results about the effect of corridor traversals and state of doors.

9.1.2. Localization results

Two are the main contributions of this work to Markov localization in POMDP navigation systems. The first one is the addition of visual information to accelerate the global localization stage from unknown initial position, and the second one is the usage of a novel measure to better characterize locational uncertainty. To demonstrate them, we executed the trajectory shown in figure 17.a, in which the "execution steps" of the POMDP process are numbered from 0 to 11. The robot was initially at node 14 (with unknown initial position), and a number of "Follow corridor" actions were executed to reach the end of the corridor, then it executes a "Turn Left" action and continues through the new corridor until reaching room 3 door.

In the first experiments, all doors were opened, ensuring a good transition detection. This is the best assumption for only sonar operation. Two simulations were executed in this case: the first one using only sonar information for transition detection and observation, and the second one adding visual information. As the initial belief is uniform, and there is an identical corridor to that in which the robot is, the belief must converge to two maximum hypotheses, one for each corridor. Only when the robot reaches node 20 (that is an SRS) is possible to eliminate this locational uncertainty, appearing a unique maximum in the distribution, and starting the "tracking stage". Figure 17.b shows the real state assigned probability evolution during execution steps for the two experiments. Until step 5 there are no information to distinguish corridors, but it can be seen that with visual information the robot is better and sooner localized within the corridor. Figure 17.c shows entropy and divergence of both experiments. Both measures detect a lower uncertainty with visual information, but it can be seen that divergence better characterizes the convergence to a unique maximum, and so, the end of the global localization stage. So, with divergence it's easier to establish a threshold to distinguish "global localization" and "tracking" stages.

Figures 17.d and 17.e show the results of two new simulations in which doors 13, 2 and 4 were closed. Figure 17.d shows how using only sonar information some transitions are lost (the robots skips positions 3 , 9 and 10 of figure 17.a). This makes much worse the localization results. However, adding visual information no transitions are lost, and results are very similar to that of figure 17.b.

So, visual information makes the localization more robust, reducing perceptual aliasing of

states in the same corridor, and more independent of doors state. Besides, the proposed divergence uncertainty measure better characterizes the positional uncertainty that the typical entropy used in previous works.

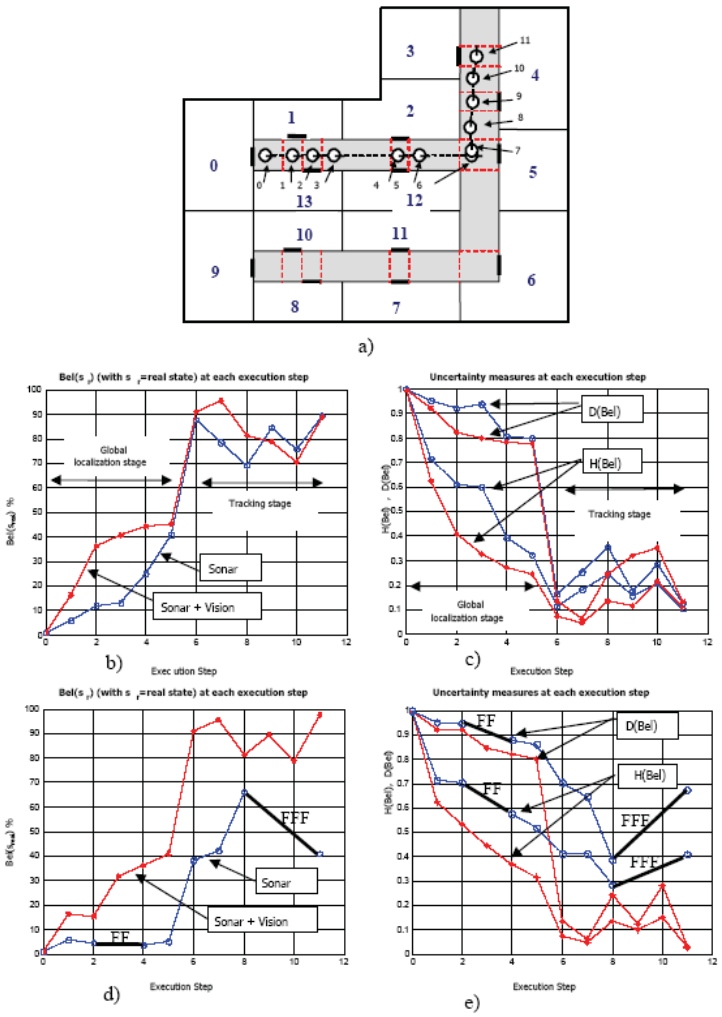


Fig. 17. Localization examples. (a) Real position of the robot at each execution step, (b) $Bel(s_{real})$ with all doors opened, with only sonar (__) and with sonar+vision (---), (c) uncertainty measures with all doors opened, (d) the same as (b) but with doors 13,2,4 closed, (e) the same as (c) but with doors 13,2,4 closed.

9.1.3. Planning Results

The two layered planning architecture proposed in this work improves the robustness of the system in "aliased" environments, by properly combining the two planning contexts: guidance and localization. To demonstrate this, we show the results after executing some simulations in the same fictitious environment of figure 17.a. In all the experiments the robot was initially at room state 0, and the commanded goal room state was 2. However, the only initial knowledge of the robot about its position is that it's a room state (initial belief is a uniform distribution over room states). So, after the "go out room" action execution, and thanks to the visual observations, the robot quickly localizes itself within the corridor, but due to the environment aliasing, it doesn't know in which corridor it is. So, it should use the localization context to reach nodes 20 or 27 of figure 2, that are sensorial relevant nodes to reduce uncertainty.

ONLY GUIDANCE CONTEXT				
	Nº Actions	Final H	Final D	Final State 2
MLS	6	0.351	0.754	54.3%
Voting	17	0.151	0.098	63.8%
Q_{MDP}	15	0.13	0.095	62.3%
GUIDANCE AND LOCALIZATION CONTEXTS (always with voting global method)				
	Nº Actions	Final H	Final D	Final State 2
H(V(a)) threshold	14	0.13	0.05	83.5%
D(V(a)) threshold	13	0.12	0.04	100%
Weighted D(V(a))	13	0.12	0.04	100%

Table 4. Comparison of the planning strategies in the virtual environment of figure 17.a.

Table 4 shows some statistical results (average number of actions to reach the goal, final values of entropy and divergence and skill percentage on reaching the correct room) after repeating each experiment a number of times. Methods combining guidance and localization contexts are clearly better, because they direct the robot to node 20 before acting to reach the destination, eliminating location uncertainty, whereas using only guidance context has a unpredictable final state between rooms 2 and 11. On the other hand, using the divergence factor proposed in this work, instead of entropy, improves the probability of reaching the correct final state, because it better detects the convergence to a unique maximum (global localization).

9.2. Real robot results

To validate the navigation system in larger corridors and real conditions, we show the results obtained with SIRA navigating in one of the corridors of the Electronics Department of the University of Alcalá. Figure 19 shows the corridor map and its corresponding graph with 71 states.
The first step to install the robot in this environment was to introduce the graph and explore the corridor to learn the Markov model. The local POMDP of each corridor direction contains 15 states. To accelerate convergence, all doors were kept opened during the active exploration stage. We evaluated several POMDP models, obtained in different ways:

- The initial model generated by the POMDP compiler, in which visual observations of corridor aligned states are initialized with uniform distributions.
- A "hand-made" model, in which visual observations were manually obtained

(placing the robot in the different states and reading the observations).

- Several learned POMDP models (using the EM-CBP algorithm), with different number of corridor traversals (from one to nine) during exploration.

Two "evaluation trajectories" were executed using these different models to localize the robot. In the first one, the robot crossed the corridor with unknown initial position and all doors opened, and in the second one, all doors were closed. The localization system was able to global localize the robot in less than 5 execution steps in both cases with all models. However, the uncertainty of the belief distribution was higher with worse models. Figure 18 shows the mean entropy of the belief distribution for all the evaluation trajectories. The "initial POMDP model" is the worst, because it doesn't incorporate information about visual observations. The learned model with one corridor traversal is not better that the "hand-made" one, but from two traversals, the obtained entropy and easy installation justifies the usage of the learning module. It can also be deduced that a good number of corridor traversals ranges from 2 to 4 in this case, because later adjustments of the model can be carried out during "active exploration". Because all doors were opened during exploration, as the number of corridor traversals increases, so does the evidence about opened doors in the model and so, the uncertainty in the "evaluation trajectory" with opened doors decreases, while in that with closed doors increases. So, the model adapts this kind of environment changes.

The time required for exploring one corridor with three traversals was about 5 minutes (with a medium speed of 0.3 m/s). The computation time of the EM-CBP algorithm, using the onboard PC of the robot (a 850MHz Pentium III) was 62 ms. These times are much lower that the ones obtained in Thrun's work (Thrun et al., 1998 b), in which for learning a metric map of an environment of 60x60 meters (18 times larger than our corridor), an exploration time of 15 minutes and computation time of 41 minutes were necessary.

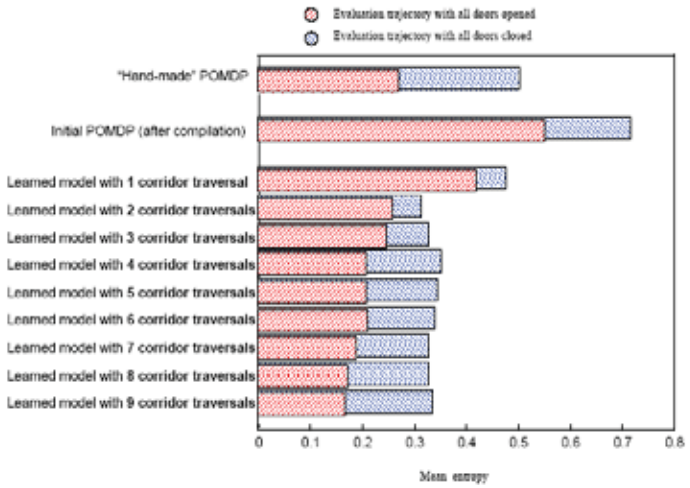


Fig. 18. Comparison of the mean entropy of the belief distribution using different models for the corridor to localize the robot in two evaluation trajectories.

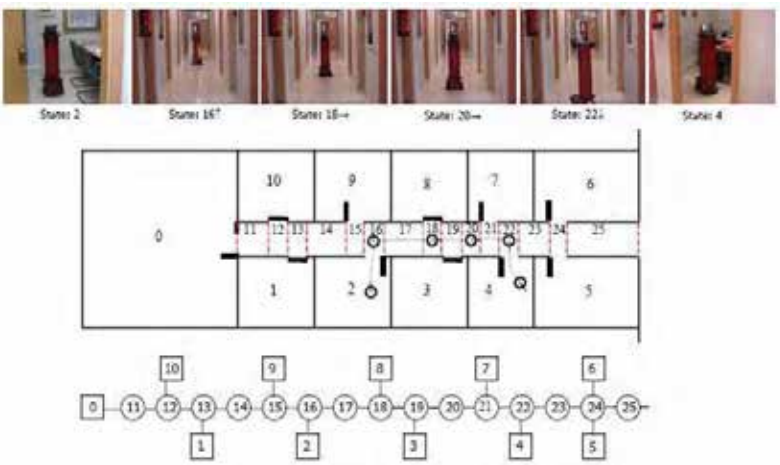


Fig. 19. Topological graph model for a corridor of the Electronics Department, and executed trajectory from room 2 to room 4 (process evolution shown in table 5).

Once shown the results of the learning process, a guidance example is included in which robot was initially in room 2 with unknown initial room state, and room 4 was commanded as goal state (see figure 19).

Process execution step	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Real robot state (node + dir)	2	16↑	16←	16↑	16→	17→	18→	19→	20→	21→	21→	22→	22↓	4
Prob. of first maximum of Bel(S) and corresponding state (node+dir) Prob. of second maximum of Bel(S) and corresponding state (node+dir)	Uniform over rooms	The same for 13↑,16↑, 19↑,22↑, 12↓,15↓, 18↓,21↓	16← 18→	16↑ 18↓	16→ 18←	17→ 16→	18→ 19→	19→ 18→	20→ 19→	21→ 22→	21→ 23→	22→ 23→	22↓ 24↓	4 5
Divergence of Bel(S) (D(Bel))	0.961	0.940	0.453	0 .453	0.113	0.290	0.067	0.055	0.082	0.453	0.473	0.231	0.100	0.097
Most voted action in GUIDANCE context (and votes in %)	O 91 %	L 51 %	R 80 %	R 80 %	F 95 %	F 98 %	F 100 %	F 100 %	F 100 %	F 74 %	F 62 %	R 97 %	E 93 %	N 94 %
Divergence of V(a) (D(V))	0.148	**0.801**	0.327	0.327	0.081	0.032	0	0	0	0.414	**0.621**	0.049	0.114	0.098
Context (LOCALIZ. if D(V)>0.5)	GUIDE	LOCAL	GUIDE	GUIDE	GUIDE	GUIDE	GUIDE	GUIDE	GUIDE	GUIDE	LOCAL	GUIDE	GUIDE	GUIDE
Most voted action in LOCALIZ. context if activated (votes in %)		L 62 %									F 67 %			
Action command selected	O	L	R	R	F	F	F	F	F	F	F	R	E	N
Real effect (transition) of action	O	L	R	R	F	F	F	F	F	N	F	R	E	

Table 3. Experimental results navigating towards room 4 with unknown initial room state (real initial room 2)

In this example, guidance and localization contexts are combined using thresholding method with divergence of probability mass as uncertainty measure. Table 5 shows, for each execution step, the real robot state (indicated by means of node number and direction), the first and second most likely states, and divergence of the belief D(Bel). It also shows the

most voted action in guidance context and the divergence of its probability mass distribution D(V). When the last one is higher than 0.5, the most voted action of localization context is used. Finally, it shows the action command selected at each process step, and the real effect (transition) of actions from step to step.
It can be seen that after going out the room, localization context is activated and the robot turns left, in the opposite direction of the destination, but to the best direction to reduce uncertainty. After this movement, uncertainty is reduced, and starts the movement to room 4. The trajectory shown with dotted line in figure 10 was obtained from odometry readings, and show the real movement of the robot. As a global conclusion, divergence factor and context combination reduces the number of steps the robot is "lost", and so the goal reaching time.

10. Discussion and Future Work

The proposed navigation system, based on a topological representation of the world, allows the robot to robustly navigate in corridor and structured environments. This is a very practical issue in assistance applications, in which robots must perform guidance missions from room to room in environments typically structured in corridors and rooms, such as hospitals or nursing homes. Although the topological map consists of very simple and reduced information about the environment, a set of robust local navigation behaviors (the actions of the model) allow the robot to locally move in corridors, reacting to sensor information and avoiding collisions, without any previous metric information.
Another important subject in robot navigation is robustness in dynamic environments. It is demonstrated that topological representations are more robust to dynamic changes of the environment (people, obstacles, doors state, etc.) because they are not modelled in the map. In this case, in which local navigation is also based on an extracted local model of the corridor, the system is quite robust to people traversing the corridor. People are another source of uncertainty in actions and observations, which is successfully treated by the probabilistic transition and observation models. Regarding doors state, the learning module adapts the probabilities to its real state, making the system more robust to this dynamic aspect of the environment.
In order to improve the navigation capabilities of the proposed system, we are working on several future work lines. The first one is to enlarge the action and observation sets to navigate in more complex or generic environments. For example, to traverse large halls or unstructured areas, a "wall-following" or "trajectory-following" action would be useful. Besides, we are also working on the incorporation of new observations from new sensors, such as a compass (to discriminate the four orientations of the graph) and a wireless signal strength sensor. Enlarging the model doesn't affect the proposed global navigation algorithms. Regarding the learning system, future work is focused on automatically learning the POMDP structure from real data, making even easier the installation process.
Another current research lines are the extension of localization, planning and learning probabilistic algorithms to multi-robot cooperative systems (SIMCA project) and the use of hierarchical topological models to expand the navigation system to larger structured environments.

11. Conclusion

This chapter shows a new navigation architecture for acting in uncertain domains, based on

a POMDP model incorporating simple visual information. This new sensor provides better information to state transition and observation models, making possible a faster global localization when the initial position of the robot is unknown and a more robust navigation. This chapter also shows a new planning architecture for acting in uncertain domains. Instead of using POMDP exact solutions, we propose an alternative two-level layered architecture that simplifies the selection of the final action, combining several planning objectives. As local policies we propose a guidance context, whose objective is to reach the goal, and a localization context to reduce location uncertainty when necessary. As global policies, we have adopted some heuristic strategies proposed in previous works. Regarding the learning system, a new method based on a modification of EM algorithm and human-robot cooperation reduces the number of needed corridor traversals. We have demonstrated the validity of this architecture in highly aliased environments and in a real environment using the robot prototype of the SIRAPEM project.

12. Acknowledgements

The authors wish to acknowledge the contribution of the *Ministerio de Ciencia y Tecnología (MCyT)* for SIRAPEM project financing (DPI2002-02193), the *Comunidad de Madrid* and *University of Alcalá* for SIMCA project financing (CAM-UAH2005/018), and the *Comunidad de Madrid* for ROBOCITY2030 project financing (S-0505/DPI/000176).

13. References

Asoh, H.; Motomura, Y.; Hara, I.; Akaho, S.; Hayamizu, S. & Matsui, T. (1996). Combining probabilistic map and dialog for robust life-long office navigation, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'96)*, pp. 807-812.

Bellman, R. (1957). Dynamic Programming. Princeton University Press.

Bilmes, J.A. (1997). Gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and Hidden Markov Models. Technical Report, University of Berkeley, ICSI-TR-97-021

Cassandra, A. (1994). Optimal policies for partially observable Markov Decision Processes. Technical Report CS-94-14, Brown University, Department of Computer Science, Providence, RI

Gechter, F.; Thomas, V. & Charpillet, F. (2001). Robot Localization by Stochastic Vision Based Device. *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics, SCI 2001*.

Howard, R.A. (1960) Dynamic Programming and Markov Processes. *The MIT Press*, Cambridge, Massachusetts.

Kaelbling, L.P.; Cassandra, A.R. & Kurien, J.A. (1996). Acting under uncertainty: discrete bayesian models for mobile robot navigation, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*

Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, Vol.101, pp. 99-134

Koenig, S. & Simmons, R. (1996). Unsupervised Learning of Probabilistic Models for Robot Navigation, *Proceedings of the International Conference on Robotics and Automation*, pp. 2301-2308

Koenig, S. & Simmons, R. (1998). Xavier: a robot navigation architecture based on partially observable Markov decision process models. *Artificial Intelligence and Mobile Robots*, 91-122.

Konolige, K. & Myers, K. (1998). The Saphira architecture for autonomous mobile robots. *Artificial Intelligence and Mobile Robots,* pp. 211-242

Liu, Y.; Emery, R.; Chakrabarti, D.; Burgard, W. & Thrun, S. (2001). Using EM to learn 3D models with mobile robots, *Proceedings of the International Conference on Machine Learning (ICML)*

López, E.; Barea, R.; Bergasa, L.M. & Escudero, M.S. (2004). A human-robot cooperative learning system for easy installation of assistant robots in new working environments. *Journal of Intelligent and Robotic Systems*, Vol. 40, pp. 233-265.

Lovejoy, W.S. (1991). A survey of algorithmic methods for partially observed Markov decision processes. Annals of Operations Research, 28(1), pp. 47-65

Montemerlo, M.; Pineau, J.; Roy, N.; Thrun, S. & Verma, V. (2002). Experiences with a Mobile Robotic Guide for the Elderly, *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada

Nourbakhsh, I.; Powers, R. & Birchfield, S. (1995). DERVISH: an office navigating robot. *Artificial Intelligence Magazine*, 16(2)

Papadimitriou, C. & Tsitsiklis, J. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research*, Vol. 12, No.3, pp. 441-450

Pineau, P. & Thrun, S. (2002). An integrated approach to hierarchy and abstraction for POMDPs. *Technical Report CMU-RI-TR-02-21,* Carnagie Mellon University, Pittsburgh, PA

Pineau, J.; Gordon, G. & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. *International Joint Conference on Artificial Intelligence (IJCAI),* Acapulco, Mexico, pp. 1025-1032

Puterman, M.L. (1994). Markov Decision Processes-Discrete Stochastic Dynamic Programming, John Wiley & Sons, Inc., New York, NY

Regini, L; Tascini, G. & Zingaretti, P. (2002). Appearance-based Robot Navigation. *Proceedings of Artificial Inteligence and Intelligent Agents (AIIA 2002)*

Roy, N.; Baltus, G.; Gemperle, F.; Goetz, J.; Hirsch, T.; Magaritis, D.; Montemerlo, M.; Pineau, J.; Schulte, J. & Thrun, S. (2000). Towards personal service robots for the elderly, *Proceedings of the Workshop on Interactive Robotics and Entertainment (WIRE),* Pittsburgh, PA

Roy, N. (2003). Finding aproximate POMDP solutions through belief compression. PhD Thesis, CMU-RI-TR-03-25. Robotics Institute, Carnegie Mellon University.

Theocharous, G. & Mahadevan, S. (2002). Approximate planning with hierarchical partially observable markov decision processes for robot navigation, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*

Thrun, S.; Gutmann, J.; Fox, D.; Burgard, W. & Kuipers, B. (1998). Integrating topological

and metric maps for mobile robot navigation: a statistical approach, *Proceedings of the National Conference on Artificial Intelligence*, pp. 989-995

Thrun, S.; Burgard, W. & Fox, D. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine learning and autonomous robots (joint issue),* 31(5), pp. 1-25.

Winston, P.H. (1984). Artificial Intelligence. Addison-Wesley.

Zanichelli, F. (1999). Topological maps and robust localization for autonomous navigation. *IJCAI Workshop on Adaptive spatial representations of dynamic environments.*

14

Robust Autonomous Navigation and World Representation in Outdoor Environments

Favio Masson†, Juan Nieto‡, José Guivant‡, Eduardo Nebot‡
†Instituto de Investigaciones en Ingeniería Eléctrica, Universidad Nacional del Sur
Argentina
‡ARC Centre of Excellence for Autonomous Systems, University of Sydney
Australia

1. Introduction

Reliable localisation is an essential component of any autonomous vehicle system. The basic navigation loop is based on dead reckoning sensors that predict high frequency vehicle manoeuvres and low frequency absolute sensors that bound positioning errors. The problem of localisation given a map of the environment or estimating the map knowing the vehicle position has been addressed and solved using a number of different approaches. A related problem is when neither, the map nor the vehicle position is known. In this case the vehicle, with known kinematics, starts in an unknown location in an unknown environment and proceeds to incrementally build a navigation map of the environment while simultaneously using this map to update its location. In this problem, vehicle and map estimates are highly correlated and cannot be obtained independently of one another. This problem is usually known as Simultaneous Localisation and Map Building (SLAM).
As an incremental algorithm, the SLAM in large outdoor environments must address several particular problems: the perception of the environment and the nature of features searched or observables with the available sensors, the number of features needed to successfully localise, the type of representation used for the features, a real time management of the map and the fusion algorithm, the consistency of the SLAM process and the data association between features mapped and observations. A good insight into the SLAM problem can be found in Durrant-Whyte & Bailey (2006).
This chapter presents recent contributions in the areas of perception, representation and data fusion, focusing on solutions that address the real time problem in large outdoor environments. Topics such as DenseSLAM, Robust Navigation and non-Gaussian Observations in SLAM are summarised and illustrated with real outdoor tests.

2. Detailed environment representation

One of the main issues of the SLAM problem is how to interpret and synthesize the external sensory information into a representation of the environment that can be used by the mobile robot to operate autonomously. Traditionally, SLAM algorithms have relied on sparse

environment representations: maps built up of isolated landmarks observed in the environment (Guivant et al., 2002; Neira & Tardós, 2001). However, for autonomous navigation, a more detailed representation of the environment is necessary, and the classic feature-based representation fails to provide a robot with sufficient information. While a dense representation is desirable, it has not been possible for SLAM paradigms.
The next generation of autonomous systems will be required to operate in more complex environments. A sparse representation formed only by isolated landmarks will in general not fulfil the necessities of an autonomous vehicle, and a more detailed representation will be needed for tasks such as place recognition or path planning. Furthermore, not only is a dense representation of the environment required, but also an algorithm that is able to obtain *multi-layered maps*, where each layer represents a different property of the environment, such as occupancy, traversability, elevation, etc (Lacroix et al., 2002).

2.1 DenseSLAM

Mapping techniques that are able to handle vehicle uncertainty such as EKF-SLAM are not able to obtain dense representations due to the extremely high computational burden involved. On the other hand, mapping algorithms that are able to obtain detailed representations such as Occupancy Grids (Elfes, 1989) are known to have problems coping with vehicle pose uncertainty. The concept of DenseSLAM was introduced in (Nieto et al., 2004) as *the process of simultaneous vehicle localisation and dense map building*.
DenseSLAM is then a more ambitious problem than classic feature-based SLAM. A solution for DenseSLAM will have to deal with computational and consistency issues, arising from the dual purpose of trying to obtain a dense representation while simultaneously doing localisation.
This section presents the Hybrid Metric Maps. The Hybrid Metric Maps (HYMMs) algorithm (Guivant et al., 2004; Nieto et al., 2004) presents a novel solution for addressing the mapping problem with unknown robot pose. The HYMM is a mapping algorithm that combines feature maps with other metric sensory information. The approach permits the localisation of the robot and at the same time constructs a detailed environment representation (DenseSLAM). It is also a powerful technique to solve several practical problems (Masson et al., 2005)
Rather than incorporating all the sensory information into a global map, the algorithm maintains a features map and represents the rest of the sensed data in local maps defined relative to the feature positions. A joint state vector with the vehicle pose and the feature positions is maintained and the dense maps are stored in a separate data structure. When new observations are received, the state vector is augmented with the feature positions and the rest of the information is fused into the local regions. The main difference between feature-based SLAM and DenseSLAM is that feature-based SLAM incorporates the features into the map and neglects the rest of the information, whereas DenseSLAM has the ability to maintain all the sensory information to build a detailed environment representation.
The algorithm works as follows. When the robot starts to navigate, it will extract features from the environment that will be incorporated in the state vector. The feature map will be used to partition the global map into smaller regions, Fig. 1 illustrates this

process. The dense sensory information will be represented in these local regions. Fig. 2 shows a hypothetical dense map. The figure shows the division of the global map into smaller regions and the dense multi-layer maps obtained by DenseSLAM. Each of these layers depicts different environment properties. The global dense map consists of a set of local maps defined relative to the feature positions. Fig. 3 shows a basic flow diagram of the algorithm.

The main characteristic of DenseSLAM is the local representation used to fuse the dense information. The motivation behind the relative representation is to reduce correlations between states. Using this relative representation, the states represented in a local frame become strongly correlated and the states represented in different frames become weakly correlated. This is the key which allows the decorrelation of the dense maps with the rest of the system making the representation tractable.

Since the observations of the world are taken relative to the vehicle pose, any environment representation created will be correlated with the vehicle pose. Augmenting the state vector with all the information rendered by the sensors and maintaining the correlations is infeasible due to the computational burden involved. Therefore, DenseSLAM incorporates a set of landmarks in the state vector and the rest of the sensed data is decorrelated and stored in a separate data structure.

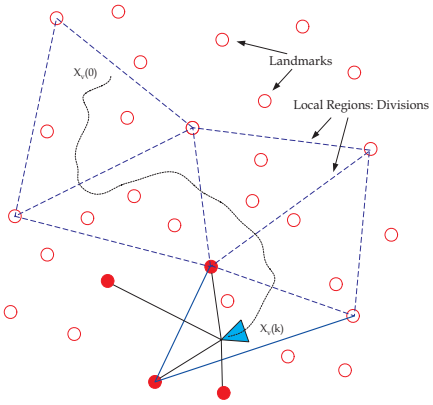


Fig. 1. Landmarks map ('o') and a particular partition of the global map in local regions. As shown, not all the landmarks are needed as vertex points in the regions definition.

The approximation made by the algorithm consists of representing the dense information in the local regions without including the correlations between the locally represented information and the rest of the system. These correlations will be zero only when there is full correlation between the local property (expressed in global coordinates) and the features that define the respective local frame (assuming the same uncertainty magnitude), so their relative positions are perfectly known. Although it can be proved that in a SLAM process the map becomes fully correlated in the limit (Gibbens et al., 2000), in practice only high correlation is achieved. However, it can be demonstrated that the assumptions made by the HYMM framework are, in practice, very good approximations for SLAM problems. The next paragraphs explain two well known properties of SLAM that justify the approximations made in the HYMMs.

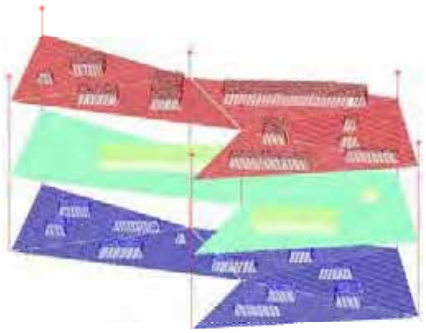


Fig. 2. Hypothetical multi-layer dense map. The '*' represent the landmark positions and the map layers depict different environment properties captured by the sensors.

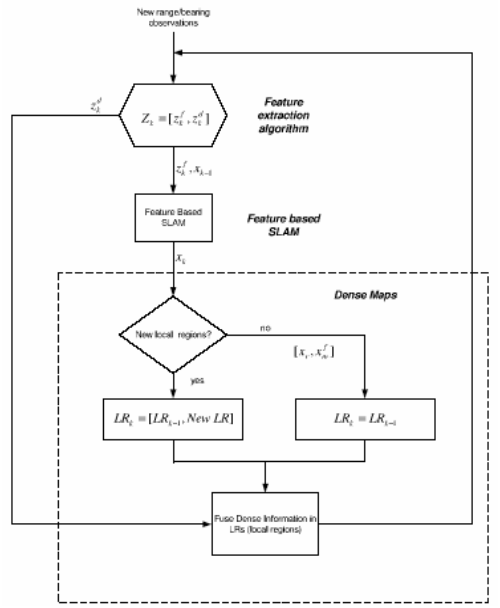


Fig. 3. HYMM algorithm flow diagram. When a sensor frame is obtained, first a feature extraction algorithm is applied and the features extracted are added to the feature-based SLAM. Then the algorithm looks for new local regions (LR) and fuses all the sensory information in the respective local frames. $\mathbf{z}_k^f$ represents the observations associated with features and $\mathbf{z}_k^d$ the rest of the observations (dense maps). $\mathbf{x}_v$ represents the vehicle position and $\mathbf{x}_m^f$ the feature map.

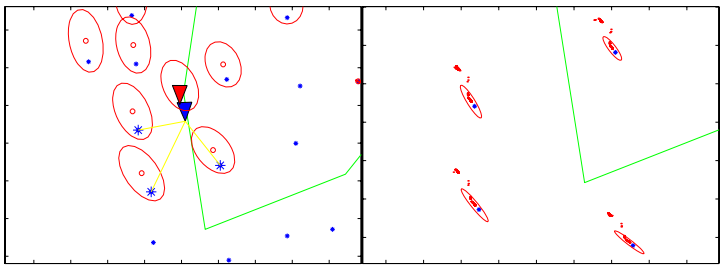
Geographically close objects have high correlation: If a set of observed objects is geographically close from the vehicle viewpoint, then the error due to the vehicle pose uncertainty will be a common

component of these estimated objects' positions. This is a typical situation in SLAM where the vehicle accumulates uncertainty in its estimated position and incorporates observations that are used to synthesize a map. Due to this fact the estimates of objects that are geographically close will present similar uncertainties (high cross-correlations). Any update of a particular object will imply a similar update of any object sufficiently close to the first one. Figure 4 shows an example of a typical SLAM map. The figure shows a landmarks map with its uncertainty bounds. It can be seen that landmarks that are geographically close have very similar uncertainty.

The relative representation stores close objects in local coordinate frames and then permits the reduction of correlation to the rest of the map (Guivant & Nebot, 2003): Assume a landmark can be represented in a local frame in the following way.

$$\mathbf{x}^L = \mathbf{h}(\mathbf{x}_v, \mathbf{z}, \mathbf{x}_b) \tag{1}$$

where $\mathbf{x}^L$ represents the relative landmark position, $\mathbf{x}_v$ the vehicle position, $\mathbf{z}$ the observations and $\mathbf{x}_b$ the position of the landmarks that define the local frame (base landmarks).



(a) (b)

Fig. 4: Map Correlation: The figures show that geographically close objects possess similar uncertainty. Figure (a) shows how the landmarks that are being observed have similar uncertainty to the robot pose. (b) shows how the estimated landmarks' means are updated after the vehicle closes the loop. The dots represent landmark position estimates over time. High correlation in geographically close objects is one of the SLAM characteristics; because the vehicle will observe close objects at similar instants it will propagate similar uncertainty to the objects.

Taking into account that the observation errors are independent of the vehicle position, the cross-correlation between the vehicle and the landmark in the local frame will be:

$$\mathbf{P}_{vL} = \mathbf{P}_{vv}\nabla\mathbf{h}_{\mathbf{x}_v}^T + \mathbf{P}_{vb}\nabla\mathbf{h}_{\mathbf{x}_b}^T \tag{2}$$

where $\mathbf{P}_{vv}$ represents the vehicle states covariance, $\mathbf{P}_{vb}$ the cross-correlation between the vehicle states and the base landmarks position estimated and $\nabla\mathbf{h}_{\mathbf{x}_i} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}_i}$ is the Jacobian matrix of h with respect to the state xi.

Taking for example the one dimensional case, Equation (1) becomes:

$$x^L = h(x_v, z, x_b) = x_v + z - x_b \tag{3}$$

Applying Equation (3) to (2):

$$\mathbf{P}_{vL} = \mathbf{P}_{vv}(1) + \mathbf{P}_{vb}(-1) = \mathbf{P}_{vv} - \mathbf{P}_{vb} \tag{4}$$

Equation (4) shows that if the magnitudes of P_{vv} and the covariance of the base landmarks P_{bb} are similar, when the robot is highly correlated with the base landmarks there will be

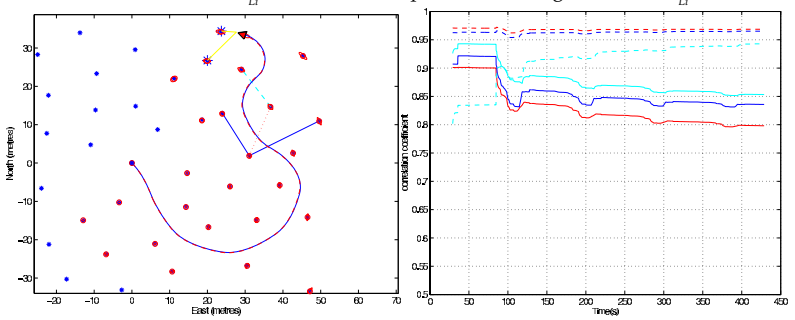
almost no correlation between the robot position and the local landmark (P_{vL}) and then no correlation between the relative landmark and the rest of the map. Since the relative and the base landmarks are geographically close, whenever the robot observes the local landmark it will be highly correlated with the base landmarks. This fact will reduce the correlation between the local landmark and the robot and therefore the correlation between the local landmark and the rest of the map will be reduced as well.

A more direct way of observing the decorrelation effect will be by evaluating the cross-correlation between a landmark in the global frame with a landmark represented in a local frame and comparing this with the cross-correlation of the same two landmarks, both represented in the global frame. In a similar manner to Equation (2), the cross-covariance matrix between the *j*-th landmark and a locally represented landmark can be evaluated in the following way:

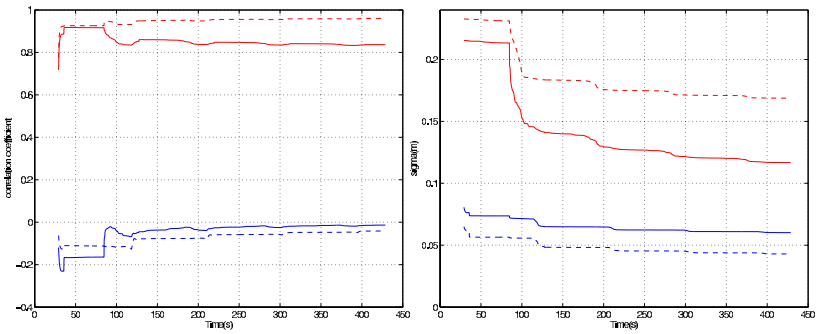
$$\mathbf{P}_{jL} = \mathbf{P}_{jb}\nabla\mathbf{h}_{\mathbf{x}_b}^{T} + \mathbf{P}_{jG}\nabla\mathbf{h}_{\mathbf{x}^G}^{T} \tag{5}$$

where the prefix *j* means the *j*-th landmark, *b* the base landmarks that define the local frame, *L* the locally represented landmark and *G* the position of the local landmark in the global frame. Then given $\mathbf{P}_{jb}$, $\mathbf{P}_{jG}$ and the transforming function from the global to the local frame **h**, it is possible to evaluate the cross-correlation between the local landmark and the *j* landmark in the map. Although the effect of decorrelation happens regardless of the particular local representation used, finding an expression to demonstrate that $\mathbf{P}_{jL} << \mathbf{P}_{jG}$ will be dependent on the local representation used. Equation (5) shows that for a particular local representation **h**, the decorrelation effect between the local object and the rest of the map will depend on the cross correlation between the rest of the map and the base landmarks and the cross-correlation between the rest of the map and the local represented object in the global frame.

Fig. 5 (a) shows the simulation environment utilised to illustrate the decorrelation effect. In the example, a local region is defined using three landmarks and another landmark is locally represented in this local frame. The *i*-th landmark is then represented in the local frame after a few observations which ensures high correlation with the base landmarks. After that, the landmark is not observed again, as it actually occurs with the dense sensory information (it is assumed that is not possible to observe exactly the same point of the environment more than once). The blue solid line in Fig. 5 (a) shows the local frame axis and the red dotted line the local landmark position $\mathbf{x}_{Li}$. The cyan dashed line joins the local represented landmark $\mathbf{x}_{Li}^{L}$ with a global landmark $\mathbf{x}_{Lj}$. The cross-correlation between $\mathbf{x}_{Lj}$ and the local *i*-th landmark will be evaluated when the *i* landmark is represented in the local frame $\mathbf{x}_{Li}^{L}$ and when it is represented in the global frame $\mathbf{x}_{Li}^{G}$.



(a) (b)



(c) (d)

Fig. 5: Decorrelation effect: (a) shows a zoom of the navigation environment where three landmarks were used to define a local region and one landmark (red dashed line) was represented in this local frame. (b) shows the correlation between the landmark represented in the global frame and the base landmarks. (c) shows the decorrelation effect; when the landmark is represented in local coordinates (blue line) the cross-correlation with other landmarks is considerably reduced in respect to the correlation between other landmarks and the landmark in global coordinates (red line). (d) shows the landmark deviation when it is represented in local (blue line) and global (red line) coordinates.

Fig. 5 (b) shows the evolution in time of the correlation coefficient of the i landmark represented in global coordinates $\mathbf{x}_{Li}^{G}$, with the landmarks used to define the local frame. The solid line depicts the cross-correlation in the east axis and the dashed line in the north axis. The different colours represent the cross-correlation with the different base landmarks. As can be seen, the landmark $\mathbf{x}_{Li}^{G}$ possesses high correlation with the base landmarks. This is due to the geographical proximity between the landmarks. Fig. 5 (c) shows the correlations between $\mathbf{x}_{Li}^{G}$ and $\mathbf{x}_{Lj}$ in red, and the correlations between the j landmark and the landmark i represented in the local frame $\mathbf{x}_{Li}^{L}$ (Equation (5)) in blue. The correlation was reduced from almost one when the landmark was represented in global coordinates, to almost zero when the landmark was represented in the local frame.

Finally Fig. 5 (d) shows the variance of the landmark i. The blue line depicts the variance when the landmark is in the local frame and the red line when it is in global. Because of the high correlation between $\mathbf{x}_{Li}^{G}$ and the base landmarks, the uncertainty in their relative position is very low, and so is the variance of $\mathbf{x}_{Li}^{L}$.

In summary the relative representation used by DenseSLAM permits the local represented information to be decorrelated with the rest of the system. This permits the incorporation of more information without increasing the computational cost.

2.2 DenseSLAM: Applications

This section shows how the detailed multi-dimensional environment description obtained by DenseSLAM can be used to improve the vehicle navigation process. Two particular applications are shown. (i) Complex landmarks can be extracted and incorporated as they

become identified using the dense representation. (ii) The dense maps can be used to estimate the variations in time of the areas explored by the robot which can be used to discriminate whether a region has potential dynamic objects.

High Level Landmarks (HLLs): One of the main problems in SLAM algorithms is the error accumulation due to non-linearities in the system. This error accumulation can be reduced if more information is added into the localisation map, since the vehicle error will remain smaller. Among the reasons to avoid including more landmarks is the computational burden required to maintain the map. However, in many situations, even when the computational cost may not be a problem, the difficulties of finding *stable and easily detectable* features cause the algorithm to use only a small number of landmarks for the localisation process, which results in a major accumulation of errors due to non-linearities.

DenseSLAM yields a rich environment representation, which gives the possibility of adding landmarks extracted from the dense maps into the landmarks map. In many situations an object cannot be detected using the measurements taken from only one vantage point. This can be due to a variety of reasons: occlusion between objects, the size of the object in relation to the sensor field of view, an inappropriate feature model, or just because the nature of the sensor makes the estimation of the landmark location impossible from only one vantage point (e.g. wide-beam sonar; Leonard J. et al. 2002, McKerrow P. 1993). Estimating partially observable features has been an important research topic in computer vision using stereo vision and bearing only information, where the initialisation of the feature position is a significant problem. The problem of partially observable features has also been studied for localisation and SLAM applications. In Leonard et al. (2002) an approach is presented that delays the decision to incorporate the observations as map landmarks. Consistent estimation is achieved by adding the past vehicle positions to the state vector and combining the observations from multiple points of view until there is enough information to validate a feature. In McKerrow (1993), intersection of constant depth range of ultrasonic sensors is used to determine the location of features from multiple vantage points.

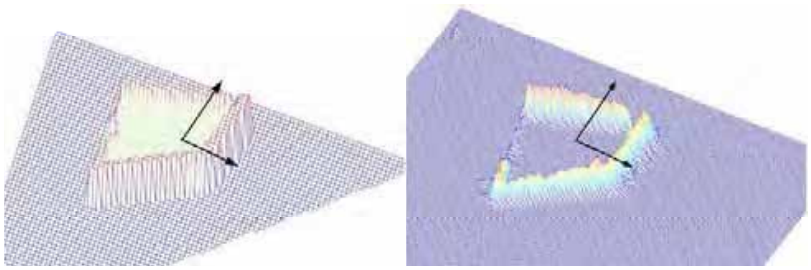
Having a comprehensive representation of the environment will enable a delayed processing to determine whether part of the map can qualify as a landmark. The rich representation obtained by DenseSLAM will enable postprocessing capabilities to continuously detect high-level landmarks using the dense map layers. The newly detected landmarks can then be added to the feature map. This approach has the potential of incorporating a large number of landmark models, some of them to be applied online at the time the observations are taken and the rest to run in the background when computer resources become available. The landmarks can then be incorporated into the features map.

High Level Landmarks representation. The only condition for the incorporation of a HLL is to represent the information in the same form as the feature map. For example, if EKF-SLAM is used, the HLLs have to be represented in state vector form.

The HLLs could be represented using geometric parameters. Experimental results of SLAM using trees as landmarks are presented in (Guivant et al., 2002). An EKF is run which estimates the trees' parameters, which consist of the centre and the diameter of the trees' trunks. In Thrun (2001) an algorithm is presented that employs expectation maximization to fit a low-complexity planar model to 3D data collected by range finders and a panoramic camera. After this model is obtained, its parameters could be added to the state vector to represent a HLL.

Fig. 6 shows an example of HLLs. In the example, the HLLs are represented as a local coordinate system and a template which is defined relative to the local axes. The templates

are formed with the information extracted from the dense maps. Scan correlation can be used to generate observations of the landmarks (see Nieto et al. 2005, for more details).



(a) (b)

Fig. 6: The figure examples of HLLs extracted from the dense maps. (a) shows a HLL represented extracted from an occupancy grid map and (b) a HLL extracted from a Sum of Gaussian dense map.

Dynamic Environments: Most of the mapping algorithms assume the world is static (Thrun, 2002). Dynamic environments require an extension of the typical representation used for static environments. That extension should allow for modelling the temporal evolution of the environment. Dynamic objects can induce serious errors in the robot localisation process. Only a few approaches that include moving objects have been presented so far. The next paragraphs review some of them.

A SLAM algorithm with generic objects (static and dynamic) is presented in (Chieh-Chih Wang, 2004). Similar to classic SLAM, the approach calculates the joint posterior over the robot and object's pose, but unlike traditional SLAM it includes also the object's motion model. The problem is shown to be computationally intractable and so a simplified version called *SLAM with Detection and Tracking of Moving Objects* (SLAM with DATMO) is presented (Chieh-Chih Wang et al., 2003). The latest algorithm decomposes the estimation process into two separate problems: (i) the SLAM problem, using static landmarks as the classic approach, and (ii) the detection and tracking of moving objects, using the robot pose estimated by the SLAM algorithm. This simplification makes updating both the SLAM and the tracking algorithm possible in real-time since they are now considered two independent filters.

In Hahnel et al. (2003) an algorithm for mapping in dynamic environments is presented. The aim of the approach is to determine which measurements correspond to dynamic objects and then filter them out for the mapping process. The approach uses the EM algorithm; the expectation step computes an estimate of which measurements might correspond to static objects. These estimates are then used in the maximization step to determine the position of the robot and the map.

An approach called *Robot Object Mapping Algorithm* (ROMA) is presented in Biswas et al. (2003). The main goal is to identify non-stationary objects and model their time varying locations. The approach assumes that objects move sufficiently slowly that they can safely be assumed static for the time it takes to build an occupancy grid map of the whole area explored by the robot. Assuming the robot is able to acquire static occupancy grid maps at different times, changes in the environment are detected using a differencing technique. The algorithm learns models of the

objects using EM. The expectation step calculates the correspondences between objects at different points in time and the maximisation step uses these correspondences to generate refined object models, represented by occupancy grid maps.

The algorithms presented in Chieh-Chih Wang et al. (2003) and Montemerlo et al. (2002) have one thing in common; they rely on pre-defined models of the specific objects they aim to track. ROMA, however, is able to learn about the shape of the objects, but the algorithm presents a number of limitations. Objects have to move slowly (it is not able to cope with fast-moving objects such as people), it is assumed the robot is able to obtain static maps at different times. The results presented include only four different objects in an environment where these objects can be perfectly segmented from a laser scan. The extension to a real environment with a larger number of objects may not be possible and will be computationally very expensive.

If navigation is the primary objective, the accurate shape of objects, or even their classification may not be important in general. What may be more useful is an algorithm able to identify observations that may be coming from objects that are not static and eliminate them from the list of observations to be used for the navigation process. Furthermore the algorithm could identify areas where it is more likely to find dynamic objects (e.g. a corridor where people walk) and then avoid their use or give a low priority to observations coming from objects in those areas.

The rich environment representation obtained by DenseSLAM allows a map layer identifying the most likely areas to possess dynamic objects to be built. As shown in Biswas et al., (2003), dynamic objects can be identified by differentiation of maps taken at different times. There are two different classes of object motions in a dynamic environment; slow motion, as for example the motion of a bin, which will be static during most of the day but will eventually be moved; and fast motion, such as people. Using DenseSLAM, and applying a straightforward differentiation, it is possible to identify regions with dynamic objects for either fast or slow motion objects.

One of the main problems with the differentiation is that maps obtained at different times will have different uncertainty. If a global map is maintained and two maps acquired at different times want to be differentiated, the uncertainty in the maps will make the matching process very difficult.

In DenseSLAM the global map is divided into smaller regions, so the whole map can be differentiated by applying differentiation between the corresponding local regions. As a consequence, the differentiation process will be prone only to local errors (which were shown to be much smaller than the global ones) eliminating detection errors due to the uncertainty between maps acquired at different moments.

A particular case where the DenseSLAM representation will not present advantages over other approaches is in decentralised multi-robot mapping. If multiple robots are used to map an area, each robot will form the map using different local regions. If the objective is to detect dynamic objects by fusing the maps built by different robots, a global map will have to be used and DenseSLAM loses advantages with respect to other approaches.

Fast motion can be captured by differentiation, in a similar way to slow motion. The main difference is that the differentiation is done over shorter periods of time and only in the region under the sensor view. As in the detection of objects with slow motion, using

DenseSLAM the differentiation is done using local regions instead of the global map. The motion detection will be included in a map layer as will the other properties captured by the sensors, then the global position of the dynamic map will be updated together with the other map properties (colour, occupancy, etc.).
It is important to note that fast motion detection can be also done with other techniques that are able to create a dense map in proximity to the vehicle position. The advantage of DenseSLAM is that the dense representation is already obtained, therefore, the detection of moving objects is a straightforward procedure that does not add computational cost.

2.3 Experimental Results

This section presents experimental results of DenseSLAM in an outdoor environment. The environment is a large area of 120 by 200 metres and the run is approximately 1 km long. The experimental platform used for the experiments is a conventional Holden UTE equipped with Sick lasers, a linear variable differential transformer sensor for the steering mechanism, back wheel velocity encoder, inertial unit and GPS.
In order to test the DenseSLAM algorithm, the GPS information was fused with the feature-based SLAM to obtain a laser image of the environment that is used as a reference to compare with the estimates by DenseSLAM. Fig. 7 shows the laser image obtained with the GPS information and the final map obtained with DenseSLAM. The dense map was obtained by fusing the raw laser observations into the local regions. The figure also shows the landmark positions.

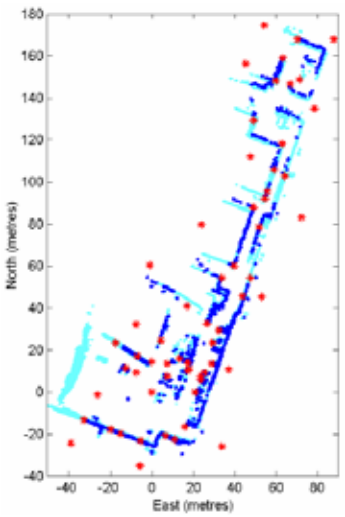


Fig. 7: Final map obtained with DenseSLAM. The light points represent the laser image obtained using GPS and SLAM. The dark points depict the dense map estimated by DenseSLAM.

Fig. 8 shows a zoom of the top part of the run. Fig. 8 (a) shows the result obtained by DenseSLAM before closing the loop. The figure also shows the laser image used as a reference. The error in the estimated map before closing the loop can be easily observed. Fig. 8 (b) shows the result after closing the first loop. Although there is still some residual error, it is clear how the estimated map has been corrected. Looking at Fig. 8 (b) it can be seen that there is some remaining uncertainty in these landmarks even after the loop is closed. This is because the vehicle does not return to the top part of the run after closing the first loop. As a result, the error in that region is not reduced as much as in the bottom part of the run. Nevertheless, an important correction in all the regions has still been made.

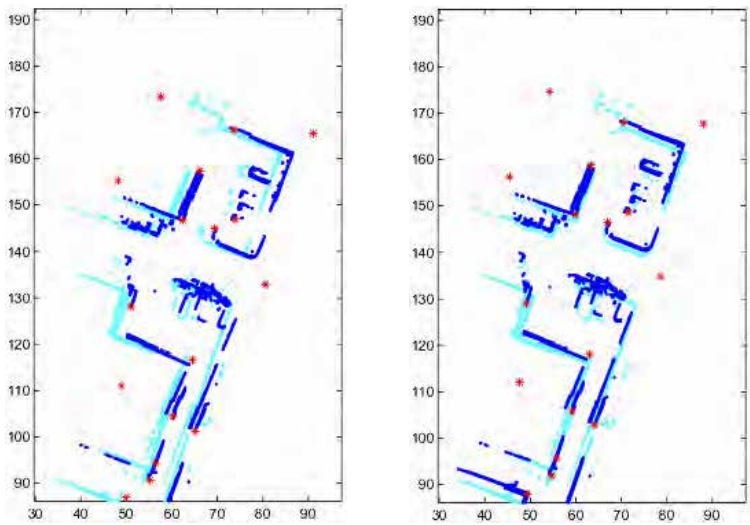


Fig. 8: The lighter points represent the laser image obtained using GPS/SLAM. The darker points represent the final map obtained with DenseSLAM. Figure (a) shows the result before closing the loop, and (b) after the loop is closed.

3. Fundamental issues when working in large areas

Most EKF implementations generate state estimations with mono-modal probability distributions and are not capable of handling multi-modal probability distributions. Multi-modal distributions are typical when closing large loops, that is, revisiting known places after a large exploration period. It is at this stage where the standard SLAM based on Kalman filters is especially fragile to incorrect association of landmarks (Neira & Tardós, 2001). Other data fusion algorithms, such as the ones that use particle filter (Montemerlo et al., 2002), can address this problem since they naturally deal with multi-hypothesis problems.

In Masson (2003) is proposed a robust data fusion algorithm, which uses a hybrid architecture. The algorithm uses Compressed EKF (CEKF in Guivant & Nebot, 2003) under normal conditions to perform SLAM. At a certain time the system may not be able to perform the association task due to large errors in vehicle pose estimation. This is an

indication that the filter cannot continue working assuming a mono-modal probability density distribution. At this time, we have the CEKF estimated mean and deviation of the states representing the vehicle pose and landmark positions. With the currently estimated map, a decorrelated map is built using a coordinate transform and a decorrelation procedure (Guivant & Nebot, 2002). A particle filter (Gordon et al., 1993) is initialised using the available statistics and is then used to resolve the position of the vehicle as a localisation problem. Once the multi-hypothesis problem is solved, the CEKF is restarted with the states values back propagated to the time when the data association problem was detected. Then the CEKF resumes operation until a new potential data association problem is detected.

There are several important implementation issues that need to be taken into account to maximise the performance of the hybrid architecture proposed. The solutions they need to consider are the uncertainties in vehicle, map and sensor to maximise the number of particles in the most likely position of the vehicle.

The SLAM algorithm builds a map while the vehicle explores a new area. The map states will be, in most cases, highly correlated in a local area. In order to use the particle filter to solve the localisation problem, a two dimensional map probability density distribution needs to be synthesised from an originally strongly correlated n dimension map. The decorrelation procedure is implemented in two steps. The map, originally represented in global coordinates is now represented in a local frame defined by the states of two beacons that are highly correlated to all the local landmarks. The other local landmarks are then referenced to this new base. A conservative bound matrix can be obtained as a diagonal matrix with bigger diagonal components and deleting the cross-correlation terms (Guivant & Nebot, 2002).

In most practical cases the local map is very large when compared to the sensor field of view. Most of the landmarks are usually beyond the range of the sensor. It is then possible to select only the *visible beacons* from the entire map by considering the estimated uncertainties. This will significantly reduce the computation complexity for the evaluation of the likelihood for each predicted particle. The boundaries of the reduced map are fixed based on the beacons that are close to the vehicle location, the particle positions, the observation and their respective uncertainty. Only a few beacons are within the field of view of any of the particles. The other beacons are not considered to be part of the reduced map.

As the number of particles affects both the computational requirements and the convergence of the algorithm, it is necessary to select an appropriate set of particles to represent the a priori density function at time T_0, that is, the time when the data association fails. Since the particle filters work with samples of a distribution rather than its analytic expression it is possible to select the samples based on the most probable initial pose of the rover. A good initial distribution is a set of particles that is dense in at least a small sub-region that contains the true states value. The initial distribution should be based in the position and standard deviations reported by the CEKF, and in at least one observation in a sub-region that contains this true state's value. In Lenser & Veloso (2000) a localisation approach is presented that replaces particles with low probability with others based on the observations. Although this algorithm is very efficient it considers that the identity of that landmark is given (known data association). This is true in some applications such as the one addressed in this work but not common in natural outdoor environments where landmarks have similar aspects and the presence of spurious objects or new landmarks is common. Here, the data association is implicitly done by the localisation algorithm.

The multi-hypotheses considered are defined by the uncertainty of the robot pose estimation. In addition the method presented is able to deal with false observations. Spurious observations and landmarks that do not belong to the map are naturally rejected by the localiser. The technique presented considers the information from a set of observations to select particles only in the initial distribution and combined with the CEKF estimates as was mentioned previously. In fact, this localisation filter is a Monte Carlo Localisation.

The initial distribution is created from range/bearing observations of a set of landmarks. This probability distribution is dominant in a region that presents a shape similar to a set of helical cylinders in the space (x, y, φ). Each helix centre corresponds to a hypothetical landmark position with its radio defined by the range observation. The landmarks considered are only the ones that the vehicle can *see* from the location reported by the CEKF and within the range and field of view of the sensors.

Although it is recognised that some observations will not be due to landmarks, all range and bearing observations in a single scan are used to build the initial distribution. Even though a set of families of helices will introduce more particles than a single family of helices (one observation), it will be more robust in the presence of spurious observations. By considering that the range/bearing observations are perfect then the dominant region becomes a discontinuous one dimensional curve (family of helices) *C*, in the three dimensional space (x, y, φ)

$$C = \bigcup_{i=1}^{N} C_i = \bigcup_{i=1}^{N} \left\{ (x, y, \varphi) \setminus \begin{array}{c} x = x(\tau) = x_i + z_r \cdot \cos(\tau) \\ y = y(\tau) = y_i + z_r \cdot \sin(\tau) \\ \varphi = \varphi(\tau) = \tau - z_\beta - \dfrac{\pi}{2} \\ \tau \in [0, 2\pi) \end{array} \right\} \tag{6}$$

These regions can be reduced by adjusting the variation of τ according to the uncertainty in φ. Assuming the presence of noise in the observations and in the landmark positions

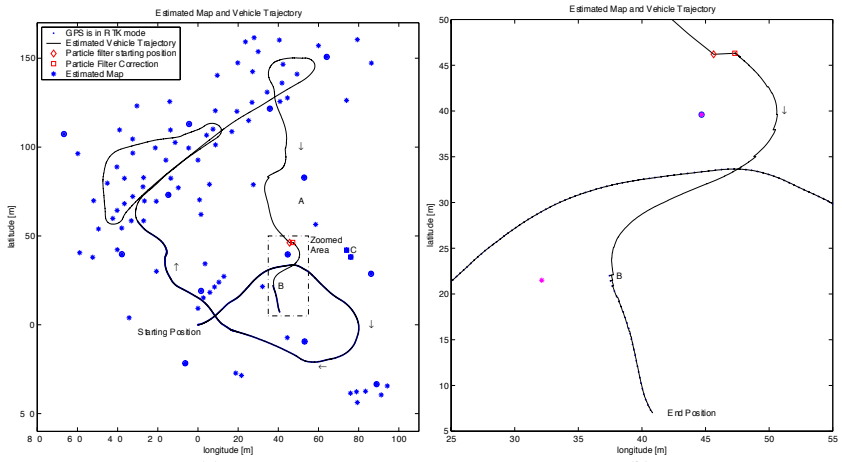
$$\begin{array}{c} z_r = z_r^* + \gamma_r, \;\; z_\beta = z_\beta^* + \gamma_\beta \\ x_i = x_i^* + \gamma_{x_i}, \;\; y_i = y_i^* + \gamma_{y_i} \end{array} \tag{7}$$

this family of helices becomes a family of cylindrical regions surrounding the helices. The helical cylinder section can be adjusted by evaluating its sensitivity to the noise sources $\gamma_{x_i}, \gamma_{y_i}, \gamma_r, \gamma_\beta$.

The same assumptions can be made for the case of using bearing only observations. Although this method can be more efficient than the standard uniform o Gaussian distribution it is still very demanding in the number of particles. A more efficient algorithm can be designed considering two observations at a time. With no data association a pair of observations will generate a family of curved cylinders to cover all possible hypotheses. This initialisation is significantly less expensive than a uniform distributed sample in a large rectangular region in the (x, y, φ) space or even a Gaussian distribution in this region. In the case of range only observations, the initialisation is very similar to the range and bearing problem. In this case the main difference is in the evaluation of the orientation (Masson et al., 2003).

Finally, two main issues need to be addressed to implement the switching strategy between the CEKF and the SIR filter. The first problem involves the detection of a potential data association failure while running the CEKF. This is implemented by monitoring the estimated error in vehicle and local map states and the results of the standard data association process. The second issue is the

reliable determination that the particle filter has resolved the multi-hypothesis problem and is ready to send the correct position to the CEKF back propagating its results. This problem is addressed by analysing the evolution of the estimated standard deviations. The filter is assumed to converge when the estimated standard deviation error becomes less than two times the noise in the propagation error model for x, y and φ. The convergence of the filter is guaranteed by the fact that the weights are bounded (Masson et al., 2003) above at any instant of time (Crisan & Doucet, 2000).
The following are results obtained using the hybrid architecture in an outdoor environment populated by trees that are used as the most relevant features to build a navigation map (Guivant et al., 2002). Full details of the vehicle and sensor model used for this experiment are available in Nebot (2002).
The CEKF filter is used to navigate when no potential data association faults are detected. When a data association failure is detected the particle filter is initialised according to the procedure presented in section 4.2 and is run until convergence is reached. At this point the filter reports the corrections to the CEKF that continues the SLAM process using EKF based methods.

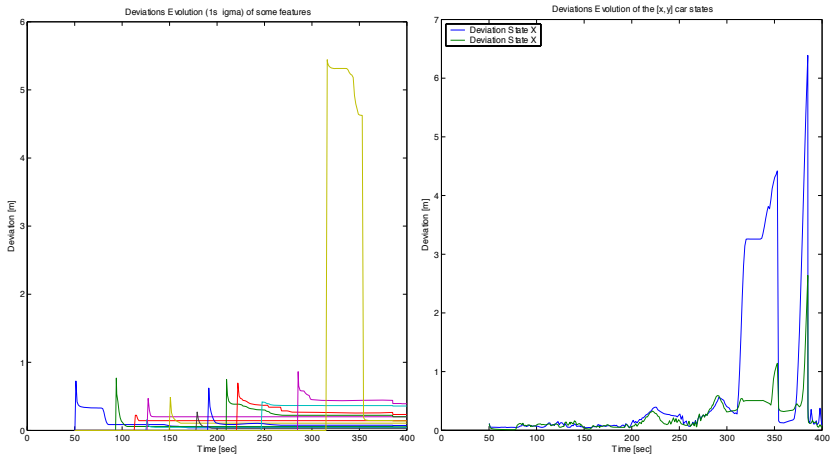


(a) (b)

Fig. 9: (a) Experimental run implementing SLAM using all the available information. (b) A zoomed area. A diamond and a square show the start and end position respectively of the particle filter correction. The dots represent the RTK GPS information.

The algorithms were tested in an environment with areas of different feature density as shown in Fig. 9. In this experiment we logged GPS, laser and dead reckoning information. The GPS used is capable of providing position information with 2 cm accuracy. This accuracy is only available in open areas and is shown in Fig. 9 with a thick line. The vehicle started at the point labelled "Starting Position" and the filter used GPS, laser and dead reckoning to perform SLAM (Guivant et al., 2002) until it reached the location at coordinates (-30,60) where GPS is no longer available. The SLAM remained operating using Laser and dead-reckoning information only. High accuracy GPS was again available close to the end of the run and will be essential to demonstrate the consistency and performance of the hybrid navigation architecture proposed.

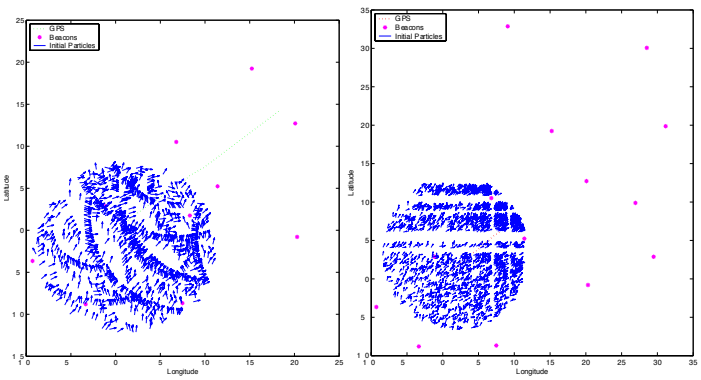
The stars and encircled stars in Fig. 9 (a) represent the natural features incorporated into the map and the selected landmarks whose deviations are shown in Fig. 10(a) respectively. A diamond and a square represent the starting and ending position resulting from the particle filter correction and are clearly shown in Fig. 9 (b). The beacons that produce the association failure are the squared stars marked as C in the figure.
Fig. 10(b) presents the vehicle position estimated error. It can be seen that the error was very small when the system was operating with GPS, *time* < *200ms*. It is then maintained below 0.5 m while in the area with high feature density. The error then started to increase before reaching point "A" since the laser cannot detect any known feature. At this time (320 sec) a new feature was incorporated but with large uncertainty as shown in Fig. 10(a). Then a known landmark was detected and since it can be associated correctly, the error in vehicle and landmark position dramatically decreased as expected.



(a) (b)

Fig. 10: Standard deviation (a) of selected beacons in the map and (b) of the car positions over time. These beacons are shown as rounded stars in Fig. 9.

A different situation is presented in Fig. 9 (b) that corresponds to the area marked as zoomed area in Fig. 9 (a). Once the laser stopped seeing the previous known landmarks the error built up again to the point where the system can no longer associate the detected landmarks to a single known landmark. The location of the vehicle at this time is represented as a diamond at coordinates (45,45) in this figure. In this case the system has to activate the Monte Carlo localiser to generate the relocalisation results shown as a square at coordinates (47,45) in the same figure.
Examples of the Monte Carlo filter initialisation are shown in Fig. 11. Fig. 11(a) shows the initialisation for the range and bearing case. The figure clearly shows the helical shape of the initial distributions. The arrows represent the position and orientation of the vehicle and the stars the beacons present in the map. The initialisation for the case of bearing only is also shown in Fig. 11(b).



(a) (b)

Fig. 11. Initialisation of the particle filter (a) using range and bearing information and (b) using bearing only information

The relocalisation result is then reported to the CEKF to continue with the SLAM process for the rest of the run. At the end of the trajectory high accuracy GPS was again available (thick line). It can be clearly seen, specially in Fig. 9 (b), that the estimated vehicle pose just before GPS became available is very close to the high accuracy GPS position reported. This demonstrates the performance and consistency of the hybrid architecture proposed.

3.1 Assimilation of non-Gaussian observations

A pure SLAM algorithm is based in measures relative to the vehicle. Nevertheless a practical application of localisation must fuse all the available sources of information that are available, included absolute information. This is a fundamental issue in navigation. Although many pure SLAM algorithms can work in large areas they could also benefit from absolute position information such as GPS. In many applications, it is not possible to obtain GPS information for long periods of time. However, at some locations this sensor will be able to report navigation data with an estimated error. It is clearly important to be able to incorporate this information to improve the localisation estimates and at the same time enable the SLAM algorithm to explore and incorporate new features while bounding the absolute pose error with the absolute information.
In order to add this information in a consistent manner some important issues need to be considered. The quality of the models and the relative navigation information used in SLAM algorithms could lead to very large innovations errors when the absolute information is fused. This occurs after long periods of navigation when only relative information is used (pure SLAM). A strong correction will make the linearisation of the models not valid generating incorrect update of covariance. The innovations may not be large but can generate strong updates in the covariance matrix. This can potentially introduce serious numerical errors. In order to prevent these problems, it is possible to treat new absolute information as L observations such that the total information introduced becomes equivalent to a single update (Guivant et al., 2002). In this case, the filter will perform L updates with the observation value and modified noise covariance. The sequential updates generate the same results as the single update but alleviate numerical problems arising from large covariance updates.

Even so, there is another potential issue that must be considered with some sensors. A typical measurement obtained from a GPS occurs when it operates in environments where there are forest and/or buildings. In open places GPS operation is usually satisfactory but is not the case in forest or *urban canyons*. The problem arises from total unavailability of satellite signals to partial occlusion and performance degradation due to multi path effects. Others sensors such as compasses present similar behaviour in static and dynamic environments where magnetic field perturbations affect the sensor operation. However there is no doubt that both sensors can provide useful information to contribute in the localisation process. In the case of range only and bearing only sensors, one measurement generates a non-Gaussian distribution and the way to deal with it is delaying the fusion collecting several measures and recording the vehicle pose (Bailey, 2002; Sola et al., 2005).

Essentially, these kinds of sensors could introduce non-Gaussian noise and some could also introduced noise correlated in time. In the case of the GPS in autonomous mode for example, the uncertainty will be introduced as a result of many factors such as satellite availability, satellites distribution, signal reflections, multi-path, atmospheric distortion, etc. It is obvious that this cannot be modelled as Gaussian, nor white. Similarly the compass usually presents biased noise due to distortion in the magnetic field, and the change depends on time and geographical position. An unknown and changing bias that varies according to the position, orientation or time represents a difficult modelling problem.

Additional to the non-Gaussian or time correlated nature of the noise, the probability distribution of the uncertainty in the observations could be unknown or only partially known. Estimators such the EKF and also any Bayesian filters cannot deal with those measurements. The improper use of them can produce inconsistent estimations. For example, if the noise is not white and this is ignored assuming that the measurements are independent, then the estimates will be over-confident. As a conservative policy these correlated measurements could be ignored to avoid inconsistent results. However in many practical applications those measurements are crucial sources of information and should be considered in a consistent way.

Consider the following situation. At time k there exists a Gaussian estimation and an available observation. This one is neither Gaussian, nor white and with partially known probability distribution, or any of these situations.

Initially it is assumed that the observation involves only one state variable and that all its probability is concentrated in an interval $a \leq x \leq b$. The shape of the probability distribution inside that interval is completely unknown and subsequent measurements are not independent, i.e. statistical dependence exists between k and $k+1$. However even under that undesirable condition it is possible to extract information from such observations. The effect of these observations will improve the full estimates state vector and will reduce the covariance matrix. In fact, a new Gaussian probability distribution is obtained. The rest of this section explains how to obtain a conservative and consistent update.

The summary of the proposed update process is the following. At time k the estimator produces a Gaussian estimate of the states $x = \{x, y, \varphi, \mathbf{m}\}$ in the form of a joint probability distribution $p_{\mathbf{x},k+1}(\mathbf{x})$, where $\{x, y, \varphi\}$ is the pose of the vehicle and $\{\mathbf{m}\}$ are the states of the landmarks' positions. A bearing observation of φ is performed and with it a marginal probability $p_{\varphi k+1}(\varphi)$ is obtained. With the update of the marginal probability of the observed state, a total update of the joint probability $p_{\mathbf{x},k+1}(\mathbf{x})$ is obtained.

With the non-Gaussian, non-white and partially known probability observation, a new couple $(\hat{\varphi}, \sigma_\varphi)$ is estimated. This pair completely defines the marginal Gaussian density

$$p_{\varphi,k+1}(\varphi) = \frac{1}{\sqrt{2\pi}\sigma_\varphi} e^{\frac{(\varphi-\hat{\varphi})^2}{\sigma_\varphi^2}}, \qquad \hat{\varphi} = E\{\varphi\}, \quad \sigma_\varphi = E\{(\varphi - \hat{\varphi})^2\} \tag{8}$$

The non zero cross-correlation terms in the covariance matrix means that all the states are connected. Then, with this new couple $(\hat{\varphi}, \sigma_{\varphi})$ it is necessary to carry out a virtual update with the purpose of transmitting the new information acquired to the whole density $p_{\mathbf{x},k+1}(\mathbf{x})$ whose expression is

$$p_{\mathbf{x},k+1}(\mathbf{x}) = \frac{1}{\sqrt{2\pi \det(\mathbf{P})}} e^{-(\varphi-\hat{\mathbf{x}})^T \mathbf{P}(\varphi-\hat{\mathbf{x}})}, \qquad \hat{\mathbf{x}} = E\{\mathbf{x}\}, \qquad \mathbf{P} = E\{(\mathbf{x}-\hat{\mathbf{x}})(\mathbf{x}-\hat{\mathbf{x}})^T\} \tag{9}$$

As a result of this update a new joint Gaussian density is obtained, and the normal estimation process is pursued.

In general (Guivant & Masson, 2005), for an arbitrary density $p(\varphi)$ that concentrates all its energy inside the interval (a, b), a Gaussian density with expected value b is a better approximation to $p(\varphi)$ than any other Gaussian density with expected value greater than b if the better previous estimation obtained is greater than b. In particular, this is better than discarding the observation. The same happens with Gaussian densities whose expected value is smaller than a and it is independent of the form that take $p(\varphi)$ inside the interval (a, b). Consequently, the mean ξ of the new density it is selected as

$$if \quad b < c \qquad \Rightarrow \quad \xi = b \tag{10}$$

$$if \quad a > c \qquad \Rightarrow \quad \xi = a \tag{11}$$

$$if \quad a > c > b \qquad \Rightarrow \quad \xi = c \tag{12}$$

where c is the mean of the better previous estimate. The deviation of this new Gaussian must be obtained by solving the following implicit equation

$$if\ c \notin (a,b), \qquad \ln\left(\frac{\sigma_\xi}{\sigma_c}\right) + \frac{-(a-c)^2}{\sigma_c^2} + \frac{(a-b)^2}{\sigma_\xi^2} = 0 \tag{13}$$

$$if\ c \in (a,b), |a-c| > |b-c| \qquad \ln\left(\frac{\sigma_\xi}{\sigma_c}\right) + (a-c)^2\left(-\frac{1}{\sigma_c^2} + \frac{1}{\sigma_\xi^2}\right) = 0 \tag{14}$$

$$if\ c \in (a,b), |a-c| \le |b-c| \qquad \ln\left(\frac{\sigma_\xi}{\sigma_c}\right) + (b-c)^2\left(-\frac{1}{\sigma_c^2} + \frac{1}{\sigma_\xi^2}\right) = 0 \tag{15}$$

Then, unless the mean is updated, the deviation is always improved. This is an important result because it is always necessary to maintain the absolute error between the true value and the mean of the Gaussian bounded. This condition guarantees a secure condition for the EKF as estimator. If the mean value estimated is near the true value the filter will perform almost as a linear estimator. In particular, the Jacobians will be calculated properly. In several cases, the filter could behave in a consistent way. But, given great deviations, the Jacobians evaluated at the mean value will be different from the one calculated at the true value. This fact is widely known in the EKF estimation theory.

At this point the calculation was focused on the marginal density $p(\varphi)$. However the full probability density is a Gaussian multi-dimensional density. The covariance matrix is a full matrix and this shows the correlation between the states of the vehicle and the map. It was shown (Gibbens et al., 2000) that neglecting this correlations leads to non-consistent estimations.

A virtual update is a form to update the full covariance matrix. The desired update over the individual deviation σ_φ is known. With it, it is possible to obtain the complete update without violating conditions of consistency of the estimation. The updated covariance will be

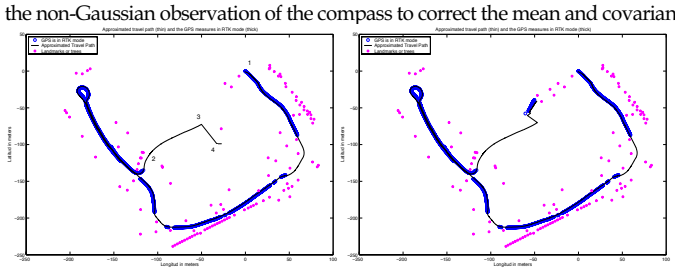
$$\mathbf{P}_{k+1|\,k+1} = \mathbf{P}_{k+1|\,k} - \Delta\mathbf{P} \tag{16}$$

There,

$$\Delta \mathbf{P} = \mathbf{P}_{k+1|k}(:, i_\varphi) \frac{\Delta \sigma_\varphi^2}{\sigma_{\varphi,k}^4} \mathbf{P}_{k+1|k}(i_\varphi, :) \tag{17}$$

where $\mathbf{P}_{k+1|\,k}(:, i_\varphi)$ is the row vector i_φ of the predicted covariance matrix, $\mathbf{P}_{k+1|\,k}(i_\varphi, :)$ is the column vector i_φ, $\Delta \sigma_\varphi^2$ is the improvement in the deviation incorporating the non Gaussian observation and $\sigma_{\varphi,k}$ is the deviation predicted in the state φ.

Fig. 12 shows the proposed approach when it is applied in a SLAM process where non-Gaussian observations come from compass measurements. Details about the vehicle model and the SLAM algorithm could be referred from (Guivant et al., 2002). In this experiment GPS, laser, compass and dead reckoning information was available. The GPS used is capable of providing position information with 2 cm of accuracy when it works in RTK mode. This quality is only available in relatively open areas and is shown in Fig. 12 by using a thick line. The vehicle started at the point labelled 1. An EKF performs SLAM by using all the available information (thin line). When the vehicle arrives at point 2, there is no GPS information and the laser and compass are intentionally disconnected until the vehicle reaches point 3. The reason for this is to allow the uncertainty to grow and clearly show the impact of the algorithm. In Fig. 12 (a), at point 4, it could be seen how the estimator goes far away from the real path that can be seen in Fig. 12 (b). In this last case, the filter uses the non-Gaussian observation of the compass to correct the mean and covariance.



(a) (b)

Fig. 12. Figure (a) shows results from a standard SLAM algorithm which does not use the available compass measurements. At point 4 the data association fails. Figure (b) shows the result from a SLAM process which does use the available compass measurements and at point 4 the data association is successful.

4. Conclusion

A solution to the SLAM problem is necessary to make a robot truly autonomous. For this reason, SLAM has been one of the main research topics in robotics, especially during the last fifteen years. While the structure of the problem is today well known, there are still many open problems, particularly when working in outdoor environments. We presented here some of the latest SLAM algorithms that address the problem of localisation and mapping in large outdoor areas.

5. References

Bailey T. (2002). Mobile Robot Localisation and Mapping in Extensive Outdoor Environments. PhD thesis, University of Sydney, Australian Centre for Field Robotics, 2002.

Durrant-Whyte, H. & Bailey T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, Vol. 13, No. 2, June 2006, 99 – 108, ISSN 1070-9932

Biswas R., Limketkai B., Sanner S. & Thrun S. (2003). Towards object mapping in non-stationary environments with mobile robots. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003*, pp. 1014-1019, ISBN 0-7803-7736-2, Taipei, Taiwan, September 2003, IEEE.

Chieh-Chih Wang, Thorpe C. & Thrun S. (2003). Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003*, pp. 842-849, ISBN 0-7803-7736-2, Taipei, Taiwan, September 2003, IEEE.

Chieh-Chih Wang (2004). Simultaneous Localization, Mapping and Moving Object Tracking. *PhD thesis*, Robotics Institute, Carnegie Mellon University, 2004

Crisan D. & Doucet A. (2000). Convergence of sequential monte carlo methods. *Technical Report Cambridge University*, CUED/FINFENG /TR381, 2000.

Elfes A.(1989). Occupancy Grids: A probabilistic framework for robot perception and navigation. *Phd thesis, Department of Electrical Engineering*, Carnegie Mellon University, 1989

Gibbens P. W., Dissanayake G. M. W. M. & Durrant-Whyte H. F. (2000). A closed form solution to the single degree of freedom simultaneous localisation and map building (SLAM) problem. *Proceedings of the 39th IEEE Conference on Decision and Control*, pp. 191-196, ISBN 0-7803-6638-7, Sydney, Australia, December 2000, IEEE.

Gordon N. J., Salmond D. J. & Smith A. F. M. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings-F Radar and Signal Processing*, Vol. 140, No. 2, April 1993, 107–113, ISSN 0956-375X

Guivant J. & Nebot E. (2002). Improving computational and memory requeriments of simultaneous localization and map building algorithms. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002*, pp. 2731-2736, ISBN 0-7803-7272-7, Washington DC, May 2002, IEEE.

Guivant J., Masson F. & Nebot E. (2002). Simultaneous localization and map building using natural features and absolute information. *Robotics and Autonomous Systems*, Vol. 40, No. 2-3, August 2002, 79–90, ISSN 0921-8890.

Guivant J. & Nebot E. (2003). Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms. *IEEE Transaction on Robotics and Automation*, Vol. 19, No. 4, August 2003, 749 - 755, ISSN 1042-296X.

Guivant J., Nieto J., Masson F. & Nebot E. (2004). Navigation and mapping in large unstructured environments. *The International Journal of Robotics Research*, Vol. 23, No. 4, April 2004, 449- 472, ISSN 0278-3649.

Guivant, J. & Masson, F. (2005). Using Absolute Non-Gaussian Non-White Observations in Gaussian SLAM. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005*, pp. 336 – 341, ISBN 0-7803-8914-X/05, Barcelona, Spain, April 2005, IEEE.

Hahnel D., Triebel R., Burgard W. & Thrun S. (2003). Map building with mobile robots in dynamic environments. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003*, pp. 1557-1563, ISBN 0-7803-7736-2, Taipei, Taiwan, September 2003, IEEE.

Lacroix S., Mallet A., Bonnafous D., Bauzil G., Fleury S., Herrb M., & Chatila R. (2002). Autonomous rover navigation in a unknown terrains: Functions and integrations.

The International Journal of Robotics Research, Vol. 21, No. 10- 11, October - November 2002, 917-942, ISSN 0278-3649.

Lenser S. & Veloso M. (2000). Sensor resetting localization for poorly modelled mobile robots. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000*, pp. 1225–1232, ISBN 0-7803-5886-4, San Francisco, CA, April 2000, IEEE.

Leonard J. J., Rikoski R. J., Newman P. M. & Bosse M. (2002). Mapping partially observable features from multiple uncertain vantage points. *The International Journal of Robotics Research*, Vol. 21, No. 10- 11, October - November 2002, 943-975, ISSN 0278-3649.

Masson F., Guivant J. & Nebot E. (2003). Robust Navigation and Mapping Architecture for Large Environments. *Journal of Robotics Systems*, Vol. 20, No. 10, October 2003, pp. 621 – 634, ISSN 0741-2223

Masson F., Guivant J., Nieto J. & Nebot E. (2005). The hybrid metric map: a solution for precision farming. *Latin American Applied Research*, Vol. 35, No 2, April 2005, pp. 105-110, ISSN 0327-0793.

McKerrow P. J. (1993). Echolocation - from range to outline segments. *Robotics and Autonomous Systems*. Vol. 11, No. 3-4, December 1993, 205-211, ISSN 0921-8890.

Montemerlo M., Thrun S. & Whittaker W. (2002). Conditional particle filters for simultaneous mobile robot localization and people-tracking. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002*, pp. 695-701, ISBN 0-7803-7272-7, Washington DC, May 2002, IEEE

Nebot E. (2002). Experimental outdoor dataset. ACFR, University of Sydney. http://www.acfr.usyd.edu.au/homepages/academic/enebot/dataset.htm

Neira J. & Tardós J.D. (2001). Data association in stochastic mapping using the joint compatibility test. *IEEE Transaction on Robotics and Automation*, Vol. 17, No. 6, December 2001, 890 - 897, ISSN 1042-296X.

Nieto J., Guivant J. & Nebot E.(2004). The hybrid metric maps (HYMMs): A novel map representation for DenseSLAM. *Proceedings of the 2004 International Conference on Robotics and Automation, ICRA 2004*, pp. 391-396, ISBN 0-7803-8232-3/04, New Orleans, LA, April 2004, IEEE.

Nieto J., Bailey T. & Nebot E. (2005). Scan-slam: Combining EKF-SLAM and scan correlation. *Proceedings of the International Conference on Field and Service Robotics*, pp. 129-140, Port Douglas, Australia, July 2005.

Solá J., Monin A., Devy M. & Lemaire T. (2005). Undelayed initialization in bearing only SLAM. *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2005*, pp. 2751–2756, ISBN 0-7803-7736-2, Alberta, Canada, August 2005, IEEE.

Thrun S., Burgard W., Chakrabarti D., Emery R., Liu Y. & Martin C. (2001). A real-time algorithm for acquiring multi-planar volumetric models with mobile robots. *Proceedings of the 10th International Symposium of Robotics Research, ISRR'01*, pp. 21-36, ISBN 3540005501, Lorne, Australia, November 2001.

Thrun S. (2002). Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*, Lakemeyer G. & Nebel B., ,Morgan Kaufmann.

15

Unified Dynamics-based Motion Planning Algorithm for Autonomous Underwater Vehicle-Manipulator Systems (UVMS)

Tarun K. Podder*, Nilanjan Sarkar**
*University of Rochester, Rochester, NY, USA
**Vanderbilt University, Nashville, TN, USA

1. Introduction

Among the underwater robotic systems that are currently available, remotely operated vehicles (ROVs) are the most commonly used underwater robotic systems. A ROV is an underwater vehicle that is controlled from a mother-ship by human operators. Sometimes a ROV is equipped with one or more robotic manipulators to perform underwater tasks. These robotic manipulators are also controlled by human operators from a remote site (e.g., mother-ship) and are known as tele-manipulators. Although the impact of ROVs with tele-manipulators is significant, they suffer from high operating cost because of the need for a mother-ship and experienced crews, operator fatigue and high energy consumption because of the drag generated by the tether by which the ROV is connected to the ship. The performance of such a system is limited by the skills, coordination and endurance of the operators. Not only that, communication delays between the master and the slave site (i.e., the mother-ship and the ROV) can severely degrade the performance.

In order to overcome some of the above-mentioned problems, autonomous underwater vehicles (AUVs) are developed. However, an AUV alone cannot interact with the environment. It requires autonomous robotic manipulator(s) attached to it so that the combined system can perform some useful underwater tasks that require physical contact with the environment. Such a system, where one or more arms are mounted on an AUV, is called an autonomous underwater vehicle-manipulator system (UVMS).

One of the main research problems in underwater robotics is how to design an autonomous controller for a UVMS. Since there is no human operator involved in the control of a UVMS, the task planning has become an important aspect for smooth operation of such a system. Task planning implies the design of strategies for task execution. In other words, a task planning algorithm provides a set of desired (i.e., reference) trajectories for the position and force variables, which are used by the controller to execute a given task. Task planning can be divided into motion planning and force planning. In this research, we focus on the design of motion planning algorithms for a UVMS.

The motion planning of a UVMS is a difficult problem because of several reasons. First, a UVMS is a kinematically redundant system. A kinematically redundant system is one which has more than 6 degrees-of-freedom (DOF) in a 3-D space. Commonly, in a UVMS, the AUV has 6 DOF. Therefore, the introduction of a manipulator, which can have n DOF, makes the

combined system kinematically redundant. Such a system admits infinite number of joint space solutions for a given Cartesian space coordinates, and thus makes the problem of motion planning a difficult one. Second, a UVMS is composed of two dynamic subsystems, one for the vehicle and one for the manipulator, whose bandwidths are vastly different. The dynamic response of the vehicle is much slower than that of the manipulator. Any successful motion planning algorithm must consider this different dynamic bandwidth property of the UVMS. There are several other factors such as the uncertainty in the underwater environment, lack of accurate hydrodynamic models, and the dynamic interactions between the vehicle and the manipulator to name a few, which makes the motion planning for a UVMS a challenging problem.

In robotics, trajectory planning is one of the most challenging problems (Klein & Huang, 1983). Traditionally, trajectory planning problem is formulated as a kinematic problem and therefore the dynamics of the robotic system is neglected (Paul, 1979). Although the kinematic approach to the trajectory planning has yielded some very successful results, they are essentially incomplete as the planner does not consider the system's dynamics while generating the reference trajectory. As a result, the reference trajectory may be *kinematically admissible* but may not be *dynamically feasible*.

Researchers, in the past several years, have developed various trajectory planning methods for robotic systems considering different kinematic and dynamic criteria such as obstacle avoidance, singularity avoidance, time minimization, torque optimization, energy optimization, and other objective functions. A robotic system that has more than 6 dof (degrees-of-freedom) is termed as kinematically redundant system. For a kinematically redundant system, the mapping between task-space trajectory and the joint-space trajectory is not unique. It admits infinite number of joint-space solutions for a given task-space trajectory. However, there are various mathematical tools such as Moore-Penrose Generalized Inverse, which map the desired Cartesian trajectory into the corresponding joint-space trajectory for a kinematically redundant system. Researchers have developed various trajectory planning methods for redundant systems (Klein & Huang, 1983; Zhou & Nguyen, 1997; Siciliano, 1993; Antonelli & Chiaverini, 1998; shi & McKay, 1986). Kinematic approach of motion planning has been reported in the past. Among them, Zhou and Nguyen (Zhou & Nguyen, 1997) formulated optimal joint-space trajectories for kinematically redundant manipulators by applying Pontryagin's Maximum Principle. Siciliano (Siciliano, 1993) has proposed an inverse kinematic approach for motion planning of redundant spacecraft-manipulator system. Antonelli and Chiaverini (Antonelli & Chiaverini, 1998) have used pseudoinverse method for task-priority redundancy resolution for an autonomous Underwater Vehicle-Manipulator System (UVMS) using a kinematic approach.

Several researchers, on the other hand, have considered dynamics of the system for trajectory planning. Among them, Vukobratovic and Kircanski (Vukobratovic & Kircanski, 1984) proposed an inverse problem solution to generate nominal joint-space trajectory considering the dynamics of the system. Bobrow (Bobrow, 1989) presented the Cartesian path of the manipulator with a B-spline polynomial and then optimized the total path traversal time satisfying the dynamic equations of motion. Shiller and Dubowsky (Shiller & Dubowsky, 1989) presented a time-optimal motion planning method considering the dynamics of the system. Shin and McKay (Shin & McKay, 1986) proposed a dynamic programming approach to minimize the cost of moving a robotic manipulator. Hirakawa and Kawamura (Hirakawa & Kawamura, 1997) have proposed a method to

solve trajectory generation problem for redundant robot manipulators using the variational approach with B-spline function to minimize the consumed electrical energy. Saramago and Steffen (Saramago & Steffen, 1998) have formulated off-line joint-space trajectories to optimize traveling time and minimize mechanical energy of the actuators using spline functions. Zhu *et al.* (Zhu *et al.* , 1999) have formulated real-time collision free trajectory by minimizing an energy function. Faiz and Agrawal (Faiz & Agrawal, 2000) have proposed a trajectory planning scheme that explicitly satisfy the dynamic equations and the inequality constraints prescribed in terms of joint variables. Recently, Macfarlane and Croft (Macfarlane & Croft, 2003) have developed and implemented a jerk-bounded trajectory for an industrial robot using concatenated quintic polynomials. Motion planning of land-based mobile robotic systems has been reported by several researchers. Among them, Brock and Khatib (Brock & Khatib, 1999) have proposed a global dynamic window approach that combines planning and real-time obstacle avoidance algorithms to generate motion for mobile robots. Huang *et al.* (Huang *et al.*, 2000) have presented a coordinated motion planning approach for a mobile manipulator considering system stability and manipulation. Yamamoto and Fukuda (Yamamoto & Fukuda, 2002) formulated trajectories considering kinematic and dynamic manipulability measures for two mobile robots carrying a common object while avoiding a collision by changing their configuration dynamically. Recently, Yamashita *et al.* (Yamashita *et al.*, 2003) have proposed a motion planning method for multiple mobile robots for cooperative transportation of a large object in a 3D environment. To reduce the computational burden, they have divided the motion planner into a global path planner and a local manipulation planner then they have designed it and integrated it. All the previously mentioned researches have performed trajectory planning for either space robotic or land-based robotic systems. On the other hand, very few works on motion/trajectory planning of underwater robotic systems have been reported so far. Among them, Yoerger and Slotine (Yoerger & Slotin, 1985) formulated a robust trajectory control approach for underwater robotic vehicles. Spangelo and Egeland (Spangelo & Egeland, 1994) developed an energy-optimum trajectory for underwater vehicles by optimizing a performance index consisting of a weighted combination of energy and time consumption by the system. Recently, Kawano and Ura (Kawano & Ura, 2002) have proposed a motion planning algorithm for nonholonomic autonomous underwater vehicle in disturbance using reinforcement learning (Q-learning) and teaching method. Sarkar and Podder (Sarkar & Podder, 2001) have presented a coordinated motion planning algorithm for a UVMS to minimize the hydrodynamic drag. Note that UVMS always implies an autonomous UVMS here.

However, majority of the trajectory planning methods available in the literature that considered the dynamics of the system are formulated for land-based robots. They have either optimized some objective functions related to trajectory planning satisfying dynamic equations or optimized energy functions. Moreover, for the land-based robotic system, the dynamics of the system is either homogeneous or very close to homogeneous. On the other hand, most of the trajectory planning methods that have been developed for space and underwater robotic systems use the pseudoinverse approach that neglects the dynamics of the system (Siciliano, 1993; Antonelli & Chiaverini, 1998; Sarkar & Podder, 2001).

In this research, we propose a new trajectory planning methodology that generates a kinematically admissible and dynamically feasible trajectory for kinematically

redundant systems whose subsystems have greatly different dynamic responses. We consider the trajectory planning of underwater robotic systems as an application to the proposed theoretical development. In general, a UVMS is composed of a *6* dof Autonomous Underwater Vehicles (AUV) and one (or more) *n* dof robotic manipulator(s). Commonly, the dynamic response of the AUV is an order of magnitude slower than that of the manipulator(s). Therefore, a UVMS is a kinematically redundant heterogeneous dynamic system for which the trajectory planning methods available in the literature are not directly applicable. For example, when the joint-space description of a robotic system is determined using pseudoinverse, all joints are implicitly assumed to have same or similar dynamic characteristics. Therefore, the traditional trajectory planning approaches may generate such reference trajectories that either the UVMS may not be able to track them or while tracking, it may consume exorbitant amount of energy which is extremely precious for autonomous operation in oceanic environment.

Here, we present a new unified motion planning algorithm for a UVMS, which incorporates four other independent algorithms. This algorithm considers the variability in dynamic bandwidth of the complex UVMS system and generates not only kinematically admissible but also dynamically feasible reference trajectories. Additionally, this motion planning algorithm exploits the inherent kinematic redundancy of the whole system and provides reference trajectories that accommodates other important criteria such as thruster/actuator faults and saturations, and also minimizes hydrodynamic drag. All these performance criteria are very important for autonomous underwater operation. They provide a fault-tolerant and reduced energy consuming autonomous operation framework. We have derived dynamic equations of motion for UVMS using a new approach Quasi-Lagrange formulation and also considered thruster dynamics. Effectiveness of the proposed unified motion planning algorithm has been verified by extensive computer simulation and some experiments.

2. UVMS Dynamics

The dynamics of a UVMS is highly coupled, nonlinear and time-varying. There are several methods such as the Newton-Euler method, the Lagrange method and Kane's method to derive dynamic equations of motion. The Newton-Euler approach is a recursive formulation and is less useful for controller design (Kane & Lavinson, 1985; Fu *et al*., 1988; Craig, 1989). Kane's method is a powerful approach and it generates the equations of motion in analytical forms, which are useful for control. However, we choose to develop the dynamic model using the Lagrange approach because of two reasons. First, it is a widely known approach in other fields of robotics and thus will be accessible to a larger number of researchers. Second, this is an energy-based approach that can be easily extended to include new subsystems (e.g., inclusion of another manipulator).

There is a problem, however, to use the standard form of the Lagrange equation to derive the equations of motion of a UVMS. When the base of the manipulator is not fixed in an inertial frame, which is the case for a UVMS, it is convenient to express the Lagrangian not in terms of the velocities expressed in the inertial frame but in terms of velocities expressed in a body attached frame. Moreover, for feedback control, it is more convenient to work with velocity components about body-fixed axes, as sensors

measure motions and actuators apply torques in terms of components about the body-fixed reference frame. However, the components of the body-fixed angular velocity vector cannot be integrated to obtain actual angular displacement. As a consequence of this, we cannot use the Lagrange equation directly to derive the dynamic equations of motion in the body-fixed coordinate frame. This problem is circumvented by applying the *Quasi-Lagrange* approach. The Quasi-Lagrange approach was used earlier to derive the equations of motion of a space structure (Vukobratovic & Kircanski, 1984). Fossen mentioned the use of the same approach to model an AUV (Fossen, 1984).

However, this is the first time that a UVMS is modeled using the Quasi-Lagrange approach. This formulation is attractive because it is similar to the widely used standard Lagrange formulation, but it generates the equations of motion in the body-attached, non-inertial reference frame, which is needed in this case.

We, for convenience, commonly use two reference frames to describe underwater robotic systems. These two frames are namely the earth-fixed frame (denoted by XYZ) and the body-fixed frame (denoted by $X_v Y_v Z_v$), as shown in Fig. 1.

The dynamic equations of motion of a UVMS can be expressed as follows:

$$M_b(q_m)\dot{w} + C_b(q_m, w)w + D_b(q_m, w)w + G_b(q) = \tau_b \tag{1}$$

where the subscript '*b*' denotes the corresponding parameters in the body-fixed frames of the vehicle and the manipulator. $M_b(q_m) \in \Re^{(6+n)\times(6+n)}$ is the inertia matrix including the added mass and $C_b(q_m, w) \in \Re^{(6+n)\times(6+n)}$ is the centrifugal and Coriolis matrix including terms due to added mass. $D_b(q_m, w) \in \Re^{(6+n)\times(6+n)}$ is the drag matrix, $G(q) \in \Re^{(6+n)}$ is the vector of restoring forces and $\tau_b \in \Re^{(6+n)}$ is the vector of forces and moments acting on the UVMS. The displacement vector $q = [q_v, q_m]^T$, where $q_v = [q_1, \ldots, q_6]^T$, and $q_m = [q_7, \ldots, q_{6+n}]^T$. q_1, q_2 and q_3 are the linear (surge, sway, and heave) displacements of the vehicle along X, Y, and Z axes, respectively, expressed in the earth-fixed frame. q_4, q_5 and q_6 are the angular (roll, pitch, and yaw) displacements of the vehicle about X, Y and Z axes, respectively, expressed in the earth-fixed frame. $q_7, q_8, \ldots\ldots, q_{6+n}$ are the angular displacements of joint 1, joint 2, ……., joint *n* of the manipulator in link-fixed frames. The quasi velocity vector $w = [w_1, \ldots\ldots, w_{6+n}]^T$, where w_1, w_2 and w_3 are the linear velocities of the vehicle along X_v, Y_v, and Z_v axes respectively, expressed in the body-fixed frame. w_4, w_5 and w_6 are the angular velocities of the vehicle about X_v, Y_v, and Z_v axes, respectively, expressed in the body-fixed frame. $w_7, w_8, \ldots\ldots, w_{6+n}$ are the angular velocities of manipulator joint 1, joint 2, …. ., joint n, expressed in the link-fixed frame. A detailed derivation of Equation (1) is given in (Podder, 2000).

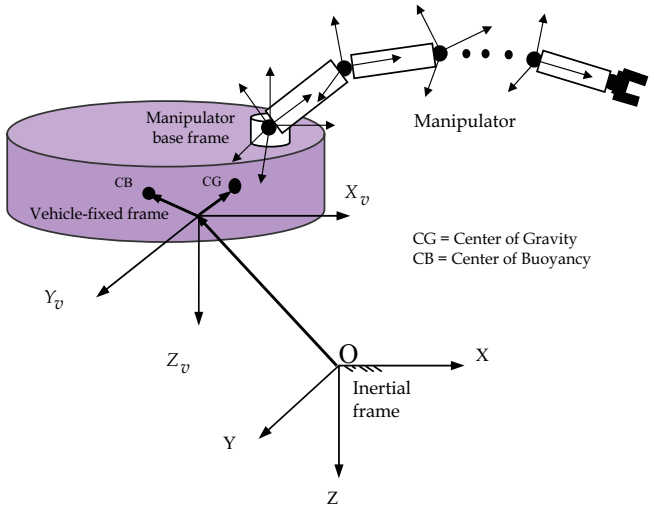


Fig. 1. Coordinate frames for underwater vehicle-manipulator system.

Equation (1) is represented in the body-fixed frame of the UVMS because it is convenient to measure and control the motion of the UVMS with respect to the moving frame. However, the integration of the angular velocity vector does not lead to the generalized coordinates denoting the orientation of the UVMS. In general, we can relate the derivative of the generalized coordinates and the velocity vector in the body-fixed frame by the following linear transformation:

$$\dot{q} = Bw \tag{2}$$

The transformation matrix B in Equation (2) is given by:

$$B(q) = \begin{bmatrix} B_{1_{6\times 6}} & O_{6\times n} \\ O_{n\times 6} & B_{2_{n\times n}} \end{bmatrix}_{(6+n)\times(6+n)}, \; B_1 = \begin{bmatrix} J_1 & O \\ O & J_2 \end{bmatrix}, \; B_2 = [I] \tag{3}$$

where the linear velocity transformation matrix, J_1 , and the angular velocity transformation matrix, J_2 , are given as:

$$J_1 = \begin{bmatrix} C_5C_6 & -S_6C_4 + S_4S_5C_6 & S_4S_6 + S_5C_4C_6 \\ S_6C_5 & C_4C_6 + S_4S_5S_6 & -S_4C_6 + S_5S_6C_4 \\ -S_5 & S_4C_5 & C_4C_5 \end{bmatrix} \tag{4}$$

$$J_2 = \begin{bmatrix} 1 & S_4T_5 & C_4T_5 \\ 0 & C_4 & -S_4 \\ 0 & S_4/C_5 & C_4/C_5 \end{bmatrix} \tag{5}$$

Here S_i, C_i and T_i represent $sin(q_i)$, $cos(q_i)$ and $tan(q_i)$, respectively, and I is the identity matrix. Note that there is an Euler angle (roll, pitch, yaw) singularity in J_2 when the pitch angle (q_5) is an odd multiple of $\pm 90^0$. Generally, the pitch angle in practical operation is

restricted to $|q_5| < 90^0$. However, if we need to avoid singularity altogether, unit quarternions can be used to represent orientation (Fossen, 1984).

3. Dynamics-Based Trajectory Planning Algorithm

Most of the trajectory planning methods found in literature is formulated for land-based robots where the dynamics of the system is homogeneous or very close to homogeneous. The study of UVMS becomes more complicated because of the heterogeneous dynamics and dynamic coupling between two different bandwidth subsystems. From practical point of view it is very difficult and expensive to move a heavy and large body with higher frequency as compared to a lighter and smaller body. The situation becomes worse in the case of underwater systems because of the presence of heavier liquid (water) which contributes significant amount of drag forces. Therefore, it will be more meaningful if we can divide the task into several segments depending on the natural frequencies of the subsystems. This will enable the heterogeneous dynamic system to execute the trajectory not only kinematically admissibly but also dynamically feasibly.
Here we present a trajectory planning algorithm that accounts for different bandwidth characteristic of a dynamic system. First, we present the algorithm for a general *n*-bandwidth dynamic system. Then we improvise this algorithm for application to a UVMS.

3.1 Theoretical Development

Let us assume that we know the natural frequency of each subsystem of the heterogeneous dynamic system. This will give us a measure of the dynamic response of each subsystem. Let these frequencies be ω_i, $i = 1,2,\cdots\cdots,s$.

We approximate the task-space trajectories using Fourier series and represent it in terms of the summation of several frequencies in ascending order.

$$x_{d\,6\times1}(t) = f_{6\times1}(t) = a_0 + \sum_{r=1}^{\infty} a_r \cos(r\pi t / L) + \sum_{r=1}^{\infty} b_r \sin(r\pi t / L) \tag{6}$$

where a_0, a_r, b_r are the coefficients of Fourier series and are represented as 6×1 column vectors, $r/2L$ is the frequency of the series and $2L$ is the time period.

Now we truncate the series at a certain value of *r* (assuming $r = p_1$ to be sufficiently large) so that it can represent the task-space trajectories reasonably. We rewrite the task-space trajectory in the following form:

$$x_{d\,6\times1}(t) = f_{6\times1}(t) = f_1(t) + f_2(t) + \cdots\cdots + f_{p_1}(t) \tag{7}$$

where $f_1(t) = a_0 + a_1 \cos(\pi t / L) + b_1 \sin(\pi t / L)$, and $f_j(t) = a_j \cos(j\pi t / L) + b_j \sin(j\pi t / L)$ for $j = 2,3,...,p_1$.

We then use these truncated series as the reference task-space trajectories and map them into the desired (reference) joint-space trajectories by using weighted pseudoinverse method as follows:

$$\dot{q}_{d_j} = J^{+}_{W_j} \dot{x}_{d_j} \tag{8}$$

$$\ddot{q}_{d_j} = J^{+}_{W_j} (\ddot{x}_{d_j} - \dot{J}\dot{q}_{d_j}) \tag{9}$$

where $\dot{q}_{d_j}$ are the joint-space velocities and $\ddot{q}_{d_j}$ are the joint-space accelerations corresponding to the task-space velocities $\dot{x}_{d_j} = d(f_j(t))/dt$ and task-space accelerations $\ddot{x}_{d_j} = d^2(f_j(t))/dt^2$ for $j = 1,2,...,p_1 \cdot\ J^+_{w_j} = W_j^{-1} J^T (J W_j^{-1} J^T)^{-1}$ are the weighted pseudoinverse of Jacobians and $W_j = diag(h_{1_j}, ..., h_{(6+n)_j})$ are diagonal weight matrices.

In our proposed scheme we use weighted pseudoinverse technique in such a way that it can act as a filter to remove the propagation of undesirable frequency components from the task-space trajectories to the corresponding joint-space trajectories for a particular subsystem. This we do by putting suitable zeros in the diagonal entries of the W_j^{-1} matrices in Equation (8) and Equation (9). We leave the other elements of W_j^{-1} as unity. We have developed two cases for such a frequency-wise decomposition as follows:

Case I – Partial Decomposition:

In this case, the segments of the task-space trajectories having frequencies ω_t $(\omega_t \le \omega_i)$ will be allocated to all subsystems that have natural frequencies greater than ω_t up to the maximum bandwidth subsystem. To give an example, for a UVMS, the lower frequencies will be shared by both the AUV and the manipulator, whereas the higher frequencies will be solely taken care of by the manipulator.

Case II- Total Decomposition:

In this case, we partition the total system into several frequency domains, starting from the low frequency subsystem to the very high frequency subsystem. We then allocate a particular frequency component of the task-space trajectories to only those subsystems that belong to the frequency domain just higher than the task-space component to generate joint-space trajectories. For a UVMS, this means that the lower frequencies will be taken care of by the vehicle alone and the higher frequencies by the manipulator alone.

To improvise the general algorithm for a (*6+n*) dof UVMS, we decompose the task-space trajectories into two components as follows:

$$f(t) = f_{11}(t) + f_{22}(t) \tag{10}$$

where $f_{11}(t) = a_0 + \sum_{r=1}^{r_1} a_r \cos(r\pi t / L) + \sum_{r=1}^{r_1} b_r \sin(r\pi t / L)$, $f_{22}(t) = \sum_{r=r_1+1}^{r_2} a_r \cos(r\pi t / L) + \sum_{r=r_1+1}^{r_2} b_n \sin(r\pi t / L)$, r_1 and r_2 $(r_2 = p_1)$ are suitable finite positive integers. Here, $f_{11}(t)$ consists of lower frequency terms and $f_{22}(t)$ has the higher frequency terms.

Now, the mapping between the task-space variables and the joint-space variables are performed as

$$\ddot{q}_{d_1} = J^+_{W_1}(\ddot{x}_{d_1} - \dot{J}\dot{x}_{d_1}) \tag{11}$$

$$\ddot{q}_{d_2} = J^+_{W_2}(\ddot{x}_{d_2} - \dot{J}\dot{x}_{d_2}) \tag{12}$$

$$\ddot{q}_d = \ddot{q}_{d_1} + \ddot{q}_{d_2} \tag{13}$$

where $W_i \in \Re^{(6+n)\times(6+n)}$ are the weight matrices, $\ddot{q}_d \in \Re^{(6+n)}$ are the joint-space accelerations and $J_{wi}^{+} = W_i^{-1} J^T (J W_i^{-1} J^T)^{-1}$ for (*i*=1,2). We have considered the weight matrices for two types of decompositions as follows:
For Case I – Partial decomposition:

$$W_1 = diag(h_1, h_2, ..., h_{6+n}) \tag{14}$$

$$W_2 = diag(0, ..., 0, h_7, ..., h_{6+n}) \tag{15}$$

For Case II- Total decomposition:

$$W_1 = diag(h_1, ..., h_6, 0, ..., 0) \tag{16}$$

$$W_2 = diag(0, ..., 0, h_7, ..., h_{6+n}) \tag{17}$$

The weight design is further improved by incorporating the system's damping into the trajectory generation for UVMS. A significant amount of energy is consumed by the damping in the underwater environment. Hydrodynamic drag is one of the main components of such damping. Thus, if we decompose the motion in the joint-space in such a way that it is allocated in an inverse ratio to some measure of damping, the resultant trajectory is expected to consume less energy while tracking the same task-space trajectory. Thus, we incorporate the damping into the trajectory generation by designing the diagonal elements of the weight matrix as $h_i = f(\zeta_i)$, where ζ_i (*i*=1,……,6+*n*) is the damping ratio of the particular dynamic subsystem which can be found out using multi-body vibration analysis techniques (James *et al.*, 1989). A block diagram of the proposed scheme has been shown in Fig. 2.

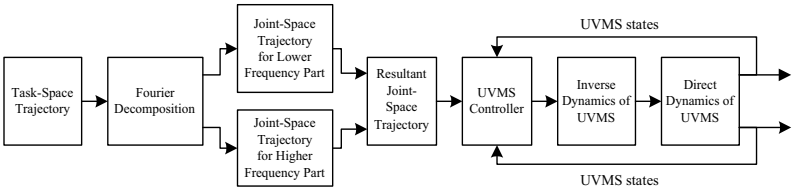


Fig. 2. Dynamics-based planning scheme.

3.2 Implementation Issues

It is to be noted that in the proposed dynamics-based method we have decomposed the task-space trajectory into two domains where the lower frequency segments of the task-space trajectories are directed to either the heavier subsystem, i.e., the vehicle in Case II, or to both the heavier and lighter subsystems, i.e., the vehicle and the manipulator as in Case I. The high frequency segments of the task-space trajectories, on the other hand, are always allocated to the lighter subsystem, i.e., the manipulator. These allocations of task-space trajectories have been mapped to corresponding joint-space trajectories by utilizing weighted pseudoinverse technique where the heterogeneous dynamics of the UVMS have been taken into consideration. Then, these reference joint-space trajectories are followed by the individual joint/dof to execute the end-effector's trajectories.
There are two basic issues of this proposed algorithm that must be discussed before it can be implemented. They are: given a nonlinear, multi degree-of-freedom (*n*-DOF) dynamic

system having different frequency bandwidth subsystems, how to find the 1) natural frequencies of each subsystem, and 2) the damping ratios of each subsystem. We briefly point out the required steps that are needed to obtain these system dynamic parameters: (1) Linearize the dynamic equations, (2) Find the eigenvalues and eigenvectors from the undamped homogeneous equations, (3) Find the orthogonal modal matrix (*P*), (4) Find the generalized mass matrix ($P^T MP$), (5) Find the generalized stiffness matrix ($P^T KP$), (6) Find the weighted modal matrix ($\widetilde{P}$), (7) Using Rayleigh damping equation find a proportional damping matrix, and (8) Decouple the dynamic equations by using $\widetilde{P}$.
After all these operations, we will obtain *(6+n)* decoupled equations similar to that of a single-dof system instead of *(6+n)* coupled equations. From this point on, finding the natural frequencies (ω_i) and the damping ratios (ζ_i) are straightforward. A detailed discussion on these steps can be found in advanced vibration textbook (James *et al.*, 1989).

3.3 Results and Discussion

We have conducted extensive computer simulations to investigate the performance of the proposed Drag Minimization (DM) algorithm. The UVMS used for the simulation consists of a 6 dof vehicle and a 3 dof planar manipulator working in the vertical plane. The vehicle is ellipsoidal in shape with length, width and height 2.0*m*, 1.0*m* and 1.0*m*, respectively. The mass of the vehicle is 1073.0*Kg*. The links are cylindrical and each link is 1.0*m* long. The radii of link 1, 2 and 3 are 0.1*m*, 0.08*m* and 0.07*m*, respectively. The link masses (oil filled) are 32.0*Kg*, 21.0*Kg* and 16.0*Kg*, respectively. We have compared our results with that of the conventional Pseudoinverse (PI) method (i.e., without the null-space term), which is a standard method for resolving kinematic redundancy.

3.3.1 Trajectory

We have chosen a square path in xy (horizontal) plane for the computer simulation. We have assumed that each side of the square path is tracked in equal time. The geometric path and the task-space trajectories are given in Fig. 3.

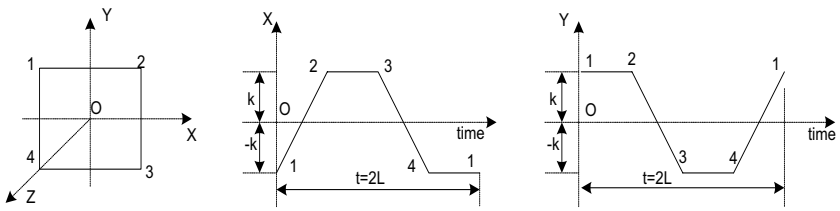


Fig. 3. Task-space geometric path and trajectories.

The task-space trajectories can be represented as

$$x(t) = f_x(t) = \begin{cases} 4kt/L - k & if\ \ 0 < t \le L/2 \\ k & if\ \ L/2 < t \le L \\ 5k - 4kt/L & if\ L < t \le 3L/2 \\ -k & if\ 3L/2 < t \le 2L \end{cases} \tag{18}$$

$$y(t) = f_y(t) = \begin{cases} k & if\ 0 < t \le L/2 \\ 3k - 4kt/L & if\ L/2 < t \le L \\ -k & if\ L < t \le 3L/2 \\ 4kt/L - 7k & if\ 3L/2 < t \le 2L \end{cases} \tag{19}$$

$$z(t) = f_z(t) = 0 \tag{20}$$

The Fourier series for the above trajectories are as follows:

$$f_j(t) = a_{0j} + \sum_{r=1}^{\infty} a_{rj} \cos(r\pi t/L) + \sum_{r=1}^{\infty} b_{rj} \sin(r\pi t/L) \tag{21}$$

where '*j*' implies the coefficients for *x, y* or *z*; *k* is a constant and *2L* is the time period. The Fourier coefficients are:

$a_{0x} = -a_{oy} = k$, $a_{rx} = -a_{ry} = 4k/(r\pi)^2 \ (\cos r\pi - 1)$ and $b_{rx} = -b_{ry} = 8k/(r\pi)^2 \ \sin(r\pi/2)$.

For this simulation, we have taken *k*= *1m*, i.e., the path is 2*m* square, *L*=5 and maximum frequency at which the Fourier series is truncated is $r = p_1 = 30$. The frequency of the manipulator is 10 times higher than that of the vehicle. We have taken the natural frequency of the vehicle as 0.15 cycles per second and the manipulator to be 10 time faster than the vehicle. We have segmented the task-space trajectories as

$$f_{j1}(t) = a_{0j} + a_{1j} \cos(\pi t/L) + b_{1j} \sin(\pi t/L) \tag{22}$$

$$f_{j2}(t) = \sum_{r=2}^{30} a_{rj} \cos(r\pi t/L) + \sum_{r=2}^{30} b_{rj} \sin(r\pi t/L) \tag{23}$$

We have compared our results from the proposed dynamics-based trajectory planning method with that from the conventional straight-line trajectory planning method using regular pseudoinverse technique. In conventional method, the trajectory is designed in three sections: the main section (intermediate section), which is a straight line, is preceded and followed by two short parabolic sections (Fu *et al*., 1988; Craig, 1989). The simulation time is 10.0*sec,* which is required to complete the square path in XY (horizontal) plane. The total length of the path is 8.0*m*; the average speed is about 1.6*knot*. This speed is more than JASON vehicle (speed = 1.0*knot*) but less than SAUVIM system (designed speed = 3.0*knot*).

We have presented results from computer simulations in Fig. 4 through Fig. 9. Results for Case I (Partial Decomposition) are plotted in Fig. 5 through Fig. 7 and that of for Case II (Total Decomposition) are provided in Fig. 8 through Fig. 9. It is observed from Fig. 4 and 5 that the end-effector tracks the task-space paths and trajectories quite accurately. The errors are very small. The joint-space trajectories are plotted in Fig. 6. It is observed that the proposed dynamics-based method restricts the motion of the heavy subsystem and allows greater motion of the lighter subsystem to track the trajectory. It is also noticed that the motion of the heavy subsystem is smoother. The errors in joint-space trajectory are almost zero.

Simulation results for surge-sway motion, power requirement and energy consumption for conventional straight-line method are plotted in the left column and that of for proposed dynamics-based method are plotted in the right column in Fig. 7. Top two plots of Fig. 7 show the differences in surge-sway movements for two methods. In case of the conventional method, the vehicle changes the motion very sharply as compared to the motion generated from the dynamics-based method. It may so happen that this type of sharp movements may be beyond the capability of the heavy dynamic subsystem and consequently large errors in trajectory tracking may occur. Moreover, the vehicle will experience large velocity and

acceleration in conventional method that result in higher power requirement and energy consumption, as we observe in Fig. 7.

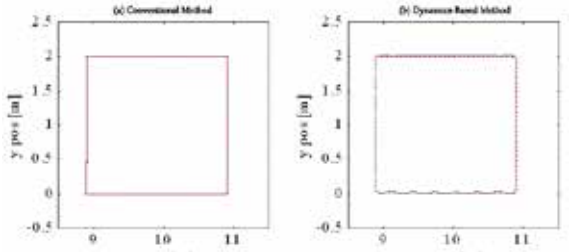


Fig. 4. Task-space geometric paths, (a) Conventional Straight-line planning method and (b) Dynamics-Based planning method for Case I. The actual path is denoted by solid line and the desired path is denoted by dashed line.

We have also presented simulation results for Case II (Total Decomposition) in Fig. 8 and 9. From Fig. 8 it is observed that even though the vehicle has moved more as compared to the conventional straight-line planning method, the motion is smooth. This type of motion is more realistic for a heavy subsystem like the vehicle here and it also avoids large acceleration of the vehicle. On the other hand, the movement of the manipulator is smaller but sharper than that of the conventional method. In the plots in the left column of Fig. 9 it is shown that the end-effector tracks the task-space trajectories quite accurately. The second plot on the right column of this figure shows that the power requirement of the UVMS is less in Case II of the proposed dynamics-based method as compared to that of in conventional method.

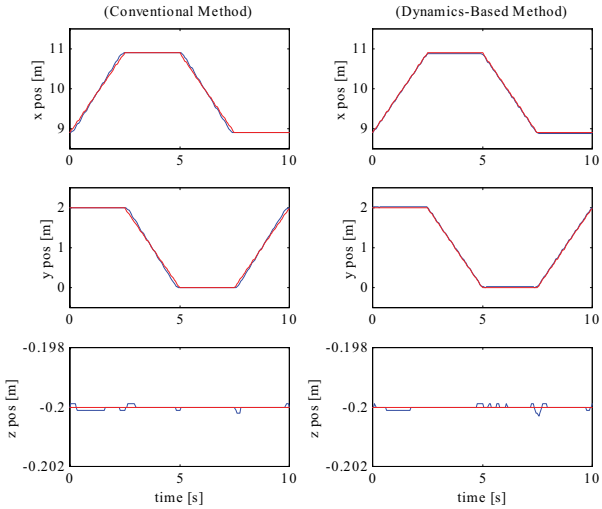


Fig. 5. Task-space trajectories: Conventional Straight-line planning method (left column) and Dynamics-Based planning method for Case I (right column). Desired trajectories are denoted by dashed lines and actual trajectories are denoted by dashed lines.

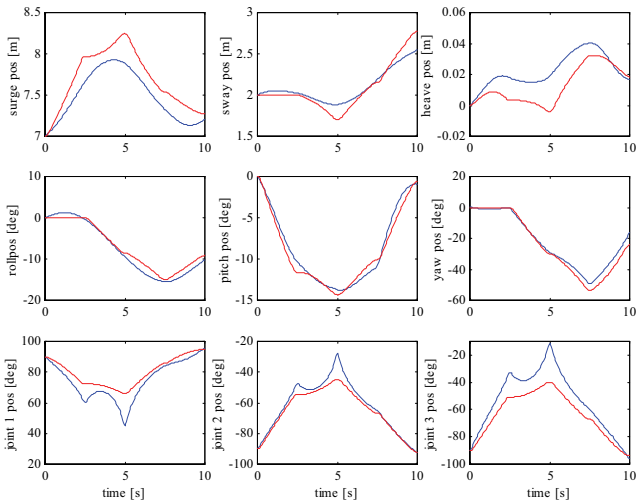


Fig. 6. Joint-space trajectories: Dynamics-Based planning method for Case I (solid/blue line) and Conventional Straight- line planning method (dashed/red line).

For Case II, we can say even though the reduction of energy consumption is not much, however, the movement is smooth that can be practically executed. The power requirement is also less as compared to the conventional straight-line planning method.

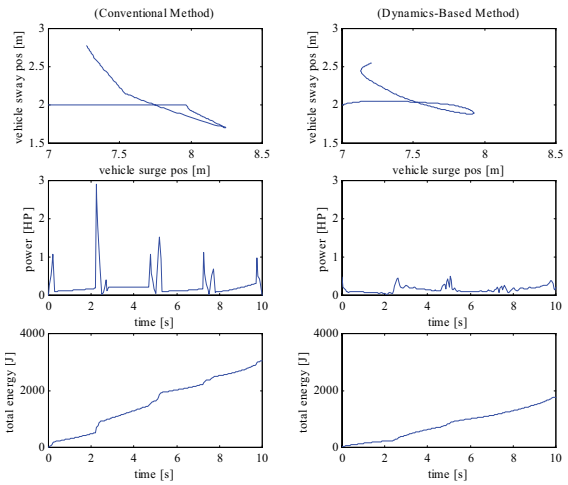


Fig. 7. X-Y motion of the center of gravity, power and energy consumption of the UVMS. Left column for Conventional Straight-line planning method and right column for Dynamics-Based planning method for Case I (partial decomposition).

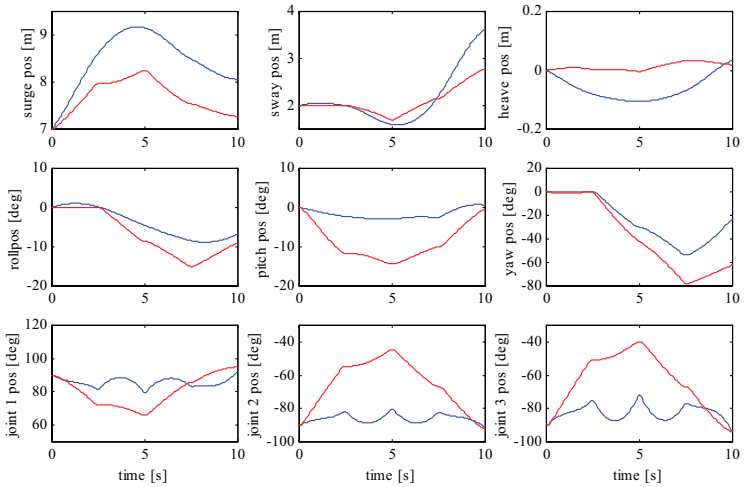


Fig. 8. Joint-space trajectories: Dynamics-Based planning method (solid/blue line) for Case II and Conventional Straight- line planning method (dashed/red line).

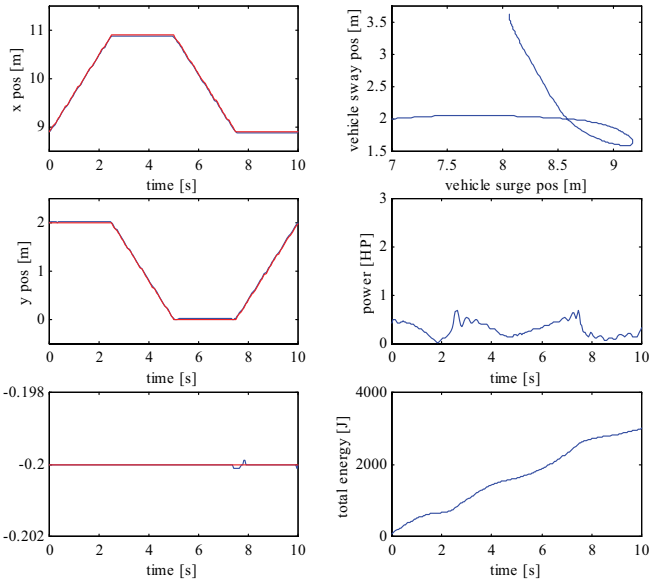


Fig. 9. Task-space trajectories (left column), and surge-sway motion, power requirement and energy consumption (right column) for Dynamics-Based planning method for Case II (total decomposition).

4. Fault Tolerant Decomposition Algorithm

A UVMS is expected to function in a hazardous and unstructured underwater environment. A thruster/actuator fault can occur due to various reasons. There are different methods to detect and isolate these faults. Without going into the details of the possible nature of thruster/actuator faults and how they can be detected and isolated, we assume in this work that we can detect and isolate thruster/actuator faults when they occur. In general, there are more thrusters and actuators than what is minimally required for the specific dof that a UVMS is designed for. Here, we develop an algorithm to exploit the thruster and actuator redundancy to accommodate thruster/actuator faults during operation.

4.1 Theoretical Development

In order to relate the generalized force vector τ_b with the individual thruster/actuator force/torque, let us consider a UVMS which has *p* thrusters and actuators where, in general, $p \geq (6+n)$. In such a case, we can write

$$\tau_b = EF_t \tag{24}$$

where $E \in \Re^{(6+n)\times p}$ *thruster configuration matrix* and $F_t \in \Re^p$ is the reference thruster and actuator forces and torques. The thruster configuration matrix is a constant matrix that depends on the geometric locations of the thrusters and actuators.

Substituting Equation (24) into Equation (1) and performing algebraic manipulation we get

$$\dot{w} = M_b^{-1}(EF_{td} - \xi_b) \tag{25}$$

where $\xi_b = C_b(q_m, w)w + D_b(q_m, w)w + G_b(q)$.

Differentiation of Equation (2) leads to the following acceleration relationship:

$$\ddot{q} = B\dot{w} + \dot{B}w \tag{26}$$

Now, from Equation (25) and Equation (26) we can write

$$\ddot{q} = \eta F_t + \lambda \tag{27}$$

where $\eta_{(6+n)\times p} = BM_b^{-1}E$ and $\lambda_{(6+n)\times 1} = \dot{B}w - BM_b^{-1}\xi_b$.

From Equation (27), using weighted pseudoinverse technique we obtain a least-norm solution to thruster and actuator forces and torques as

$$F_t = \eta_W^+(\ddot{q} - \lambda) \tag{28}$$

where $\eta_W^+ = W^{-1}\eta^T(\eta W^{-1}\eta^T)^{-1}$ is the weighted pseudoinverse of η and $W = diag(h_1 h_2, ..., h_p)$ is the weight matrix.

Now, we construct a thruster fault matrix, $\psi_{p\times p} = W^{-1}$, with diagonal entries either 1 or 0 to capture the fault information of each individual thruster/actuator. If there is any thruster/actuator fault we introduce 0 into the corresponding diagonal element of ψ, otherwise it will be 1. We can also rewrite Equation (28) in terms of thruster fault matrix, ψ, as

$$F_t = \psi\eta^T(\eta\psi\eta^T)^{-1}(\ddot{q} - \lambda) \tag{29}$$

Equation (29) provides us the fault tolerant allocation of thruster/actuator force/torque, F_t. More detailed discussion on this topic can be found in (Podder & Sarkar, 2000; Podder *et al.*, 2001).

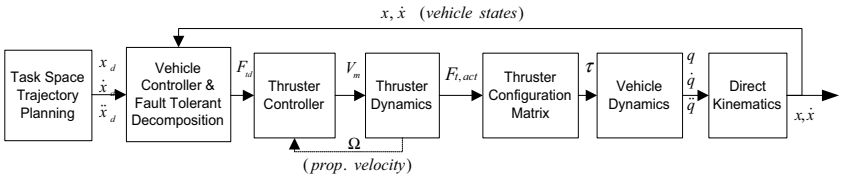


Fig. 10. Fault-tolerant control scheme.

4.2 Experimental Setup

We have conducted both computer simulations and underwater experiments to verify the proposed fault-tolerant control scheme. We have used ODIN (Omni-Directional Intelligent Navigator), which is a 6 dof vehicle designed at the University of Hawaii], as our test-bed. ODIN is a near-spherical AUV that has 4 horizontal thrusters and 4 vertical thrusters as shown in Fig. 11. We have compared our simulation results with that of actual experiments, and presented them later in this section.

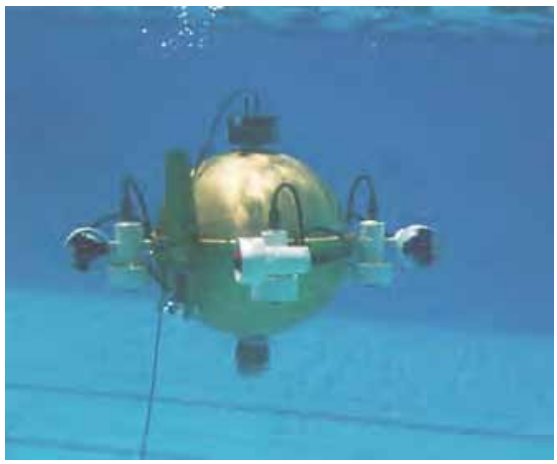


Fig. 11. Omni-Directional Intelligent Navigator (ODIN) vehicle.

The ODIN has a near-spherical shape with horizontal diameter of 0.63m and vertical diameter of 0.61m, made of anodized Aluminum (AL 6061-T6). Its dry weight is 125.0Kg and is slightly positively buoyant. The processor is a Motorola 68040/33MHz working with VxWorks 5.2 operating systems. The RS232 protocol is used for RF communication. The RF Modem has operating range up to 480m, operating frequency range 802-928MHz, and maximum transmission speed 38,400 baud data rate. The power supply is furnished by 24 Lead Gel batteries, where 20 batteries are used for the thrusters and 4 batteries are used for the CPU. ODIN can perform two hours of autonomous operation.

The actuating system is made of 8 thrusters of which 4 are vertical and 4 are horizontal. Each thruster has a brushless DC motor weighing approximately 1Kg and can provide a maximum thrust of approximately 27N. The sensor system is composed of: 1) a pressure

sensor for measuring depth with an accuracy of 3*cm*, 2) 8 sonars for position reconstruction and navigation, each with a range of 0.1-14.4*m*, and 3) an inertial system for attitude and velocity measurement. Since the sonars need to be in the water to work properly, the first 100*sec* of sonar data is not accurate.

The experiments were conducted at the University of Hawaii swimming pool. Several experiments were performed to verify the proposed control scheme. The thruster faults were simulated by imposing zero voltages to the relevant thrusters.

4.3 Results and Discussion

We have performed extensive computer simulations and a number of experiments to verify the proposed planning and control scheme. We present simulation results for two cases to demonstrate the effectiveness of the proposed method. In Case 1, all thrusters are in working condition and therefore the thruster fault matrix Ψ becomes an identity matrix. In Case 2, there are two thrusters that stop working during trajectory tracking operation. In both the cases, ODIN tries to track the following trajectories: it first moves toward the z-direction for 120*sec* to reach a depth of 2*m*. Then it moves toward the y-direction for another 120.0*sec* to traverse 2.0*m*. It subsequently moves towards the x-direction for 120*sec* to traverse 2*m*. Finally it hovers at that position for another 40*sec*. ODIN follows a trapezoidal velocity profile during this task. The attitudes are always kept constant at $[0^0\ 0^0\ 90^0]$. For Case 2, one horizontal thruster (Thruster 6) fails at 260*sec* and one vertical thruster (Thruster 2) fails at 300*sec* while tracking the same trajectories as explained in Case 1. In simulations, we have introduced sensory noise in position and orientation measurements. We have chosen Gaussian noise of 2*mm* mean and 1.5*mm* standard deviation for the surge, sway and heave position measurements, 0.15*degree* mean and 0.15*degree* standard deviation for the roll, pitch and yaw position measurements for the vehicle.

In Fig. 12, we present results from a trajectory following task when there is no thruster fault. It can be observed that both the simulation and the experimental results for all the six trajectories match their respective desired trajectories within reasonable limits. It should also be noted that the particular sonar system of ODIN requires 100.0*sec* before it works properly. Thus, x and y trajectories in experiments have data after 100.0*sec*. However, the depth and attitude sensors provide information from the beginning of the task. In Fig. 13, the same trajectory following task is performed but with thruster faults. In this case, one horizontal thruster (Thruster 6) fails at 260.0*sec* (marked as 'A') and one vertical thruster (Thruster 2) fails at 300.0*sec* (marked as 'B'). Both the faulty thrusters are located at the same thruster bracket of the ODIN. Thus, this situation is one of the worst fault conditions. The simulation results are not affected by the occurrence of faults except in the case of the yaw trajectory, which produces a small error at the last part of the trajectory. In experiment, the first fault does not cause any tracking error. There are some small perturbations after the second fault from which the controller quickly recovers. It can also be noticed that in case of experiment the tracking performance is better in z-direction (depth) as compared to other two directions, i.e., x-direction and y-direction. This happened because of two reasons: 1) less environmental and hydrodynamic disturbances in z-direction, and 2) the pressure sensor for depth measurement is more accurate as compared to sonar sensors used to measure x-position and y-position. However, the orientation of the AUV, which is measured by INS sensors, is reasonably good.

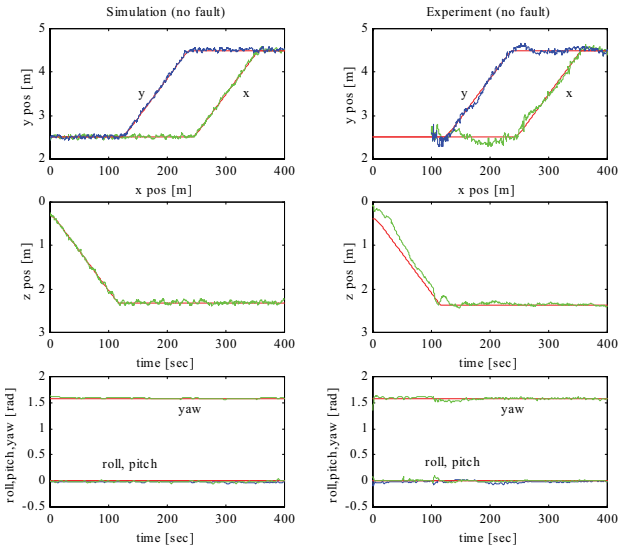


Fig. 12. Trajectory tracking with no thruster fault, simulation results in the left and experimental results in the right. The actual trajectories are denoted by solid lines and the desired trajectories are denoted by dashed lines.

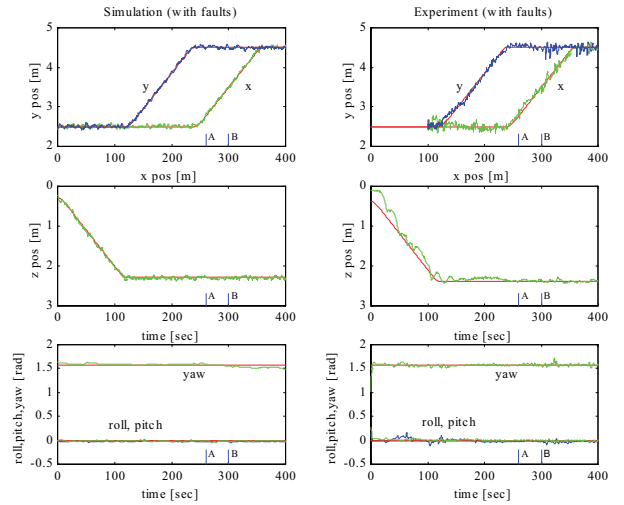


Fig. 13. Trajectory tracking with thruster faults, simulation results in the left and experimental results in the right. The actual trajectories are denoted by solid lines and the desired trajectories are denoted by dashed lines.

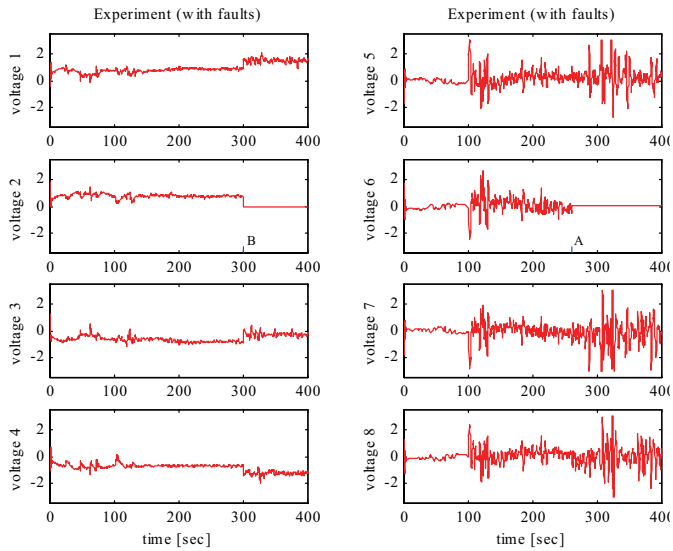


Fig. 14. Voltage versus time plots for the vertical thrusters (left) and the horizontal thrusters (right).

The voltage plots for Case 2 are presented in Fig. 14. It can be seen that voltage for Thruster 6 is zero after 260*sec* and that of Thruster 2 after 300*sec,* which imply thruster faults. From these plots it is observed that in case of simulations all the thruster voltages and in case of experiment the vertical thruster voltages are within $\pm 2volt$. Whereas, the horizontal thruster voltages in case of experiment have some spikes greater than $\pm 2volt$. The causes are as mentioned previously. We also observe that the voltage profile for the vertical thrusters matches well between simulations and experiments. This match was less obvious for horizontal thrusters. However, in all the cases, the range and general pattern seem to be consistent. More details can be found in (Podder *et al.*, 2001).

5. Saturation Limit Algorithm

In the previous section, we have derived Equation (29) for desired thruster/actuator force/torque allocation that allows the operation of the UVMS with faults. However, it cannot guarantee that the desired allocated forces/torques will remain within the saturation limit of the thrusters/actuators. As a result, if some of the forces and torques determined from those equations are beyond the capacity of the corresponding thrusters and actuators, the performance of the controller will suffer because of saturation effect.

5.1 Theoretical Development

The saturation problem cannot be solved based on the formulation given by Equations (29). In order to avoid the saturation effect, the thruster/actuator force/torque must be

controlled so that it cannot reach the saturation limit. However, in such a case, since the input to the controller and the output of the controller will be algebraically related, static state feedback technique will not be able to control the thruster and actuator forces and torques. We, therefore, propose to use the dynamic state feedback technique (Isidori *et al.*, 1968; Yun, 1988) to generate thruster forces that are within the saturation limit. The basic idea of dynamic state feedback is to introduce integrators at the input channel to enlarge the state space, and then apply the static state feedback on the enlarged system.

However, there is a difference between explicit control of the thruster/actuator forces/torques so that it can be regulated about a point or can follow a trajectory, and keep it within a specified range without regulation. The former is a force control problem and the latter is a saturation problem. We first use the dynamic state feedback technique to enlarge the state space in the following way.

We differentiate the output Equation (27) to obtain a new input ν as follows:

$$\ddot{q} = \eta \dot{F}_t + \dot{\eta} F_t + \dot{\lambda} = \eta \nu + \gamma \tag{30}$$

where $\gamma = \dot{\eta} F_t + \dot{\lambda}$ and $\nu = \dot{F}_t$.

Now, we consider the following control law

$$\nu = \eta_{WF}^{+} [\ddot{q}_d + K_1(\ddot{q}_d - \ddot{q}) + K_2(\dot{q}_d - \dot{q}) + K_3(q_d - q)] - \gamma \tag{31}$$

and integration of Equation (31) yields the desired thruster and actuator forces and torques as

$$F_{td} = \int \nu \, dt \tag{32}$$

where $\eta_{WF}^{+} = W_F^{-1} \eta^T (\eta W_F^{-1} \eta^T)^{-1}$, K_1 is the acceleration gain, K_2 is the velocity gain and K_3 is the position gain; q_d and F_{td} are desired parameters of q and F_t, respectively. The diagonal elements of the weight matrix, $W_F = diag(\hbar_1, \hbar_2 ....., \hbar_{p)})$, are computed from the thruster/actuator saturation limits as follows:

We define a function of thruster and actuator force and torque variables as

$$H(F_t) = \sum_{i=1}^{p} \frac{1}{C_i} \frac{(F_{t_{i,\max}} - F_{t_{i,\min}})}{(F_{t_{i,\max}} - F_{t_i})(F_{t_i} - F_{t_{i,\min}})} \tag{33}$$

where $F_{t_{i,\max}}$ and $F_{t_{i,\min}}$ are the upper and lower limits of thrust/torque of the *i*-th thruster/actuator, and C_i is a positive quantity which is determined from the damping property of the dynamic system. Then, differentiating Equation (24), we obtain

$$\frac{\partial H(F_t)}{\partial F_{t_i}} = \frac{(F_{t_{i,\max}} - F_{t_{i,\min}})^2 (2F_{t_i} - F_{t_{i,\max}} - F_{t_{i,\min}})}{C_i \, (F_{t_{i,\max}} - F_{t_i})^2 (F_{t_i} - F_{t_{i,\min}})^2} \tag{34}$$

Then, the diagonal elements of the weight matrix are defined as

$$\hbar_i = 1 + \left| \partial H(F_i) / \partial F_i \right| \tag{35}$$

From the above expression (34), we notice that $\partial H(F_t) / \partial F_{t_i}$ is equal to zero when the *i*-th thruster/actuator is at the middle of its range, and becomes infinity at either limits. Thus, $\hbar_i$ varies from 1 to infinity if the *i*-th thrust goes from middle of the range to its limit. If the *i*-th thrust/torque approaches its limit, then $\hbar_i$ becomes very large and the corresponding element in W_F^{-1} goes to zero and the *i*-th thruster/actuator avoids saturation. Depending upon whether the thruster/actuator is approaching toward or departing from its saturation

limit, $\hbar_i$ can be redefined as $\hbar_i = 1 + \left|\partial H(F_i)/\partial F_i\right|$ when $\Delta\left|\partial H(F_i)/\partial F_i\right| \geq 0$ (i.e., the thruster/actuator force/torque is approaching toward its limit), and $\hbar_i = 1$ when $\Delta\left|\partial H(F_i)/\partial F_i\right| \leq 0$ (i.e., the thruster/actuator force/torque is departing from its limit). Finally, the desired thruster/actuator force/torque vector, F_{td}, that is guaranteed to stay within the saturation limit.

Substituting Equation (31) into Equation (30) and denoting $e = q_d - q$, we obtain the following error equation in joint-space

$$\dddot{e} + K_1\ddot{e} + K_2\dot{e} + K_3 e = 0 \tag{36}$$

Thus, for positive values of the gains, K_1, K_2 and K_3, the joint-space errors reduce to zero asymptotically, as time goes to infinity.

Now, to incorporate both the fault and the saturation information in control Equation (31), we define a *fault-saturation matrix,* $\Gamma_{p\times p} = W_F^{-1}$, having diagonal elements either $1/\hbar_i$ or zero. Whenever there is any thruster/actuator fault, we put that corresponding diagonal element in Γ matrix as zero. If there is no fault in any thruster/actuator, that corresponding diagonal entry of the fault-saturation matrix will be $1/\hbar_i$. Thus, it accounts for the force/torque saturation limits along with the fault information. We can rewrite the Equation (31) in terms of fault-saturation matrix, Γ, as

$$\nu = \Gamma\eta^T(\eta\Gamma\eta^T)^{-1}[\ddot{q}_d + K_1(\ddot{q}_d - \ddot{q}) + K_2(\dot{q}_d - \dot{q}) + K_3(q_d - q)] - \gamma \tag{37}$$

5.2 Results and Discussion

We present the simulation results for a circular trajectory to demonstrate the effectiveness of the proposed method. In the simulation, an ODIN type vehicle (with higher thruster capacity) tries to track a circular path of diameter 2.65*m* in a horizontal plane in 20*sec*. The vehicle attitudes are kept constant at $[0^0\ 0^0\ 90^0]$. We have considered three different cases for this circular trajectory tracking task. In Case 1, all thrusters are in working condition. In Case 2, two of the thrusters (Thruster 1 and 5) develop faults during operation. Case 3 is similar to Case 2 except, in this case, the thruster saturation limits are imposed. In Case 1, all thrusters are in working condition and therefore the thruster fault matrix, Ψ, becomes an identity matrix. In Case 2 and 3, Thruster 5 stops functioning after 7*sec* and Thruster 1 stops functioning after 12*sec*. We have simulated it by incorporating zeros for the corresponding elements in the Ψ matrix.

It should be noted that the chosen circular trajectory tracking task is a much faster task (average speed = 0.808*knot*, Fig. 15) compared to the straight-line trajectory tracking task (average speed = 0.032*knot*) as discussed in Section 4.3. We wanted to see the performance of the proposed controller in a high-speed trajectory tracking with both thruster fault and thruster saturation. We could not risk the expensive ODIN for such a high-speed operation and thus, we provide only simulation results to demonstrate the efficacy of the proposed technique. Additionally, we could not experimentally verify the thruster saturation controller because ODIN was not equipped with any acceleration feedback mechanism.

We present the simulation results for the circular trajectory tracking task considering all the three cases: with no thruster fault, with thruster fault, and with thruster fault and thruster saturation in Fig. 15 and 16. We have simulated two thruster faults: one (Thruster 5, marked

by 'A') at 7*sec* and the other (Thruster 1, marked by 'B') at 12*sec*, Fig. 16. Both the faulty thrusters were chosen to be located at the same thruster bracket of the AUV. Thus, this fault was one of the worst fault conditions. We have imposed the following thruster saturation limits: $\pm 50N$ for vertical thrusters and $\pm 150N$ for horizontal thrusters. The task-space paths and trajectories are plotted in Fig. 15. It is observed that the trajectories are tracked quite accurately in all the three cases. However, the tracking errors are more for the thruster saturation case.

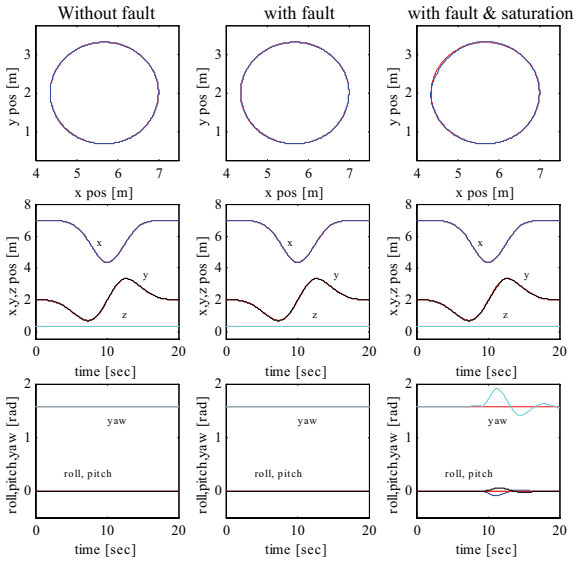


Fig. 15. Simulation results: Task-space (Cartesian) paths and trajectories, the solid lines denote the actual trajectories and the dashed lines denote desired trajectories.

The thruster forces are plotted in Fig. 16. From these plots, we can see that after the first fault, thrust for Thruster 7 becomes close to $-200N$. But by implementing the thruster saturation algorithm, we are able to keep this thrust within the specified limit ($\pm 150N$). In this process, the thrusts for Thruster 6 and Thruster 8 reach the saturation limits, but do not cross it. As a result, we observe larger errors in the trajectory tracking during this time for the saturation case (Fig. 15). However, the controller brings back the AUV in its desired trajectories and the errors are gradually reduced to zero.

6. Drag Minimization Algorithm

A UVMS is a kinematically redundant system. Therefore, a UVMS can admit an infinite number of joint-space solutions for a given task-space coordinates. We exploit this particular characteristic of a kinematically redundant system not only to coordinate the motion of a UVMS but also to satisfy a secondary objective criterion that we believe will be useful in underwater applications.

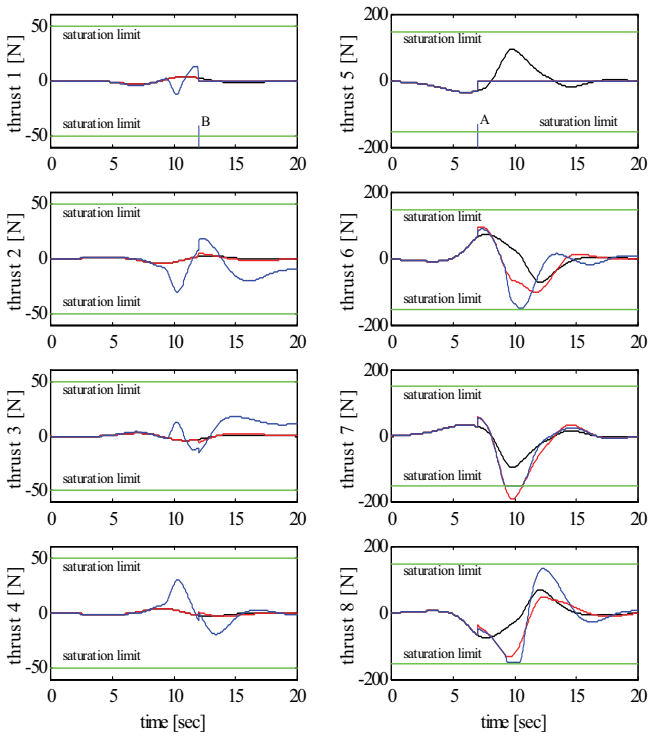


Fig. 16. Simulation results: Thrust versus time, no fault (Case 1, denoted by dashed-dot lines), with faults (Case 2, denoted by dashed lines), and saturation (Case 3, denoted by solid lines).

The secondary objective criterion that we choose to satisfy in this work is hydrodynamic drag optimization. Thus, we want to design a motion planning algorithm that generates trajectories in such a way that the UVMS not only reaches its goal position and orientation from an initial position and orientation, but also the drag on the UVMS is optimized while it follows the generated trajectories. Drag is a dissipative force that does not contribute to the motion. Actually, a UVMS will require a significant amount of energy to overcome the drag. Since the source of energy for an autonomous UVMS is limited, which generally comes from the batteries that the UVMS carries with it unlike a ROV and tele-manipulator system where the mother ship provides the energy, we focus our attention to reduce the drag on the system. Reduction of drag can also be useful from another perspective. The UVMS can experience a large reaction force because of the drag. High reaction force can saturate the controller and thus, degrade the performance of the system. This problem is less severe when human operators are involved because they can adjust their strength and coordination according to the situation. However, for an autonomous controller, it is better to reduce such a large force especially when the force is detrimental to the task.

6.1 Theoretical Development

Recalling Equation (11) and Equation (13), we can write the complete solution to the joint-space acceleration as (Ben-Israel & Greville, 1974):

$$\ddot{q}_{d_1} = J^{+}_{W_1}(\ddot{x}_{d_1} - \dot{J}\dot{q}_{d_1}) + (I - J^{+}_{W_1}J)\ddot{\phi}_1 \tag{38}$$

$$\ddot{q}_{d_2} = J^{+}_{W_2}(\ddot{x}_{d_2} - \dot{J}\dot{q}_{d_2}) + (I - J^{+}_{W_2}J)\ddot{\phi}_2 \tag{39}$$

where the null-space vectors $(I - J^{+}_{W_i}J)\ddot{\phi}_i$ (for i=1,2) will be utilized to minimize the drag effects on the UVMS.

We define a positive definite scalar potential function $p(q,\dot{q})$, which is a quadratic function of drag forces as

$$p(q,\dot{q}) = D^T(q,\dot{q})\, W_D\, D(q,\dot{q}) \tag{40}$$

where $D(q,\dot{q}) \in \Re^{(6+n)}$ is the vector of drag forces and $W_D \in \Re^{(6+n)\times(6+n)}$ is a positive definite weight matrix. Note that a proper choice of this W_D matrix can enable us to design the influence of drag on individual components of the UVMS. Generally, W_D is chosen to be a diagonal matrix so that the cross-coupling terms can be avoided. If it is chosen to be an identity, then the drag experienced on all dof of the combined system is equally weighted. However, increasing or decreasing the values of the diagonal elements of the W_D matrix, the corresponding drag contribution of each dof can be regulated. The potential function, $p(q,\dot{q})$, captures the total hydrodynamic drag on the whole vehicle-manipulator system. Therefore, the minimization of this function will lead to the reduction of drag on the whole system.

Now, taking the gradient of the potential function, $p(q,\dot{q})$, we obtain

$$\nabla p(q,\dot{q}) = \frac{\partial p(q,\dot{q})}{\partial q} + \frac{\partial p(q,\dot{q})}{\partial \dot{q}} \tag{41}$$

We take the gradient, $\nabla p(q,\dot{q})$, as the arbitrary vector, $\ddot{\phi}_i$, of Equation (38) and Equation (39) to minimize the hydrodynamic drag in the following form:

$$\ddot{\phi}_i = -\kappa_i \nabla p^T \quad for\ i = 1, 2. \tag{42}$$

where κ_i are arbitrary positive quantities, and the negative sign implies minimization of the performance criteria. A block diagram of the proposed control scheme is shown in Fig. 17. More detailed discussion on this drag minimization can be found in (Sarkar & Podder, 2001).

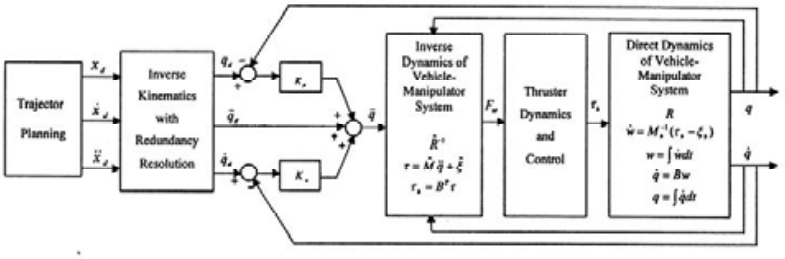


Fig. 17. Computer torque control scheme for drag minimization method.

6.2 Results and Discussion

We have conducted extensive computer simulations to investigate the performance of the proposed Drag Minimization (DM) algorithm. The details of the UVMS used for the simulation have been provided in Section 3.3. We have chosen a straight-line trajectory (length = 10m) in the task-space for the simulations. For the chosen trajectory, we have designed a trapezoidal velocity profile, which imposes a constant acceleration in the starting phase, followed by a cruise velocity, and then a constant deceleration in the arrival phase. The initial velocities and accelerations are chosen to be zero and the initial desired and actual positions and orientations are same. The simulation time is 15.0sec. Thus, the average speed of the UVMS is $0.67 m/s \approx 1.30 knot$. This speed is chosen to simulate the average speed of SAUVIM (Semi-Autonomous Underwater Vehicle for Intervention Mission), a UVMS being designed at the University of Hawaii, which has a maximum speed of 3$knot$.

In our simulation, we have introduced sensory noise in position and orientation measurements. We have chosen Gaussian noise of 1mm mean and 1mm standard deviation for the surge, sway and heave position measurements, 0.1deg mean and 0.1deg standard deviation for the roll, pitch and yaw position measurements for the vehicle, and 0.01deg mean and 0.01deg standard deviation for the joint position measurements for the manipulator. We have also incorporated a 15% modeling inaccuracy during computer simulations to reflect the uncertainties that are present in underwater environment. This inaccuracy has been introduced to observe the effect of both the uncertainty in the model and the neglected off-diagonal terms of the added mass matrix.

Thruster dynamics have been incorporated into the simulations using the thruster dynamic model described later in Section 7.2. The thruster configuration matrix is obtained from the preliminary design of SAUVIM type UVMS. It has 4 horizontal thrusters and 4 vertical thrusters. The thruster configuration matrix for the simulated UVMS is as follows:

$$E = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -R_{t3} & 0 & R_{t3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & R_{t4} & 0 & -R_{t4} & 0 & 0 & 0 \\ R_{t1} & -R_{t2} & -R_{t1} & R_{t2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (43)$$

where $R_{t1} = 1.25\,m$, $R_{t2} = 1.75\,m$, $R_{t3} = 0.75\,m$, and $R_{t4} = 1.25\,m$ are the perpendicular distances from the center of the vehicle to the axes of the side and the front horizontal thrusters, and the side and the front vertical thrusters, respectively. The thrusters for UVMS are chosen to be DC brushless thrusters, model 2010 from TECNADYNE. The thruster propeller diameter is 0.204m. It can produce approximately 580N thrust. The weight of each thruster is 7.9Kg (in water).

The simulation results are presented in Fig. 18 through Fig. 20. In Fig. 18, we have plotted both the desired and the actual 3D paths and trajectories. From the plots, it is observed that the end-effector of the manipulator tracks the desired task-space trajectories satisfactorily in both the PI and the DM methods. From the joint-space trajectories in Fig. 19, we can see that even though the UVMS follows the same task-space trajectories in both PI and DM methods, it does it with different joint-space configurations. This difference in joint-space configurations contributes to drag minimization as shown in Fig. 20. The total energy consumption of the UVMS has also been presented in Fig. 20. We find that the energy consumption is less in DM method as compared to that of in PI method. From these plots we observe that the drag on the individual components of UVMS may or may not be always smaller in DM method. But we can see in Fig. 20 that the total drag (norm of drag) on UVMS is less in DM method as compared to that of in PI method.

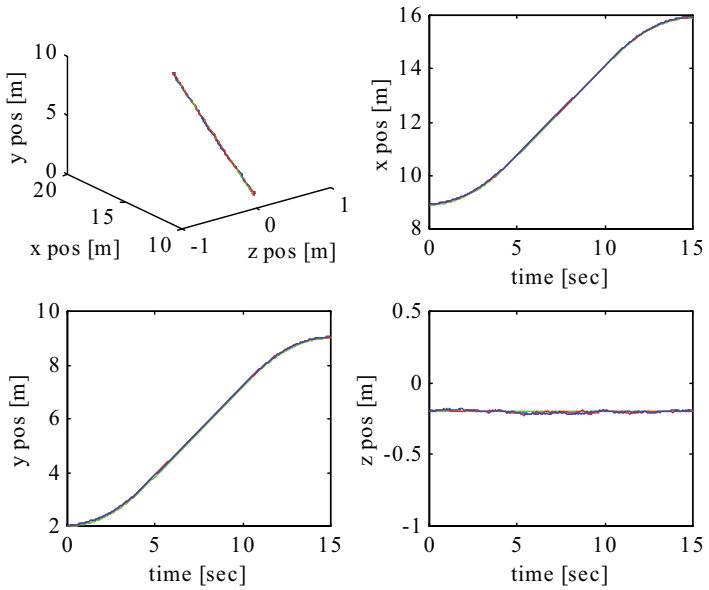


Fig. 18. Task-space (XYZ) straight-line trajectories of the end-effector of the robot manipulator, solid lines denote DM (drag minimization) method, dashed lines denote PI (pseudoinverse) method, and dashed dot lines denote the desired trajectories.

Here we have designed a model-based controller to follow a set of desired trajectories and presented results from computer simulations to demonstrate the efficacy of this newly proposed motion planning algorithm. In this context we must mention that a purely model-based controller may not be ideal for underwater applications. However, since the main thrust of this study is in motion planning, we have used this model-based controller only to compare the effectiveness of the proposed Drag Minimization algorithm with that of more traditionally used Pseudoinverse algorithm.

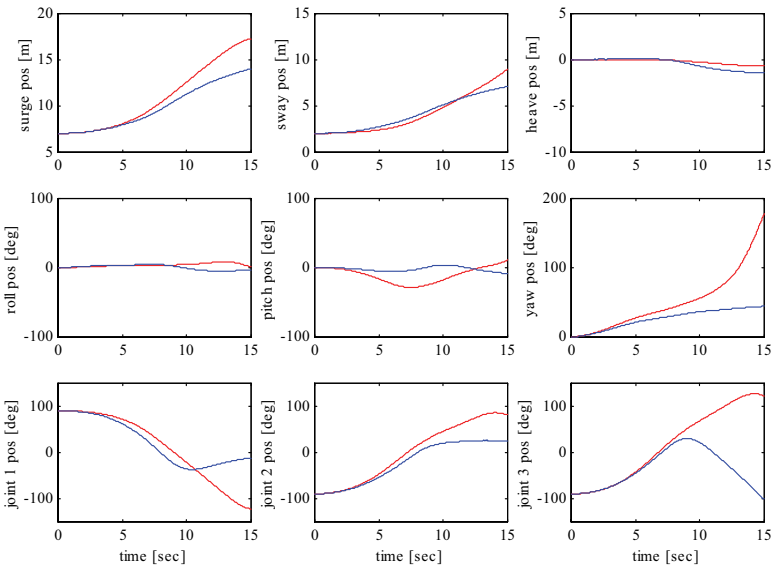


Fig. 19. Joint-space positions of the UVMS, solid lines denote DM (drag minimization) method, dashed lines denote PI (pseudoinverse) method.

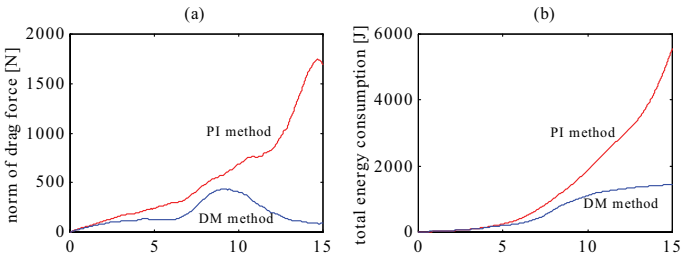


Fig. 20. (a) Norm of drag force, (b) total energy consumption of the UVMS, solid lines denote DM (drag minimization) method, dashed lines denote PI (pseudoinverse) method.

7. Unified Dynamics-Based Motion Planning Algorithm

7.1 Theoretical Development

A schematic diagram of the proposed unified dynamics-based control scheme is given in Fig. 21. For a unified dynamics-based algorithm, let us look back to Equations (38) and (39) along with Equations (40)-(42) which provide us with the reference joint-space trajectories considering the dynamics-based planning method as well as the drag minimization scheme. Now, we can obtain all the desired joint-space variables required for the control law (Eq. (31) or Eq. (37)) by integrating Equation (38) and Equation (39) and making use of Equation

(13). Then by differentiating it we can obtain the desired third derivative for the joint-space variables. Thus, we have formulated a unified motion planning algorithm by integrating the dynamics-based planning algorithm (Eq. (11)-(13)) with fault-tolerant algorithm (Eq. (29)), saturation algorithm (Eq. (30)-(31)), and drag minimization algorithm (Eq. (38)-(39)).

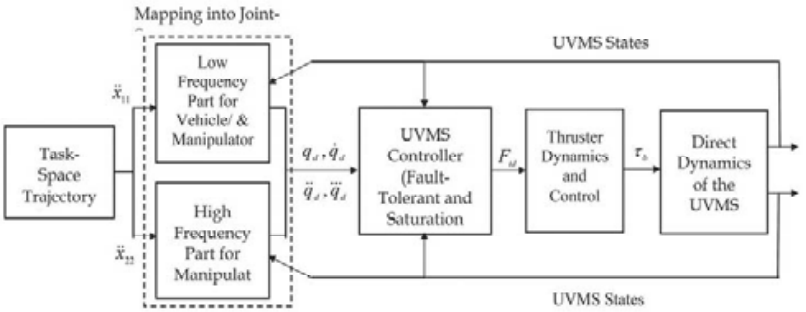


Fig. 21. Unified dynamics-based motion planning scheme.

7.2 Thruster Dynamics

The desired thruster force allocation as obtained from Equation (23) can be directly applied to the dynamic model of the UVMS given by Equation (1) (using Equation (15)) to generate the actual motion of the system. However, in such a case the dynamics of the thrusters will be neglected and the results will not accurately reflect the reality. Yoerger *et al*. (Yoerger *et al*., 1990) pointed out that the system dynamics of an underwater vehicle can be greatly influenced by the dynamics of the thrusters, and neglecting this dynamics may result in a limited bandwidth controller with limit cycle instability. There are several dynamic models of marine thrusters (Yoerger *et al*., 1990; Healey et al., 1995; Whitcomb & Yoerger, 1999) that can reliably account for thruster dynamics.

In this work we use the model proposed by Healey *et al*. (Healey et al., 1995) that included a four-quadrant mapping of the lifts and drag forces of the propeller blades and was coupled with the motor and fluid system dynamics. This model is given by the following equations:

$$\Omega_r = \alpha_2^{-0.5} sign(F_{td}) \left| F_{td} \right|^{0.5} \tag{44}$$

$$i_m = K_t^{-1} \alpha_1 F_{td} + K_f^{-1} K_{fb} (\Omega - \Omega_r) \tag{45}$$

$$\dot{\Omega} = I^{-1} [K_t - i_m - K_f \Omega - \Im] \tag{46}$$

where Ω and Ω_r are the actual and the desired/reference propeller angular velocity, respectively, and i_m is the motor current. The other parameters are: $\alpha_2 = \rho A r^2 \eta^2 \tan^2(\gamma)$, where ρ is the density of the water, r is the radius of the propeller, A is the thruster duct area, η is the propeller efficiency, γ is the average pitch of the propeller blade, α_1 is an experimentally determined constant, K_t is the motor torque constant, K_f is the motor viscous friction constant, K_{fb} is the motor feedback gain, and $\Im$ is the propeller shaft torque. Neglecting the motor inductance (Healey et al., 1995), the motor input voltage can be written as

$$V_m = i_m R_m + K_{emf} \Omega \tag{47}$$

where V_m is the motor input voltage, R_m is the motor resistance and K_{emf} is the motor back emf constant.

The propeller torque and the axial thrust are related to the blade lift, *L* and the drag, *D* as follows:

$$\Im = 0.7rL\sin\theta + D\cos\theta \tag{48}$$

$$F_{t,act} = L\cos\theta - D\sin\theta \tag{49}$$

where $F_{t,act}$ is the propeller shaft thrust, $\theta = \gamma - \alpha$, and α is the angle of attack.

7.3 Results and Discussion

We have performed extensive computer simulation to investigate the efficacy of the proposed Unified Dynamics-based Motion Planning (UDMP) algorithm. To verify the effectiveness of the proposed method, we have compared the results of UDMP approach with that of Conventional Motion Planning (CMP) method. In conventional method, the trajectory is designed in three sections: the main section (intermediate section) that is a straight line is preceded and followed by two short parabolic sections. The UVMS used for these simulations is same as mentioned in Section 3.3. The simulation time is 10*sec* that is required to complete the square path. The total length of the path is 8*m*, thus the average speed is about 1.6*knot*. This speed is close to JASON II vehicle (speed=1.5*knot*).

We have simulated two thruster faults: one horizontal thruster (Thruster 1) and the other one vertical thruster (Thruster 5). Both the thrusters stop functioning from 6*sec*. It is to be noted that both the thrusters are located at the same bracket of the UVMS, which is one of the worst thruster fault situations. In our simulation, we have considered the following thruster/actuator thrust/torque saturation limits: $\pm 400N$ for horizontal thrusters (Thruster 1-4), $\pm 200N$ for vertical thrusters (Thruster 5-8), $\pm 200N.m$ for actuator 1, $\pm 100N.m$ for actuator 2 and $\pm 50N.m$ for actuator 3.

To make the simulation close to reality, we have introduced sensory noise in the measurements of positions and its derivatives. We have considered Gaussian noise of 1 mean and 1 standard deviation in the measurement of linear quantities (in *mm* unit), and 0.01 mean and 0.05 standard deviation in measurement of angular quantities (in *deg* unit). We have considered 10% modeling inaccuracy during computer simulation to reflect the uncertainties that are present in underwater environment.

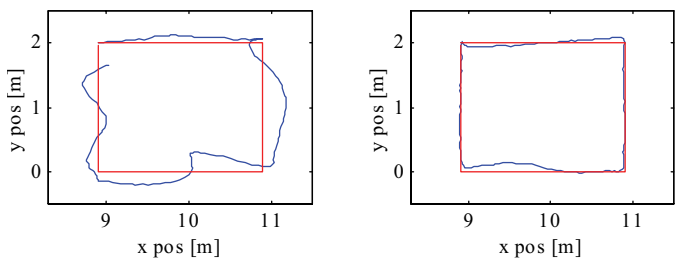


Fig. 22. Task-space geometric paths: Conventional Motion Planning (CMP) method in the left and Unified Dynamics-based Motion Planning (UDMP) method in the right. Doted lines denote the desired paths and solid lines denote actual paths.

We have presented results from the computer simulations in Fig. 22 through Fig. 26. The results we have provided here are from *Case I: Partial Decomposition* of the proposed UDMP method. The

task-space geometric paths are plotted in Fig. 22, where we can see that the path tracking errors in our proposed UDMP method are much smaller as compared to that of CMP method. We have also plotted task-space trajectories in Fig. 23. It is also observed from plots in Fig. 23 that the end-effector tracks the task-space trajectories quite accurately in UDMP method. The errors are less in proposed UDMP method as compared to the CMP method. The joint-space trajectories are plotted in Fig. 24. From these plots it is observed that the proposed UDMP method effectively reduces the motions of the heavy subsystem (the vehicle) and allows greater and sharper motions to the lighter subsystem (the manipulator) while tracking the same task-space trajectories. It is also noticed that the motion of the heavy subsystem is smoother in the proposed method. We find that these sharper and larger motions of the heavy subsystem in case of CMP method demand higher driving force that we see in Fig. 25. From the plots in this Fig. (Fig. 25) it is also observed that in case of UDMP method thrusters 4, 7, 8 and actuator 1 have reached the saturation limits, but they have not exceeded the limits. On the other hand, in case of CMP method all the thrusters and actuators have reached the saturation limits, however the saturation scheme was able to keep them to within the specified limits. Because of this, the path and trajectory tracking performance in CMP method has been degraded, as we can see in Fig. 22 and Fig. 23. Thus, the conventional planning method demands more powerful actuation system to track the same trajectories with reasonable accuracy. We also observe that the thrust 1 and thrust 5 are zero from 6*sec* as marked by "A" and "B", respectively (see Fig. 25). These imply they have developed faults at 6th second and remain non-functional for rest of the time. At this moment we observe some perturbations in trajectories and paths, however, the proposed UDMP scheme gradually brings the system to the desired directions and reduces the tracking errors. On the other hand, after the occurrence of faults the paths and the trajectories are tacked poorly in case of CMP method, because this algorithm cannot account for the dynamics of the system while generating the reference trajectories.

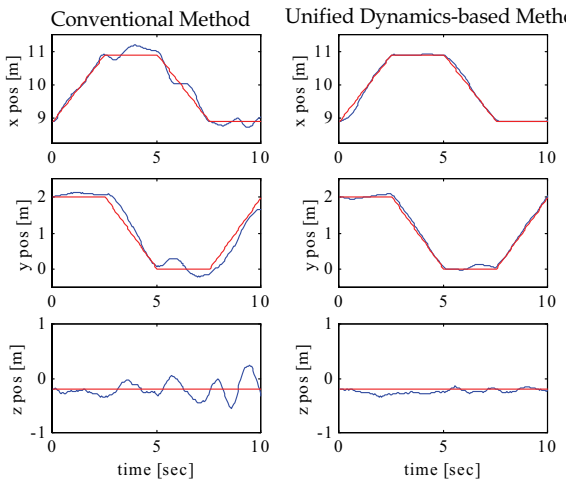


Fig. 23. Task-space trajectories: Conventional Motion Planning method (left column) and Unified Dynamics-based Motion Planning method (right column). Actual trajectories (solid lines) are superimposed on desired trajectories (doted lines).

We have also plotted the simulation results for surge-sway motion, power requirement and energy consumption of the UVMS in case of CMP method (in the left column) and that of in case of proposed UDMP method (in the right column) in Fig. 26. Top two plots in this figure show the profile of the surge- sway movements of the vehicle in the said two methods. In case of the CMP method, the vehicle changes the motion sharply and moves more as compared to the motion generated from the UDMP method. It may so happen that, in practice, this type of sharp and fast movements may be beyond the capability of the heavy dynamic subsystem and consequently large errors in trajectory tracking will occur. Additionally, this may cause saturation of the thrusters and the actuators resulting in degradation in performance. Moreover, the vehicle will experience large velocity and acceleration in CMP method that result in higher power requirement and energy consumption, as we observe it in next two sets of plots in Fig. 26. Thus, this investigation reveals that our proposed Unified Dynamics-Based Motion Planning method is very promising for autonomous operation of dynamic system composed of several subsystems having variable dynamic responses.

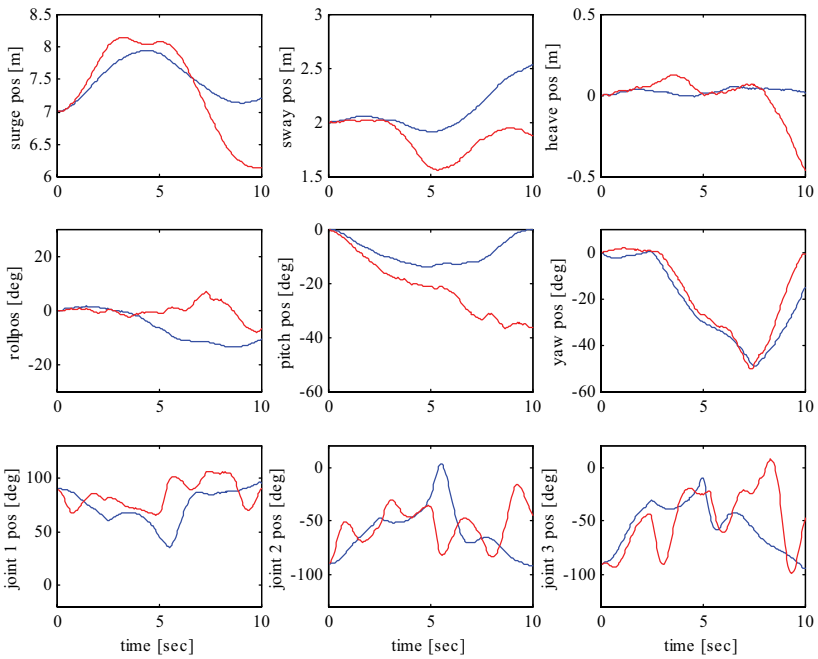


Fig. 24. Joint-space trajectories: Unified Dynamics-based Motion Planning method (solid lines) and Conventional Motion Planning method (dashed lines).

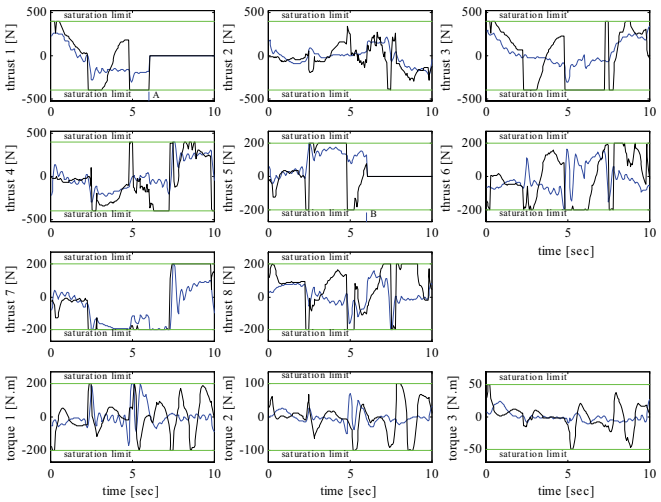


Fig. 25. Thruster and actuator forces and torques of the UVMS. Unified Dynamics-based Motion Planning method (solid lines) and Conventional Motion Planning method (dashed lines). Thruster faults are marked by "A" (Thruster 1) and "B" (Thruster 5).

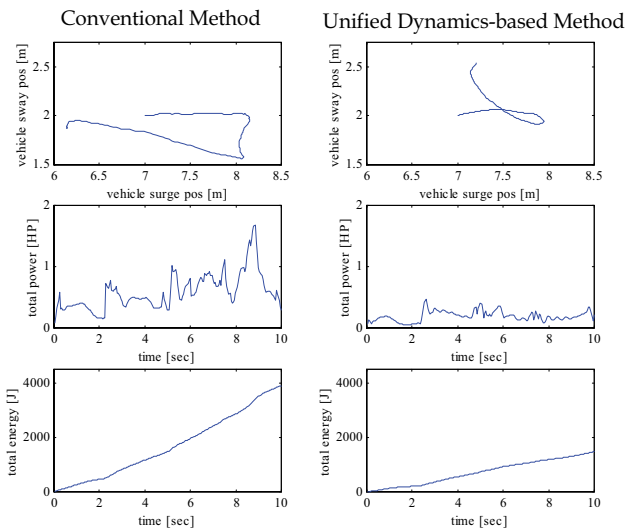


Fig. 26. Surge-sway motion of the vehicle, power requirement and energy consumption of the UVMS. Results from Conventional Motion Planning method are in the left and that of Unified Dynamics-based Motion Planning method are in the right.

8. Conclusions

We have proposed a new unified dynamics-based motion planning algorithm that can generate both kinematically admissible and dynamically feasible joint-space trajectories for systems composed of heterogeneous dynamics. We have then extended this algorithm for an autonomous underwater vehicle-manipulator system, where the dynamic response of the vehicle is much slower than that of the manipulator. We have also exploited the kinemetic redundancy to accommodate the thruster/actuator faults and saturation and also to minimize hydrodynamic drag. We have incorporated thruster dynamics when modeling the UVMS. Although, some researchers have exploited kinematic redundancy for optimizing various criteria, but those work have mainly addressed to problems with land-based robotics or space-robotics. Hardly any motion planning algorithm has been developed for autonomous underwater vehicle-manipulator system. In this research, work we have formulated a new unified motion planning algorithm for a heterogeneous underwater robotic system that has a vastly different dynamic bandwidth. The results from computer simulation demonstrate the effectiveness of the proposed method. It shows that the proposed algorithm not only improves the trajectory tracking performance but also significantly reduce the energy consumption and the power requirements for the operation of an autonomous UVMS. We have not presented results from Case II (*Total Decomposition*) because of the length of the paper. However, these results are comparable to the conventional motion planning approach. In future, instead of Fourier decomposition, one can try to use wavelet approach to decompose the task-space trajectory into system's sub-component compatible segments.

There are a few drawbacks of this paper as well. We used a model-based control technique to evaluate our planning algorithm. However, the underwater environment is uncertain and we need to use adaptive control techniques in future. Although the fault-tolerant control algorithm has been experimentally verified, the other proposed algorithms need to be validated by experiments.

9. References

Klein, C.A. & C.H. Huang, C.H. (1983). Review of pseudoinverse control for use with kinematically redundant manipulators, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-13, No. 3, pp. 245-250.

Paul, R. (1979). Manipulator Cartesian path planning, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-9, No. 11, pp. 702-711.

Zhou, Z.L. & Nguyen, C.C. (1997). Globally optimal trajectory planning for redundant manipulators using state space augmentation method, *Journal of Intelligent and Robotic Systems*, Vol. 19, No. 1, pp. 105-117.

Siciliano, B. (1993). Closed-loop inverse kinematics algorithms for redundant spacecraft/ manipulator systems, *Proceedings of IEEE International Conference on Robotics and Automation,* pp. 95-100, Atlanta, GA, May 1993.

Antonelli, G. & Chiaverini, S. (1998). Task-priority redundancy resolution for underwater vehicle-manipulator systems, *Proceedings of IEEE International Conference on Robotics and Automation,* pp. 756-761, Leuven, Belgium, May 1998.

Vukobratovic, M. & Kircanski, M. (1984). A dynamic approach to nominal trajectory synthesis for redundant manipulators, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC 14, No. 4, pp. 580-586, July 1984.

Bobrow, J.E. (1988). Optimal robot path planning using the minimum-time criterion, *IEEE Journal of Robotics and Automation*, Vol. 4, No. 4, pp. 443-450, August 1988.

Shiller, Z. & Dubowsky, S. (1989). Robot path planning with obstacles, actuator, gripper, and payload constraints, *International Journal of Robotics Research*, Vol. 8, No. 6, pp. 3-18, December 1989.

Shin, K. & McKay, N. (1986). A dynamic programming approach to trajectory planning of robot manipulators, *IEEE Transactions on Automatic Control*, Vol. AC-31, No. 6, pp. 491-500, June 1986.

Hirakawa, A.R. & Kawamura, A. (1997). Trajectory planning of redundant manipulators for minimum energy consumption without matrix inversion, *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 2415-2420, Albuquerque, New Mexico, April 1997.

Saramago, S.F.P. & Steffen Jr. V. (1998). Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system, *Journal of Mechanism and Machine Theory*, Vol. 33, No. 7, pp. 883-894, 1998.

Zhu, Z.H.; Mayorga, R.V. & Wong, A.K.C. (1999). Dynamic robot manipulator trajectory planning for obstacle avoidance, *Journal of Mechanics Research Communications*, Vol. 26, No. 2, pp. 139-144, 1999.

Faiz, N. & Agrawal, S.K. (2000). Trajectory planning of robots with dynamics and inequalities, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3977-3983, San Francisco, CA, April 2000.

Macfarlane, S. & Croft, E.A. (2003). Jerk-bounded manipulator trajectory planning: design for real-time applications, *IEEE Transactions on Robotics and Automation*, Vol. 19, No. 1, pp. 42-52, February 2003.

Brock, O. & Khatib, O. (1999). High-speed navigation using the global dynamic window approach, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 341-346, Detroit, MI, May 1999.

Huang, Q.; Tanie, K. & Sugano, S. (2000). Coordinated motion planning for a mobile manipulator considering stability and manipulation, *International Journal of Robotics Research*, Vol. 19, No. 8, pp. 732-742, August 2000.

Yamamoto, Y. & Fukuda, S. (2002). Trajectory planning of multiple mobile manipulators with collision avoidance capability, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3565-3570, Washington D.C., May 2002.

Yamashita, A.; Arai, T.; Jun, O. & Asama, H. (2003). Motion planning of multiple mobile robots for cooperative manipulation and transportation, *IEEE Transactions on Robotics and Automation*, Vol. 19, No. 2, pp. 223-237, April 2003.

Yoerger, D.R. & Slotine, J.J.E. (1985). Robust trajectory control of underwater vehicles, *IEEE Journal of Oceanic Engineering*, Vol. 10, No. 4, pp. 462-470, October 1985.

Spangelo, I. & Egeland, O. (1994). Trajectory planning and collision avoidance for underwater vehicle using optimal control, *IEEE Journal of Oceanic Engineering*, Vol. 19, No. 4, pp. 502-511, 1994.

Kawano, K. & Ura, T. (2002). Motion planning algorithm for nonholonomic autonomous underwater vehicle in disturbance using reinforced learning and teaching method," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4032-4038, Washington D.C., May 2002.

Sarkar, N. & Podder, T.K. (2001). Coordinated motion planning and control of underwater vehicle-manipulator systems subject to drag optimization," *IEEE Journal of Oceanic Engineering*, Vol. 26, No. 2, pp. 228-239, 2001.

Fu, K.S. Gonzalez, R.C. & Lee, C.S.G. (1988). *Robotics: Control, Sensing, Vision, and Intelligence,* McGraw-Hill International Edition, 1988.

Craig, J.J. (1989). *Introduction to Robotics: Mechanics and Control*, Addison-Wesley Publishing Company, 1989.

Kane, T.R. & Lavinson, D.A. (1985). *Dynamics: Theory and Applications*, McGraw-Hill Book Co., 1985.

Fossen, T.I. (1984). *Guidance and Control of Ocean Vehicles*, John Willey & Sons Ltd., 1994.

Podder, T,K. (2000). *Motion Planning of Autonomous Underwater Vehicle-Manipulator Systems*, Ph.D. Thesis, Department of Mechanical Engineering, University of Hawaii at Manoa, Honolulu, December 2000.

James, M.L.; Smith, G.M.; Wolf, J.C. & Whaley, P.W. (1989). *Vibration of Mechanical and Structural Systems*, Harper and Row Publishers, New York, 1989.

Podder, T.K. & Sarkar, N. (2000). Fault tolerant control of an autonomous underwater vehicle under thruster redundancy: simulation and experiments, *Proceedings of IEEE International Conference on Robotics and Automation*, pp.1251 -1256, San Francisco, April 2000.

Podder, T.K.; Antonelli, G. & Sarkar, N. (2001). An experimental investigation into the fault-tolerant control of an autonomous underwater vehicle, *Journal of Advanced Robotics*, Vol. 15, No. 5, pp. 501-520, 2001.

Isidori, A.; Moog, C.H. & De Luca, A. (1986). A sufficient condition for full linearization via dynamic state feedback, *Proceedings of IEEE Conference on Decision and Control*, pp. 203-208, Athens, Greece, December 1986.

Yun, X. (1988). Dynamic state feedback control of constrained robot manipulators, *Proceedings of IEEE Conference on Decision and Control*, pp. 622-626, Austin, Texas, December 1988.

Ben-Israel, A. & Greville, T.N.E. (1974). *Generalized Inverse: Theory and Applications*, New York, Wiley, 1974.

Yoerger, D.R.; Cooke, J. & Slotine, J.J.E. (1990). The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design, *IEEE Journal of Oceanic Engineering*, Vol. 15, No. 3, pp. 167-178, 1990.

Healey, A.J.; Rock, S.M.; Cody, S. Miles, D. & Brown, J.P. (1995). Toward an improved understanding of thruster dynamics for underwater vehicles, *IEEE Journal of Oceanic Engineering*, Vol. 20, No. 4, pp. 354-361, 1995.

Whitcomb, L.L. & Yoerger, D.R. (1999). Development, comparison, and preliminary validation of non-linear dynamic thruster models, *IEEE Journal of Oceanic Engineering*, Vol. 24, No. 4, pp. 481-493, 1999.

16

Optimal Velocity Planning of Wheeled Mobile Robots on Specific Paths in Static and Dynamic Environments

María Prado
Dept. Mechanical Engineering, University of Malaga
Spain

1. Introduction

The control system of a mobile robot generally comprises two different modules: a trajectory planner and a trajectory tracking controller, although some researchers have proposed algorithms that integrate both tasks.
To completely solve the trajectory planning problem is to define an open-loop path and its velocity profile from an initial to a final posture, while avoiding any potential obstacles.
In time-optimal planning of a wheeled mobile robot (WMR), the problem is solved by defining control inputs for the wheels that minimize navigation time from the origin to the target posture. This goal implies two tasks, which can be carried out simultaneously or sequentially: path-planning (PP), which involves the computation of the shortest feasible path; and velocity-planning (VP), which involves the computation of the fastest feasible velocity profile for the entire domain of the path.
Several approaches have been developed to perform both tasks. The most widely used approaches are free configuration-time space based methods, (Reinstein & Pin, 1994), but these algorithms are computationally expensive, even when one is only dealing with PP or VP separately. To reduce the computational cost, researchers have recently published methods which do not require computing the C-space obstacles (Wang et al., 2004), as well as methods that search for a probabilistic road map (LaValle & Kuffner, 2001). Some other approaches that use intelligent computing-based methods have also been presented, such as those that use artificial potential fields-based methods (Liu & Wu, 2001), fuzzy logic (Takeshi, 1994), genetic algorithms (Nerchaou, 1998) or neural networks (Zalama et al., 1995).
In order to find an optimal and feasible solution for the two problems, mechanical, kinematic and dynamic characteristics of the WMR that limit its motion must be taken into account, as well as other environmental, task-related and operational issues. These constraints can be summarized by upper boundary functions of the velocity, acceleration and deceleration of the WMR. In general, the functions are not constant, nor are they even continuous. They are therefore nonintegrable constraints, and the time optimal planning is a nonholonomic problem.
A significant number of nonholonomic constraints, which include not only mechanical and kinematic but also dynamic characteristics of the WMR, are difficult to deal with when PP and VP are approached simultaneously. The vast majority of existing algorithms consider

only kinematic constraints or some dynamic conditions derived from simplified models of the WMR and/or its environment. But the resulting trajectory may be unexecutable, or tracked by the robot with high spatial and temporal errors. However, when PP and VP are approached sequentially, the difficulty of both problems is significantly reduced. Such approaches make it possible to include more complex constraints for the WMR's velocity and acceleration, especially with regards to its kinematic and dynamic characteristics.

To our knowledge, the first references addressing VP with kinematic and dynamic constraints for WMR is (O'Dunlaing, 1987). This paper, like a number of other algorithms to solve the VP stage, is based on constant maximum values for robot velocity and acceleration, set to arbitrary constants which are unrelated to the mechanical characteristics of the system. More recent works seek to find more efficient bounds for these operating variables, but never in a global way and always based on simplified dynamic robot models. (Weiguo et al., 1999) propose a velocity profile planner for WMRs on flat and homogeneous terrains, where velocity and acceleration are limited only by the outer motor torques and by the absolute slippage of the vehicle on the ground. (Choi & Kim, 2001) develop another planner where velocity and acceleration are constrained by dynamic characteristics related to the performance of the robot's electric motors and its battery's power. (Guarino Lo Bianco & Romano, 2005) present a VP algorithm for specific paths that generate a continuous velocity and acceleration profile, both into safety regions limited by upper boundary functions not described in the paper. The method involves an optimization procedure that has a significant computational cost.

Some other limitations have been studied, mainly within the framework of projects for planetary exploration. (Shiller, 1999) deals with some dynamic constraints: sliding restrictions, understood as the avoidance of absolute vehicle slippage, tip-over and loss of wheel-ground contact constraints, which are important issues when dealing with irregular outdoor terrains. The author works with a very simplified robot model, neglecting sideslip and assuming pure rolling, so wheel deformations and microslippages which can cause important tracking errors are not quantified. (Cheriff, 1999) also proposes a set of kinematic and dynamic constraints over the robot's path, dealing specifically with 3D irregular and non-homogeneous grounds. The resulting trajectory planner directly incorporates a complete dynamic WMR model, considering non-linear motions and specifically accounting for wheel-ground interactions, which makes it necessary to run complex algorithms that significantly increase computational cost.

(Lepetic et al., 2003) present a VP method that considers dynamic constraints by bounding the acceleration by the maximum wheel-ground adherence capacity. This maximum is computed as a function of a constant friction coefficient for every posture and of the weight borne by the wheel. Load transfer due to lateral forces is considered to calculate the weight on the wheel, but only as a constant maximum value, derived from a simplified model of the WMR, that reduces the lateral maximum acceleration to the same value for every posture. The VP method published by (Krishna et al., 2006) builds a trajectory continuous in space and velocity, which incorporates environment and sensory constraints by setting a maximum velocity for the entire path of the robot that is decreased when an obstacle is detected within its visibility circle. The velocity constraint is computed as a function of the position and velocity of the obstacle and of a maximum acceleration or deceleration value of the WMR, established as constant values for every posture.

This chapter deals with time-optimal planning of WMRs when navigating on specific spatial paths, i.e., when the PP is previously concluded. First, the computation of the upper

boundary functions of its velocity, acceleration and deceleration are described. Then a method for time-optimal planning is proposed, the main goals of which are:

- To fully exploit velocity, acceleration and deceleration constraints, avoiding the planning of velocities or accelerations that lead to dangerous motions.
- To plan a feasible trajectory, with continuous velocity and deceleration
- To bound the jerk of the WMR
- To be of low computational cost.

The method firstly deals with velocity planning in static environments and then presents an algorithm to modify the resulting trajectory to avoid moving obstacles. Special attention is paid to the efficiency of the second algorithm, an advantage which makes it highly useful for local and/or reactive control systems.

2. Problem definition

Problem 1: Given a WMR's path, computed to navigate in a static and known environment, plan the fastest, feasible and safe trajectory, considering the constraints imposed by the mechanical configuration, kinematics and dynamics of the robot and by environmental and task-related issues.

Problem 2: Modify the trajectory quickly and locally to avoid moving obstacles.

A generalized posture of a WMR, parameterizing by the path length, s, can be defined by the vector $q(s) = [X(s), Y(s), \theta(s), \delta(s)]^T$. [X(s), Y(s)] is the position and θ(s) the orientation of the WMR's guide point on a global frame (Z coordinate is constant by assuming navigation is on flat ground). δ(s) is a function kinematically related to the curvature of the trajectory, κ(s); specifically, it is a function of the steer angles of the wheels of WMRs with steering wheels or a function of the difference between the angular velocities at the traction wheels for WMRs with differential drive.

The path, P(s), can be defined by a continuous series of generalized postures from the initial posture, q_0, to the final one, q_f. Therefore, if S is the total length of the path:

$$P(s) = \{q(s)\} : [0, S] \rightarrow \Re^4 ; P(0) = q_0 \vee P(S) = q_f \tag{1}$$

To transform P(s) into a trajectory, a velocity function must be generated for the entire path domain. It must be defined in positive real space (if the WMR is only required to move forward, as is the usual case) and planned to make the robot start from a standstill and arrive at the final posture also with null velocity. That is:

$$V(s) = \{v(s)\} : [0, S] \rightarrow \Re^+ ; V(0) = 0 \vee V(S) = 0 \tag{2}$$

Additional conditions are strongly required of V(s) to obtain a feasible trajectory:

1. Continuity, since the kinematics of WMR make it impossible to develop other types of maneuvers.
2. Confinement into a safety region of the space-velocity plane (s×v), upper limited by a boundary function of the velocity, $V_{Lim}(s)$.
3. Confinement of its first derivative with respect to time, acceleration or deceleration, into a safety region of the space-acceleration plane (s×a), upper limited by a boundary function of the acceleration, $a_{Lim}(s)$, and lower limited by the negative value of a boundary function of deceleration $d_{Lim}(s)$.

4. Continuity of acceleration or deceleration: this condition ensures that the jerk of the robot, the second derivative of its velocity, is finite, so that the robot's movements are smooth. High jerk is not recommended for WMRs for a number of reasons: it causes the robot to shake significantly and thus complicates on-board tasks; it makes tracking control more difficult, since wheel microslippage increases and wheel behavior becomes less linear (Wong, 2001); and it increases the error of on-board sensor systems.
5. Additionally, low computational cost is beneficial for the generation of the velocity profile. This goal is especially pursued when solving problem 2, for the purpose of possibly incorporating the algorithm into local controls or reactive planners, to adjust the trajectory in the presence of new unexpected obstacles that appear in the visibility area of the robot's sensorial systems (Krishna et al., 2006).

3. Velocity constraints

This section deals with constructive characteristics, kinematic configuration and the dynamic behaviour of a WMR, as well as operational matters, in order to identify the constraints that influence the maximum velocity of a WMR's guide point.
For all the constraints detailed in the following subsections, an upper boundary function of velocity, parametrized by s, can be generated. The function is built by assigning the lowest upper bound of all the velocity constraints to each posture:

$$V_{Lim}(s) = \min\left\{V_{Lim1}, V_{Lim2} \text{ ..... } V_{Limn}\right\} \quad / s \subset [0, S] \tag{3}$$

This chapter addresses the case of a WMR guided by steering wheels; in the case of WMRs with differential drive, the approach will be similar and therefore the constraints can easily be deduced under the same considerations.

3.1. Construction constraints

Thermal and mechanical characteristics of motors and batteries impose maximum rotational velocities on the tractive and steering servomotors, ω_{tm}^{max} and ω_{sm}^{max}, respectively (Choi & Kim 2001). Thus, if ξ_t is the reduction ratio of the drive-train and R the wheel's radius, the maximum linear velocity of driven wheels on the ground is:

$$v_{tw}^{max} = \xi_t \omega_{tm}^{max} R \tag{4}$$

Further, if ξ_s is the reduction ratio of the steering-train, the maximum velocity of variation of the steering angle, i.e. the maximum steering gain, is:

$$G_s^{max} = \xi_s \left|\omega_{sm}^{max}\right| \tag{5}$$

3.2. Kinematic constraints

With regards to kinematic linkages between the driven wheels and the guide point, if d_{tw}^{max} is the position vector on the ground of the most distant driven wheel with respect to the guide point, an upper bound for the WMR's velocity is given by:

$$V_{Lim1} = v_{tw}^{max} \frac{\left|1/\vec{\kappa}\right|}{\left|1/\vec{\kappa} + \vec{d}_{tw}^{max}\right|} \tag{6}$$

On the other hand, by considering kinematic linkages between the steering wheels and the guide point, a second boundary function for the velocity is found as:

$$V_{Lim2} = \frac{\left|d\kappa/dt\right|}{\left|d\kappa/ds\right|} \tag{7}$$

The numerator must be calculated from a kinematic model of the robot, whereas the denominator can be directly computed from the known spatial path.

3.3. Dynamic constraints

A dynamic model of the robot is needed to generate boundary functions relating to its dynamic characteristics. Since this model is only used to fully define the VP algorithm, specifically when defining V_{Lim}, a_{Lim} and d_{Lim}, but not when computing the trajectory, it can be as complex as needed for successful results without increasing computational cost. One may therefore use a model which is not limited by its degrees of freedom, geometric non-linearities, integration tolerances, etc...

3.3.1. Maximum velocity to bound spatial error

Let the quadratic spatial error of a WMR be the square of the distance from the actual position of its guide point tracking a trajectory to the position planned by the PP, measured on the ground plane and parameterised by the normalised arc length, $\tilde{s}$, defined as the ratio of s to the total path length, S, i.e. $\tilde{s} = s/S$.
Let the actual tracked trajectory, which will involve a side-slip angle with a value that is generally non-zero, be expressed by a two dimensional function on a world reference frame as:

$$P_A(\tilde{s}) = \left[X_\beta, Y_\beta, \theta_\beta, \delta_\beta\right]: \quad [0, 1] \rightarrow \Re^4 \tag{8}$$

Then, the quadratic spatial error can be calculated by:

$$E_s^2(\tilde{s}) = \sqrt{(X(\tilde{s}) - X_\beta(\tilde{s}))^2 + (Y(\tilde{s}) - Y_\beta(\tilde{s}))^2} \tag{9}$$

And the total quadratic spatial error is the integral of (9) over the entire path:

$$TE_s^2(\tilde{s}) = \int_0^1 E_s^2(\tilde{s}) d\tilde{s} \tag{10}$$

If the planned path, P(s) in (1) is particularized for stationary manoeuvres, i.e. with constant velocity and curvature, the WMR's planned position in the same world reference frame can be expressed as a function of $\tilde{s}$ as:

$$\left[X(\tilde{s}), Y(\tilde{s})\right] = \left[\frac{\sin(2\pi\tilde{s})}{\kappa}, \frac{1}{\kappa}\left(1 - \cos(2\pi\tilde{s})\right)\right] \tag{11}$$

The actual tracked trajectory, with side-slip angle β_A and curvature κ_A, will generally differs from the planned trajectory and is given by (Prado et al., 2002):

$$\left[X_\beta(\tilde{s}), Y_\beta(\tilde{s})\right] = \left[\frac{1}{\kappa_A}\left[\sin(\beta_A + 2\pi\tilde{s}) - \sin(\beta_A)\right], \frac{1}{\kappa_A}\left[\cos(\beta_A) - \cos(\beta_A + 2\pi\tilde{s})\right]\right] \tag{12}$$

κ_A and β_A can be approximated by the trajectory obtained as simulation output of a sufficiently accurate dynamic model or by experimental results. If enough simulations or tests are performed to characterize the dynamics of the WMR in stationary trajectories, it is possible to fit κ_A and β_A to functions of the planned curvature and velocity. Therefore, by substituting (11) and (12) into (10) and considering the simulation or experimental results to compute κ_A and β_A, the total spatial error of the WMR when navigating a whole stationary cycle can be calculated as a function of the planned V and κ. Although it is known that in general the planned variables V and κ do not stay constant at adjacent postures, it is understood that they will experience smooth variations when smooth paths are planned. Therefore, the error at each posture will be approximated by the error in (10) computed as described in this section, i.e. considering neighbourhoods where the V and κ are kept constant.

Finally, TE_s^2 will be upper limited by a magnitude relative to the total area of the circle that defines the stationary planned trajectory. Therefore, if tol_s is the percentage of the admissible tolerance for the spatial error, the following constraint is imposed:

$$TE_s^2 \le \frac{\pi}{\kappa^2}\left(1 - (tol_s)^2\right) \tag{13}$$

When (13) is applied to (10), a velocity constraint V_{Lim3} for the WMR is obtained.

3.3.2. Maximum velocity to bound temporal error

When a WMR is navigating, it must do more than consider position error; temporal error can also be important if one wishes to fit or synchronise several objects. Let the temporal error of a WMR be the time gap between the actual time when the robot arrives at a posture, t_A, and the time when it is planned to arrive, t:

$$E_t = t_A - t \tag{14}$$

For a stationary trajectory of length S tracked with actual velocity V_A, this error is:

$$E_t = S\left(\frac{1}{V_A} - \frac{1}{V}\right) \tag{15}$$

V_A in a stationary trajectory can be approximated by the velocity obtained in experimental tests or simulations of a sufficiently accurate dynamic model of the WMR. As stated for the spatial error, such outcomes make it possible to express V_A as a function of the characteristics of V and κ.

The velocity planner fixes an upper bound for the temporal error associated to each posture, given by a value relative to the time that the path tracker estimates the robot will spend in the stationary trajectory, with relative tolerance t_t^{rel}. Then the following inequality must be satisfied:

$$E_t \le tol_t\, t = tol_t \frac{S}{V} \tag{16}$$

By substituting (15) in (22), a new upper boundary function V_{Lim4} is generated as:

$$V_{Lim4} = (tol_t + 1)V_A \tag{17}$$

3.3.3. Tip-over limitation

Tip-over occurs when the robot's entire weight shifts to one side of the vehicle, and the other wheels are about to lose contact. Thus, the robot is at risk of tipping-over when its total weight is entirely borne by the outer wheel (Shiller, 1999). The extreme situation, depicted in Fig. 1 for positive κ, where h is the height of the centre of gravity (c.g.) of the robot and B_1 and B_2 are the lateral distances between the outer wheel and the c.g. for positive and negative κ, respectively (although generally $B_1=B_2$), yields a relation between the lateral force, F_y, and the vertical force, F_z, given by:

$$\begin{cases} F_y = F_z \dfrac{B_1}{h} & .....\text{if } \kappa \geq 0 \\ F_y = F_z \dfrac{B_2}{h} & .....\text{if } \kappa < 0 \end{cases} \tag{18}$$

By neglecting gyroscope torques, the lateral force, Fy, on flat grounds is simply the centrifugal force, while F_z is the robot's weight. Thus, if g is the gravity constant, equation (18) requires V to be lower than:

$$\begin{cases} V_{Lim5} = \sqrt{\dfrac{gB_1}{h}\dfrac{1}{\kappa}} & .....\text{if } \kappa \geq 0 \\ V_{Lim6} = \sqrt{\dfrac{gB_2}{h}\dfrac{1}{(-\kappa)}} & .....\text{if } \kappa < 0 \end{cases} \tag{19}$$

3.4. Operational constraints

The need to fit and synchronise the robot's motion with its environment, whether static or dynamic, makes operational constraints necessary.

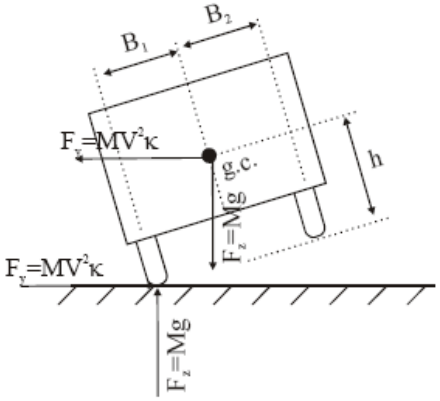


Fig. 1. Tip-over of the WMR.

3.4.1. Maximum velocity to prevent collisions

V is limited by a value that ensures the WMR will come to a complete stop at a distance greater than a safety distance from any obstacle, L_{safe}. Such a stop will be performed at the maximum deceleration, b^{max}, a constant calculated in section 4. Therefore, the distance run by the robot until it stops, is:

$$s = \frac{V^2}{2\,b^{max}} \tag{20}$$

When V_{obs} is the maximum estimated velocity of the obstacle towards the robot (0 in static environments), the distance covered by the object is:

$$s_{obs} = \frac{V_{obs}}{b^{max}} V \tag{21}$$

If D_{obs} is the distance from the robot guide point to the obstacle in completely known environments, or the radius of vision of the external sensor system in partially known environments (Krishna et al., 2006), in order to ensure that the robot maintains its safety distance, it must satisfy:

$$s + s_{obs} = D_{obs} - L_{safe} \tag{22}$$

By replacing (20) and (21) in (22), a new upper limit for the velocity is found as:

$$V_{Lim7} = \sqrt{(V_{obs})^2 + 2b^{max}(D_{obs} - L_{safe})} - V_{obs} \tag{23}$$

3.4.2. Maximum velocity to approach the target posture

In the same way, in order to ensure safe stopping at the target point of the path, another upper boundary function is given by:

$$V_{Lim8} = \sqrt{2\,C_2(S-s)\,b^{max}} \tag{24}$$

Where C_2 is an arbitrary constant greater than 1, which reflects a security percentage for stopping, and S is the total path length.

3.4.3. Environmental constraints

A set of velocity constraints which are solely dependent on the robot's working environment can be defined as a function which assigns a maximum speed V_i to each portion of the path, with expressions such as:

$$V_{Lim\,9} = \begin{cases} V_1 & \text{if } 0 \le s \le s_1 \\ \vdots & \vdots \\ V_n & \text{if } s_n s \le s_{n+1} \end{cases} \tag{25}$$

4. Acceleration and deceleration constraints

The same constructive, kinematic, dynamic and environmental topics which were analysed for velocity are studied for acceleration and deceleration in this section. From all the constraints detailed in next subsections an upper boundary function of acceleration, a_{Lim}, and deceleration, d_{Lim}, can be generated as:

$$a_{Lim}(s) = \min\{a_{Liml}, a_{Lim2} \ldots\ldots a_{Limn}\} \quad / s \subset [0, S] \tag{26}$$

$$d_{Lim}(s) = \min\{d_{Liml}, d_{Lim2} \ldots\ldots d_{Limn}\} \quad / s \subset [0, S] \tag{27}$$

4.1. Constructive constraints

The maximum torques of tractive motors, T_{tm}, steering motors, T_{sm}, and the braking mechanism, T_{bm}, dictate the maximum torques achievable at the wheels. If η_t and η_s are the efficiency of the drive-train and of the steering-train, these values are given by:

$$T_{tw}\,|^{mot} = \frac{\eta_t}{\xi_t} T_{tm}; \qquad T_{sw}\,|^{mot} = \frac{\eta_s}{\xi_s}\frac{T_{sm}}{2}; \qquad T_{bw}\,|^{mot} = \frac{\eta_t}{\xi_t} T_{bm}\Big|^{max} \tag{28}$$

4.2. Kinematic constraints

As occurs with velocity, the robot's kinematics would make its acceleration be a function of the acceleration of the wheels. But as is argued in section 4.4, this value is limited exclusively by the dynamic capabilities relative to the resisting loads.

4.3. Dynamic constraints

4.3.1. Wheel-ground adhesion constraint

In order to avoid slippage, the maximum effort that the wheel-ground contact can support in a direction j is limited by the wheel-ground friction coefficient,μ, as:

$$F_{wi}^{j}\Big|^{ad} = \mu_j F_{wi}^{n} \tag{29}$$

μ can be assumed to be constant for uniform rubber wheels if slippage does not occur and terrain characteristics are uniform (Wong, 2001). $F_{wi}{}^{n}$ is the vertical load borne by the i-th wheel, which changes with: ground irregularities, transients for non-stationary manoeuvres, lateral load transference produced by the centrifugal force on curved paths and longitudinal load transference on accelerated or decelerated paths. The first two phenomena can be neglected, especially for navigation in industrial environments. Regarding the two dynamic load transfers, $F_{wi}{}^{n}$ can be computed as a function of the static weight borne by the i-th wheel, $F_{wi}^{n}\big|_0$, as:

$$F_{wi}^{n} = F_{wi}^{n}\,|_0 + \frac{h}{\vec{L}_{wi}} Ma + \frac{h}{\vec{B}_{wi}} MV^2\vec{\kappa} \tag{30}$$

Where $\vec{L}_{wi}$ is the vector of longitudinal position of the centre of the i-th wheel with respect to the c.g. of the WMR and $\vec{B}_{wi}$ is the vector of its lateral position.

The maximum lateral effort that can be borne by the steering wheel is computed by replacing in (29) the sum of (30) extended for those wheels:

$$F_y\Big|^{ad} = \mu_y \sum_{\substack{\text{i= wheel}\\ \text{steering}}} F_{wi}^{n} \tag{31}$$

Therefore, if d_c is the castor distance, i.e. the longitudinal distance from the rotation axis of the steering system to the wheel centre, the available steering torque limited by the wheel-ground adherence capacity is:

$$T_s\Big|^{ad} = \mu_y d_c \sum_{\substack{i=\text{steering}\\ \text{wheel}}} F_{wi}^n \tag{32}$$

Regarding driven and braking wheels, by replacing in (29) the sum of (30) for all those wheels, the longitudinal driven and braking efforts limited by the wheel-ground adherence capacity are given by the following equations, respectively:

$$F_t\Big|^{ad} = \mu_x \sum_{\substack{i=\text{driven}\\ \text{wheel}}} F_{wi}^n \; ; \qquad F_b\Big|^{ad} = \mu_x \sum_{\substack{i=\text{braking}\\ \text{wheel}}} F_{wi}^n \tag{33}$$

4.3.2. Maximum tractive and braking efforts

The maximum acceleration is that which is reached applying the maximum available effort when rolling, grade and aerodynamic resistances are overcome. For WMRs, aerodynamic resistance is habitually neglected, because of their low navigation velocity. Therefore, the maximum acceleration when negotiating a grade j, and with rolling resistance coefficient f_r, is:

$$a^{max} = g\left(\frac{F_{long}\Big|^{max}}{Mg} - f_r\cos[a\tan(j/100)] - \sin[a\tan(j/100)]\right) \tag{34}$$

$F_{long}|^{max}$ is the maximum longitudinal effort, limited either by the motors or by the wheel-ground adherence capacity. Therefore, by introducing the power capacities computed in (28) and the maximum adhesion of (33) into (34), the following upper boundary functions are defined:

$$a_{Lim1} = \frac{1}{M}\cdot\min\left\{2\frac{T_{wt}\Big|^{mot}}{R},\quad F_t\Big|^{ad}\right\} - g\left(f_r\cos[a\tan(j/100)] + \sin[a\tan(j/100)]\right) \tag{35}$$

$$d_{Lim1} = \frac{1}{M}\cdot\min\left\{2\frac{T_{bw}\Big|^{mot}}{R},\quad F_b\Big|^{ad}\right\} - g\left(f_r\cos[a\tan(j/100)] + \sin[a\tan(j/100)]\right) \tag{36}$$

These are constant functions as long as f_r can be considered constant, which occurs when the operating variables stay within a small interval, which is the most common situation for WMR motion in industrial environments (Wong, 2001).

4.3.3. Maximum steering efforts

The maximum acceleration available for the steering angle, δ, can be calculated as:

$$\left|\frac{d^2\delta}{dt^2}\right|^{max} = \frac{2\,T_{steer}\Big|^{max} - \sum_{\substack{i=\text{steering}\\ \text{wheel}}} T_{wi}^{res}}{I_s} \tag{37}$$

$T_{steer}|^{max}$ is the maximum steering torque at the wheel, limited either by power of the steering system, in (28) or by adhesion, in (32); I_s is the mass moment of inertia of the whole steering system; and T_{wi}^{res} is the self-aligning torques of the i-th wheel.
Looking at the left-hand side of (37), it can be expressed by:

$$\frac{d^2\delta}{dt^2} = \frac{d^2s}{d^2t}\frac{d^2\kappa}{ds^2}\frac{d^2\delta}{d\kappa^2} \tag{38}$$

Thus, the acceleration of δ depends on three terms: the acceleration of the trajectory; the spatial acceleration of curvature, a characteristic directly derived from the spatial path; and

on the last term, which is a characteristic of the robot that can be approximated from its kinematic model.
By replacing (38) and in (37) and by isolating the acceleration, a new boundary function is given by:

$$a_{Lim2} = \frac{\min\left\{2T_w^s\left|^{mot}, T^s\right|^{ad}\right\} - \frac{MV^2|\kappa|}{2}d_c}{I_s}\frac{1}{\left|d^2\delta/d\kappa^2\right|}\frac{1}{\left|d^2\kappa/ds^2\right|} \quad (39)$$

This upper bound depends on the velocity, which would take the VP to an iteration loop. In order to avoid this costly procedure, V is substituted by the single-valued upper boundary function of velocity, $V_{LIM}(s)$, defined in section 3, making the acceleration boundary even more restrictive.

5. Velocity Planning in Static Environments

5.1. Path Segmentation

In order to reduce its complexity, the proposed method divides the path into segments where all of the velocity constraints are continuous and monotone or constant, i.e. where the velocity of the path can be continuous and monotone or constant when it matches its upper boundary function. The velocity profile for the entire path is generated sequentially from the first segment to the last one according to the method describe later on in this section.
A set of p+2 segmentation points, $P_s=\{0, {}^1ss, \ldots {}^pss, S\}$, sorted in increasing order, divides P(s) into p+1 segments, $SG=\{{}^1Sg, \ldots {}^{p+1}Sg\}$, where ${}^iSg$ is the subset of P(s) corresponding to the domain $s\subset[{}^{i-1}ss, {}^iss]$.
P_s, comprises: the initial point of the path, s=0; its final point, s=S; and every point that satisfies at least one of the following conditions:

- To be a local extremum of the velocity boundary function: consequently V_{Lim} is monotone over ${}^iSg$, so that V(s) can also be monotone when it matches V_{Lim}.
- To be a point of discontinuity of V_{Lim}: at the first stage, the VP algorithm proposed herein will only deal with the velocity limits at the end points of the segment. Later it will check if $V_{Lim}(s)$ is exceeded at any intermediate point and, in such a case, a time-consuming iterative process will be carried out. Since any discontinuity of V_{Lim} increases the risk of failing to meet the velocity constraint, they are shifted to the ends of the segments by selecting them as segmentation points.

5.2. VP of a segment of the path in static environments

The piece of V(s) for ${}^iSg$ is generated from the time-space (t×s) function ${}^i\sigma(t)$, which computes the path length navigated as:

$$s = {}^i\sigma(t) \quad ; s \subset \left[{}^{i-1}ss,\ {}^iss\right] \quad (40)$$

Thus, the velocity profile for the segment is:

$${}^iV(s) = \frac{d}{dt}\left({}^i\sigma\left({}^i\sigma^{-1}(s)\right)\right) \quad ; s \subset \left[{}^{i-1}ss,\ {}^iss\right] \quad (41)$$

${}^i\sigma(t)$ must start at the initial position of ${}^iSg$ and arrive at its final position. If the origin of time is shifted to the first point of ${}^iSg$, without loss of generality, and if ${}^it$ is the time taken by the WMR to navigate ${}^iSg$, the position boundary conditions are:

$$\begin{aligned} {}^{i}\sigma(0) &= {}^{i-1}ss \\ {}^{i}\sigma({}^{i}t) &= {}^{i}ss \end{aligned} \tag{42}$$

The first derivative of ${}^{i}\sigma(t)$ with respect to time, ${}^{i}\sigma'(t)$, must also satisfy the velocity boundary conditions, which are given at the edge points of the segment by:

$$\begin{aligned} {}^{i}\sigma'(0) &= {}^{i-1}v = \begin{cases} 0 & \text{if } i = 1 \\ {}^{i-1}\sigma'({}^{i-1}t) & \text{if } i \neq 1 \end{cases} \\ {}^{i}\sigma'({}^{i}t) &= {}^{i}v = V_{Lim}({}^{i}s) \end{aligned} \tag{43}$$

The second equation of (43) sets the velocity at the end of ${}^{i}Sg$ to the maximum value permitted by the V_{Lim} at this point, in order to obtain the fastest trajectory; while the first equation of (43) compels the WMR to start from a standstill for the first segment of the path, i=1, or ensures continuity between adjacent segments for any other case. Note that ${}^{i-1}\sigma'({}^{i-1}t)$ was set to its highest feasible value when planning the previous segment ${}^{i-1}Sg$.

The cubic polynomial is selected to generate ${}^{i}\sigma(t)$, since it has just enough parameters to satisfy the boundary conditions in (42) and (43) and it has inverse, so that (41) can be computed. Thus, the t×s function of the path can be expressed as:

$${}^{i}\sigma(t) = {}^{i}\sigma_0 t^3 + {}^{i}\sigma_1 t^2 + {}^{i}\sigma_2 t + {}^{i}\sigma_3 \tag{44}$$

By applying (42) and (43) to (44), the boundary conditions can be summed up as:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ ({}^{i}t)^3 & ({}^{i}t)^2 & {}^{i}t & 1 \\ 0 & 0 & 1 & 0 \\ 3({}^{i}t)^2 & 2\,{}^{i}t & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{i}\sigma_0 \\ {}^{i}\sigma_1 \\ {}^{i}\sigma_2 \\ {}^{i}\sigma_3 \end{bmatrix} = \begin{bmatrix} {}^{i-1}ss \\ {}^{i}ss \\ {}^{i-1}v \\ {}^{i}v \end{bmatrix} \tag{45}$$

${}^{i}t$ must be computed to confine ${}^{i}V(s)$ into the safety zone of the s×v plane limited by $[0 \le V(s), V(s) \le V_{Lim}(s)]$, and its first derivative into the safety zone of the s×a plane limited by $[-d_{Lim}(s) \le V'(s), V'(s) \le a_{Lim}(s)]$; further, ${}^{i}t$ must be computed to ensure the continuity of ${}^{i}\sigma(t)$ up to its second derivative for the entire domain of the path, specifically between adjacent segments.

The magnitude ${}^{i}\sigma''(t)$ is important because local extrema of V_{Lim} are always located at the ends of the segments that partition the path. The velocity planned for these points is the maximum possible, under (43). If a maneuver with positive acceleration is planned for a segment whose end point is a local maximum of V_{Lim}, the velocity boundary will be violated at the beginning of the next segment. A similar situation would occur for a local minimum when negative acceleration is planned.

A direct approach to the problem would involve solving a large system of non-linear inequalities, a process of very high computational cost (Muñoz, 1995). Therefore, a method is proposed, working mainly with closed mathematical expressions and thereby reducing computational cost significantly. The method is based on setting the acceleration to zero at the segmentation points, so that acceleration continuity is ensured and the problem of failing to meet the velocity constraints just after local extrema of V_{Lim}, explained in a previous paragraph, is overcome.

The maximum acceleration of ${}^{i}Sg$, ${}^{i}A$, is planned to be reached at an intermediate point, ${}^{i}t_c$, which yields one of the acceleration profiles showed in Fig. 2, therefore:

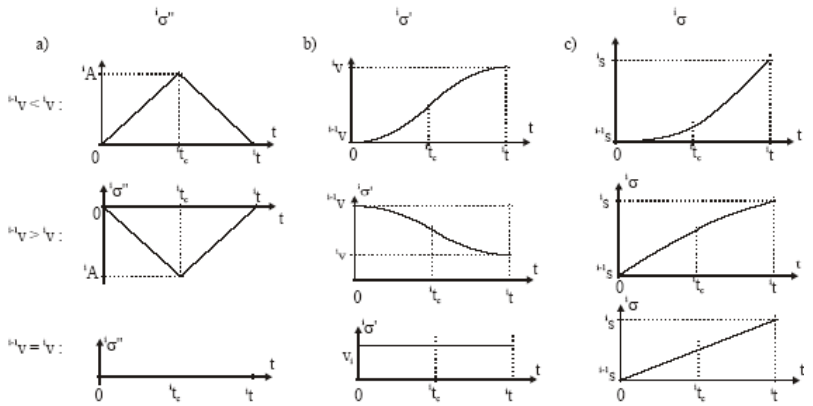


Fig. 2. Possible solutions in static environments. A) Acceleration profile; b)Velocity profile; c)Trajectory.

$$ {}^{i}\sigma''(t) = {}^{i}\sigma''_1(t) \cap {}^{i}\sigma''_2(t); \quad \begin{cases} {}^{i}\sigma''_1(t) = \dfrac{{}^{i}A}{{}^{i}t_c}\, t & .....t \le {}^{i}t_c \\ {}^{i}\sigma''_2(t) = \dfrac{{}^{i}A}{{}^{i}t - {}^{i}t_c}\left({}^{i}t - t\right) & .....t > {}^{i}t_c \end{cases} \tag{46} $$

By integrating (46) and calculating the integration constants to satisfy the initial velocity boundary condition of (43) and to ensure continuity between $^{i}\sigma'_1(t)$ and $^{i}\sigma'_2(t)$, the velocity profile of the segment, also plotted in Fig. 2, is:

$$ {}^{i}\sigma'(t) = {}^{i}\sigma'_1(t) \cap {}^{i}\sigma'_2(t); \quad \begin{cases} {}^{i}\sigma'_1(t) = {}^{i-1}v + \dfrac{{}^{i}A}{2\,{}^{i}t_c}\, t^2 & .....t \le {}^{i}t_c \\ {}^{i}\sigma'_2(t) = {}^{i-1}v + \dfrac{{}^{i}A}{{}^{i}t - {}^{i}t_c}\left({}^{i}t\, t - \dfrac{t^2}{2} - \dfrac{{}^{i}t_c\, {}^{i}t}{2}\right) & .....t > {}^{i}t_c \end{cases} \tag{47} $$

Finally $^{i}\sigma(t)$ is computed by integrating (47) with the integration constants to satisfy the first position boundary condition of (42) and to ensure its continuity at $^{i}t_c$:

$$ {}^{i}\sigma(t) = {}^{i}\sigma_1(t) \cap {}^{i}\sigma_2(t); \quad \begin{cases} {}^{i}\sigma_1(t) = {}^{i-1}ss + {}^{i-1}v\, t + \dfrac{{}^{i}A}{6\,{}^{i}t_c}\, t^3 & .....t \le {}^{i}t_c \\ {}^{i}\sigma_2(t) = {}^{i-1}ss + {}^{i-1}v\, t - {}^{i}A\dfrac{t^3 - 3\,{}^{i}t\, t^2 + 3\,{}^{i}t\,{}^{i}t_c\, t - {}^{i}t\left({}^{i}t_c\right)^2}{6\left({}^{i}t - {}^{i}t_c\right)} & .....t > {}^{i}t_c \end{cases} \tag{48} $$

The proposed algorithm initially selects $^{i}t_c$ at the half-way point of ^{i}Sg. So:

$$ {}^{i}t_c = \frac{{}^{i}t}{2} \tag{49} $$

In this case, by taking into account the second velocity condition in (43) it is found that:

$$ {}^{i}t = 2\,\frac{{}^{i}v - {}^{i-1}v}{{}^{i}A} $$

In order to arrive at the position given by the second equation of (50), ^{i}A must be:

$$ {}^{i}A = 4\,\frac{{}^{i}ss - {}^{i-1}ss}{\left({}^{i}t\right)^2} - 4\,\frac{{}^{i-1}v}{{}^{i}t} \tag{51} $$

Therefore, by replacing (51) in (50), the value of ${}^{i}t$ needed for navigating ${}^{i}Sg$ with the selected velocity profile is computed as:

$$ {}^{i}t = 2\,\frac{{}^{i}ss - {}^{i-1}ss}{{}^{i}v + {}^{i-1}v} \tag{52} $$

While the maximum acceleration must be:

$$ {}^{i}A = \frac{\left({}^{i}v\right)^2 - \left({}^{i-1}v\right)^2}{{}^{i}ss - {}^{i-1}ss} \tag{53} $$

${}^{i}\sigma(t)$ is fully defined by replacing (49), (52) and (53) in (48). It satisfies the boundary conditions of position and velocity at the initial and end points of ${}^{i}Sg$ and ensures the continuity up to its second derivative over both ${}^{i}Sg$ and the entire path domain. But $a_{Lim}(s)$ and $d_{Lim}(s)$ constraints must also be satisfied, hence it is necessary to check that:

$$ \left.\begin{array}{l} \dfrac{d}{dt}\,{}^{i}V(s) \le a_{Lim}(s) \\ -\dfrac{d}{dt}\,V(s) \ge d_{Lim}(s) \end{array}\right\} \quad \forall s \subset \left[{}^{i-1}ss,\ {}^{i}ss\right] \tag{54} $$

When ${}^{i-1}v<{}^{i}v$, an acceleration maneuver is planned for the segment. So if the acceleration restriction fails, it occurs because its upper boundary, the first inequality of (54), is violated. This problem can be solved by decreasing the final velocity to the maximum permitted by the maximum feasible acceleration in the segment. Thus, the velocity at the end of the segment is modified to be:

$$ {}^{i}v^* = \sqrt{\left({}^{i-1}v\right)^2 + {}^{i}A^*\left({}^{i}ss - {}^{i-1}ss\right)} \tag{55} $$

where the modified acceleration, ${}^{i}A^*$, is computed as:

$$ {}^{i}A^* = \min\left\{a_{Lim}(s) / s \subset \left[{}^{i-1}ss,\ {}^{i}ss\right]\right\} \tag{56} $$

And the time that the WMR takes to navigate the i-th segment is recomputed as:

$$ {}^{i}t^* = 2\,\frac{{}^{i}v^* - {}^{i-1}v}{{}^{i}A^*} \tag{57} $$

Obviously, ${}^{i}v^*<{}^{i}v$, since ${}^{i}A^*<{}^{i}A$, and it is not necessary to check the velocity constraint at the end point of ${}^{i}Sg$ again.

On the other hand, when ${}^{i-1}v<{}^{i}v$, a deceleration maneuver is planned for the segment. Therefore, if the acceleration restriction fails, it occurs because its lower boundary, the second inequality of (54), is violated. In this case it is not possible to reduce deceleration by increasing the final velocity, since ${}^{i}v$ was selected as the maximum permitted by V_{Lim}. As a consequence, it becomes necessary to decrease the initial velocity to the maximum permitted by the maximum feasible deceleration:

$$ {}^{i-1}v^* = \sqrt{\left({}^{i}v\right)^2 - {}^{i}A^*\left({}^{i}ss - {}^{i-1}ss\right)} \tag{58} $$

where ${}^{i}A^*$ is computed as:

$$ {}^{i}A^* = -{}^{i}d^* = -\min\left\{d_{Lim}(s) / s \subset \left[{}^{i-1}ss,\ {}^{i}ss\right]\right\} \tag{59} $$

And the new time it takes to navigate the i-th segment is recomputed by:

$$ {}^{i}t^{*} = 2\,\frac{{}^{i-1}v - {}^{i}v^{*}}{{}^{i}d^{*}} \tag{60} $$

This strategy requires the velocity profile of ${}^{i-1}Sg$ to also be modified by planning ${}^{i-1}v^{*}$ as the velocity at its final posture, in order to ensure the continuity of V(s).
The proposed algorithm ensures the velocity constraint will be satisfied at the end points of ${}^{i}Sg$, but not along the entire segment. Thus, it must be checked:

$$ {}^{i}V(s) \le V_{Lim}(s) \quad \forall s \subset \left[{}^{i-1}ss,\ {}^{i}ss\right] \tag{61} $$

If this constraint is violated now, the iterative processes detailed in section 5.3 must be performed. When ${}^{i}V(s)$ was computed by the proposed algorithm, inequality (61) failed in very few cases for the tests carried out with the WMR RAM, mentioned in section 7, and therefore the iterative strategies were necessary very infrequently. The same results can be expected for any WMR that works with spatial paths planned as smooth curves (continuous in curvature), because V_{Lim} depends on the curvature of the path and its first derivative with respect to the path length, and these functions are smooth if the spatial path is smooth.
The last task to build ${}^{i}\sigma(t)$ involves undoing the time shifting, i.e., setting the time at the initial point of ${}^{i}Sg$ equal to the time at the final point of ${}^{i-1}Sg$. That is:

$$ \left.\begin{array}{ll} \text{for} \quad i = 1 & {}^{i}\sigma(t) = {}^{i}\sigma(t) \\ \text{for} \quad i = 2 \text{ to } p+1 & {}^{i}\sigma(t + {}^{i-1}t) = {}^{i}\sigma(t) \end{array}\right\} \quad t \subset \left[0,\ {}^{i}t\right] \tag{62} $$

5.3. Velocity profile modification to satisfy velocity constraints in segments.

5.3.1. Acceleration maneuvers

When (61) is not satisfied in ${}^{i}Sg$ and ${}^{i-1}v<{}^{i}v$, ${}^{i}V(s)$ is iteratively slowed down by delaying ${}^{i}t_c$ from the half-way point of ${}^{i}Sg$ to a value ${}^{i}t_c^{*}$, given by:

$$ {}^{i}t_c^{*} = \frac{N-1}{N}\,{}^{i}t \quad ; N = 3, 4, 5 \ldots\ldots \tag{63} $$

Then ${}^{i}\sigma(t)$ is recomputed under (48) but substituting ${}^{i}t_c$ with ${}^{i}t_c^{*}$ and with the maximum acceleration recomputed to be a value that satisfies the conditions in (42) and (43):

$$ {}^{i}A^{*} = 2\,\frac{2\left({}^{i}v\right)^{2} - {}^{i-1}v\ {}^{i}v - \left({}^{i-1}v\right)^{2}}{3\left({}^{i}ss - {}^{i-1}ss\right) + {}^{i}t_c^{*}\left({}^{i}v - {}^{i-1}v\right)} \tag{64} $$

And ${}^{i}t$ is substituted with ${}^{i}t^{*}$ computed under (57). N in (63) keeps on increasing and ${}^{i}\sigma(t)$ being modified until (61) is satisfied.
By deriving (64) with respect to ${}^{i}t_c^{*}$:

$$ \frac{d}{d\ {}^{i}t_c^{*}}\,{}^{i}A^{*} = -\,\frac{{}^{i}A^{*}}{3\left({}^{i}ss - {}^{i-1}ss\right) + {}^{i}t_c^{*}\left({}^{i}v - {}^{i-1}v\right)}\left({}^{i}v - {}^{i-1}v\right) \tag{65} $$

It is observed that for acceleration maneuvers the derivative is always negative. Thus, delaying ${}^{i}t_c^{*}$, i.e. increasing ${}^{i}t_c^{*}$ in (73), implies reducing the maximum acceleration needed to satisfy the boundary conditions of the segment, although the time consumed in navigating it is increased. Hence a long trajectory with low velocity at all points is planned, and the velocity constraint is satisfied. Obviously, if the acceleration constraint was satisfied before the modification, it is also verified now and does not need to be checked again.

5.3.2. Deceleration maneuvers

When (61) is not satisfied in ${}^{i}Sg$ and ${}^{i-1}v>{}^{i}v$, a similar strategy to the one described in the previous section for acceleration maneuvers is applicable, but in this case, in order to plan a maneuver of lower deceleration, ${}^{i}t_c$ is advanced to a value ${}^{i}t_c{}^{*}$, which is given by:

$$ {}^{i}t_c{}^{*} = \frac{1}{N}\,{}^{i}t \quad ; N = 3, 4, 5 ..... \tag{66} $$

Then ${}^{i}\sigma(t)$ is recomputed by (48), substituting ${}^{i}t_c$ with ${}^{i}t_c{}^{*}$, maximum deceleration recomputed by (64) to be a value that satisfies the conditions in (50) and (51), and substituting ${}^{i}t$ with ${}^{i}t^{*}$ computed under (57). N in (66) keeps on increasing and ${}^{i}\sigma(t)$ being modified until (61) is satisfied.

It can be observed that the derivative of the modified deceleration, ${}^{i}d^{*}=-{}^{i}A^{*}$, given by (65), is always positive when ${}^{i-1}v>{}^{i}v$. Therefore, advancing the control point implies reducing the maximum deceleration and therefore increasing the time consumed in tracking the trajectory. Consequently if the deceleration constraint was satisfied before the modification, it is also verified now and does not need to be checked again. This modification leads to planning lower velocities at all points of the segment, except for points that are very close to the end point. Only when the velocity constraint fails in this region but not at the end point is the proposed method unable to find a feasible solution. But such a situation is not expected to occur when the spatial path is planned as smooth curves and the segmentation of section 5.1 is applied.

5.4. Velocity planning algorithm in static environments

The comprehensive algorithm proposed in this chapter for VP in static environments is summarized in the following flowchart:

1. *Compute* $V_{Lim}(s)\,;\quad s \subset [0, S]$
2. *Compute* $a_{Lim}(s)\,;\quad s \subset [0, S]$
3. *Compute* $d_{Lim}(s)\,;\quad s \subset [0, S]$
4. *Create Ps={0, 1ss, ... pss, S}*
5. *VP for iSg.* ***for*** *i=1 to p+1* ***do***
6. *Set position boundary conditions, (42)*
7. *Set velocity boundary conditions, (43)*
8. *Compute iσ(t), (48), with it, (52), itc, (49), and iA, (53)*
9. ***if*** *acceleration constraint, (54), fails* ***then***
10. ***if*** ${}^{i-1}v<{}^{i}v$ ***then***
11. *Modified iσ(t) with iv*, (55), iA*, (56), and it*, (57)*
12. ***end if***
13. ***if*** *i-1v>iv* ***then***
14. *Modified iσ(t) with i-1v*, (58),, iA*, (59), and it*, (60)*
15. *Recompute* ${}^{i-1}V(s)$ *for* ${}^{i-1}Sg$ *with new* ${}^{i}v^{*}$ *, (58)*
16. ***end if***
17. ***end if***
18. ***if*** *velocity constraint, (61), fails* ***then***
19. ***if*** ${}^{i-1}v<{}^{i}v$ ***then***
20. *N=1*
21. ***repeat***
22. *N=N+2.*
23. *Advance* ${}^{i}t_c$*, (63)*
24. *Compute* ${}^{i}A^{*}$*, (64) and* ${}^{i}t^{*}$*, (57)*
25. *Compute* ${}^{i}\sigma(t)$*, (48), with values in* ***Error! Reference source not found.***
26. ***until*** *satisfy velocity constraint, (61)*
27. ***end if***
28. ***if*** ${}^{i-1}v>{}^{i}v$ ***then***
29. *N=1*
30. ***repeat***
31. *N=N+2*
32. *Delay* ${}^{i}t_c$*, (66)*
33. *Compute* ${}^{i}A^{*}$*, (64), and* ${}^{i}t^{*}$*, (57)*
34. *Compute* ${}^{i}\sigma(t)$*, (48), with values in* ***(Error! Reference source not found.)***
35. *until satisfy velocity constraint (61)*
36. ***end if***
37. ***end if***
38. *Shift the origin of the time, (62)*
39. *Compute* ${}^{i}V(s)$*, (41)*
40. ***end for***

6. Velocity Planning in Dynamic Environments

Crossing points of moving obstacles with the WMR's path can be represented by a set R=[^{1}R ^{2}R … ^{l}R] of rectangular forbidden regions in the s×t plane (Liu & Wu, 2001). Each region ^{k}R is defined by a segment of the space with end points [$^{k}s_{ini}$, $^{k}s_{fin}$] and an interval of time limited by [$^{k}t_{ini}$, $^{k}t_{fin}$], as shown in Fig. 3.
The function $^{i}\sigma(t)$, computed previously, is not a valid solution of the VP problem if it intersects a forbidden region.

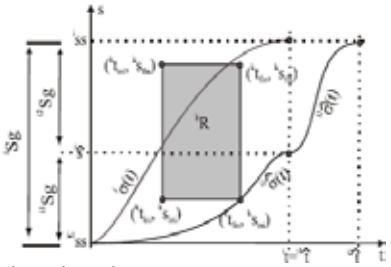


Fig. 3. Velocity planning in dynamic environments.

The problem can be solved by planning a slower trajectory that allows the moving obstacle to cross the robot's path before it arrives at the crossing region. The opposite solution, to plan a faster trajectory so that the robot passes through the dangerous space before the obstacle arrives, it is not possible, since $^{i}\sigma(t)$ was built to achieve the highest feasible velocity profile.
A secondary aim of the modification strategy, as stated in section 2, is to achieve a low computational cost. This is accomplished by both modifying only the segments adjacent to where the obstacle is found and by avoiding iterative strategies.
If $^{i}\sigma(t)$ intersects the forbidden region ^{k}R, the trajectory is slowed down by dividing iSg into two subsegments: $^{i}Sg = {}^{i1}Sg \cup {}^{i2}Sg$. The first one, ^{i1}Sg, plans a velocity profile, , that makes the WMR avoid the obstacle. The second segment, ^{i2}Sg, plans a velocity profile, $^{i2}\hat{V}(t)$, that makes the WMR arrive at the final position imposed by the second position boundary condition of (42) with the velocity imposed by the second velocity boundary condition of (43).

6.1. Planning the velocity profile to avoid moving obstacles

The first piece of the modified sxt function, $^{i1}\hat{\sigma}(t)$, is planned to make the WMR avoid ^{k}R by compelling it to pass through its first point at its last time, i.e.:

$$^{i1}\hat{\sigma}\left(^{k}t_{fin}\right) = {}^{k}s_{ini} \tag{67}$$

The total time for the subsegment ^{i1}Sg, $^{i1}\hat{t}$, is set equal to ^{i}t, the time planned for ^{i}Sg in static environments, and its point of maximum acceleration, $^{i1}\hat{t}_{c}$, to the same time as for static environments, $^{i}t_{c}$. This goal can be achieved by selecting the maximum acceleration in ^{i1}Sg so that $^{i1}\hat{\sigma}(t)$ is similar to $^{i}\sigma(t)$ computed by (48) but substituting ^{i}A with a lower

acceleration, ${}^{i1}\hat{A}$, calculated to satisfy (67). This value is given by:

$$\text{if } {}^{k}t_{fin} \le {}^{i1}\hat{t}_c \ldots\ldots \quad {}^{i1}\hat{A} = 6\left({}^{k}s_{ini} - {}^{i-1}ss - {}^{i-1}v\ {}^{k}t_{fin}\right) \tag{68}$$

$$\text{if } {}^{k}t_{fin} > {}^{i1}\hat{t}_c \ldots\ldots \quad {}^{i1}\hat{A} = 6\left({}^{k}s_{ini} - {}^{i-1}ss - {}^{i-1}v\ {}^{k}t_{fin}\right)\frac{{}^{i1}\hat{t} - {}^{i1}\hat{t}_c}{3\ {}^{i1}\hat{t}\left({}^{k}t_{fin}\right)^2 - \left({}^{k}t_{fin}\right)^3 - 3\ {}^{i1}\hat{t}_c\ {}^{i1}\hat{t}\ {}^{k}t_{fin} + \left({}^{i1}\hat{t}_c\right)^2\ {}^{i1}\hat{t}}$$

The WMR's position at the end of ${}^{i1}Sg$, ${}^{i}\hat{s}$, is a lower value than is, since a lower acceleration is used during the same time. Therefore the WMR does not arrive at the final position condition, ${}^{i}ss$, given by the second equation of (42), but rather at:

$${}^{i}\hat{s} = {}^{i-1}ss + {}^{i-1}v\ {}^{i1}\hat{t} + \frac{{}^{i1}\hat{A}}{6}\frac{\left({}^{i1}\hat{t}\right)^3}{{}^{i1}\hat{t}_c} \tag{69}$$

Likewise, the velocity of the WMR at the final posture of ${}^{i1}Sg$, ${}^{i,i1}\hat{v}(t)$ is also a lower value than ${}^{i}v$, so it does not satisfy the final velocity condition of (43), but is rather:

$${}^{i,i1}\hat{v} = {}^{i-1}v + \frac{{}^{i1}\hat{A}}{2}\frac{\left({}^{i1}\hat{t}\right)^2}{{}^{i1}\hat{t}_c} \tag{70}$$

Since ${}^{i1}\hat{\sigma}(t)$ is computed from ${}^{i}\sigma(t)$ by applying a lower acceleration for the same time and the velocity and acceleration constraints were successfully checked for ${}^{i}\sigma(t)$, the two upper limits are also satisfied by ${}^{i1}\hat{\sigma}(t)$. But two lower limits can be violated and they must be checked: the positive magnitude of the velocity and the deceleration constraint:

$${}^{i1}\hat{\sigma}'(t) \ge 0 \quad \forall t \subset \left[0,\ {}^{i1}\hat{t}\right] \tag{71}$$

$${}^{i1}\hat{A} \ge -\max\left\{d_{Lim}(s) \quad / s \subset \left[{}^{i-1}ss,\ {}^{i}\hat{s}\right]\right\} \tag{72}$$

If it ${}^{i1}\hat{\sigma}(t)$ fails to meet one of these constraints, the processes detailed in subsections 6.1.1 or 6.1.2 must be carried out, respectively.

The last task to build ${}^{i1}\hat{\sigma}(t)$ involves setting the time at the initial point of the segment equal to the time at the final point of the previous segment:

$${}^{i1}\hat{\sigma}\left(t + {}^{i-1}t\right) = {}^{i1}\hat{\sigma}(t) \tag{73}$$

Finally, the velocity profile ${}^{i1}\hat{V}(s)$ is computed by applying (41).

6.1.1. Modification of the velocity profile to keep velocity positive

When the lower boundary of velocity, (71), is violated, the initial velocity and the maximum deceleration of ${}^{i1}Sg$ are modified in order to plan for the WMR to arrive at the end posture at zero velocity, i.e.:

$${}^{i1}\hat{\sigma}'\left({}^{i}\hat{t}\right) = {}^{i,i1}\hat{v} * = 0 \tag{74}$$

By making ${}^{i1}\hat{\sigma}(t)$ fulfill (74) while still satisfying (67) to avoid ${}^{k}R$, the initial velocity and maximum acceleration of ${}^{i1}Sg$ are modified to be ${}^{i-1,i1}\hat{v}*(t)$ and ${}^{i1}\hat{A}*$:

$$\text{if} \quad {}^{k}t_{fin} \le {}^{i}\hat{t}_c \ldots\ldots \begin{cases} {}^{i1}\hat{A}* = -6\left({}^{k}s_{ini} - {}^{i-1}ss\right)\dfrac{{}^{i1}\hat{t}_c}{{}^{k}t_{fin}\left(3\,{}^{i1}\hat{t}\,{}^{i1}\hat{t}_c - \left({}^{k}t_c\right)^2\right)} \\ {}^{i-1,i1}\hat{v}* = 3\left({}^{k}s_{ini} - {}^{i-1}ss\right)\dfrac{{}^{i1}\hat{t}_c\,{}^{i1}\hat{t}}{{}^{k}t_{fin}\left(3\,{}^{i1}\hat{t}\,{}^{i1}\hat{t}_c - \left({}^{k}t_c\right)^2\right)} \end{cases}$$

$$\text{if} \quad {}^{k}t_{fin} > {}^{i}\hat{t}_c \ldots\ldots \begin{cases} {}^{i1}\hat{A}* = -6\left({}^{k}s_{ini} - {}^{i-1}ss\right)\dfrac{{}^{i1}\hat{t} - {}^{i1}\hat{t}_c}{3\,{}^{k}t_{fin}\left({}^{i1}\hat{t}\right)^2 + \left({}^{k}t_{fin}\right)^3 - 3\,{}^{i1}\hat{t}\left({}^{k}t_{fin}\right)^2 - \left({}^{i1}\hat{t}_c\right)^2\,{}^{i1}\hat{t}} \\ {}^{i-1,i1}\hat{v}* = 3\left({}^{k}s_{ini} - {}^{i-1}ss\right)\dfrac{\left({}^{i1}\hat{t} - {}^{i1}\hat{t}_c\right){}^{i1}\hat{t}}{3\,{}^{k}t_{fin}\left({}^{i1}\hat{t}\right)^2 + \left({}^{k}t_{fin}\right)^3 - 3\,{}^{i1}\hat{t}\left({}^{k}t_{fin}\right)^2 - \left({}^{i1}\hat{t}_c\right)^2\,{}^{i1}\hat{t}} \end{cases} \tag{75}$$

Obviously, in this case the previous segment, ${}^{i-1}Sg$, must be recomputed following the method stated in section 5 with a new velocity boundary condition, given by:

$$\begin{aligned} {}^{i1}\hat{\sigma}'(0) &= {}^{i-1}v = \begin{cases} 0 & \text{if } i = j \\ {}^{i-1}\sigma'({}^{i-1}t) & \text{if } i \neq j \end{cases} \\ {}^{i1}\hat{\sigma}'\left({}^{i1}\hat{t}\right) &= {}^{i-1,i1}\hat{v}* \end{aligned} \tag{76}$$

6.1.2 Modification of the velocity profile to satisfy the deceleration constraint

When the lower boundary of acceleration, (72), is violated, the maximum deceleration is decreased by setting it to its maximum feasible value. Thus, ${}^{i1}\hat{A}$ is replaced by:

$${}^{i1}\hat{A}* = -\max\left\{d_{Lim}(s) \quad / s \subset \left[{}^{i-1}ss,\ {}^{i}\hat{s}\right]\right\} \tag{77}$$

Additionally, the initial velocity of the subsegment must be sufficiently reduced to keep on satisfying (76) when the maximum deceleration is ${}^{i1}\hat{A}*$, yielding:

$$\text{if } {}^{k}t_{fin} \le {}^{i1}\hat{t}_c \ldots\ldots \quad {}^{i-1,i1}\hat{v}* = \frac{{}^{k}s_{fin} - {}^{i-1}ss}{{}^{k}t_{fin}} - \frac{{}^{i}\hat{A}*}{6}\,\frac{\left({}^{k}t_{fin}\right)^2}{{}^{i1}\hat{t}_c}$$

$$\text{if } {}^{k}t_{fin} > {}^{i1}\hat{t}_c \ldots\ldots \quad {}^{i-1,i1}\hat{v}* = \frac{{}^{k}s_{fin} - {}^{i-1}ss}{{}^{k}t_{fin}} - \frac{{}^{i}\hat{A}*}{6}\,\frac{3\,{}^{i1}\hat{t}\left({}^{k}t_{fin}\right)^2 - \left({}^{k}t_{fin}\right)^3 - 3\,{}^{i1}\hat{t}_c\,{}^{i1}\hat{t}\,{}^{k}t_{fin} + \left({}^{i1}\hat{t}_c\right)^2\,{}^{i1}\hat{t}}{\left({}^{i1}\hat{t} - {}^{i1}\hat{t}_c\right){}^{k}t_{fin}} \tag{78}$$

Thus, ${}^{i1}\hat{\sigma}(t)$ is recomputed from ${}^{i}\sigma(t)$ by replacing ${}^{i}A$ with ${}^{i1}\hat{A}*$ and ${}^{i-1}v$ with ${}^{i-1,i}\hat{v}*$. The position and velocity of the WMR at the end of ${}^{i1}Sg$ must be also recomputed by using (78) and (79) respectively, with the new values for ${}^{i1}\hat{A}*$ and ${}^{i-1,i1}\hat{v}*$.
Finally, it is necessary to recompute the previous segment, ${}^{i-1}Sg$, making its final velocity equal to ${}^{i-1,i1}\hat{v}*$ to avoid discontinuities between adjacent segments.

6.2. Planning the velocity profile to arrive at the final position

A second function ${}^{i2}\hat{\sigma}(t)$ is planned to be attached to in order to make the WMR arrive at ${}^{i}ss$. The following position boundary conditions must be satisfied by ${}^{i2}\hat{\sigma}(t)$:

$$
\begin{aligned}
{}^{i2}\hat{\sigma}\left({}^{i1}\hat{t}\right) &= {}^{i}\hat{s} \\
{}^{i2}\hat{\sigma}\left({}^{i2}\hat{t}\right) &= {}^{i}ss
\end{aligned}
\tag{79}
$$

Additionally, ${}^{i2}\hat{\sigma}(t)$ must ensure continuity between i1Sg and i2Sg. Therefore:

$$
\begin{aligned}
{}^{i2}\hat{\sigma}'\left({}^{i1}\hat{t}\right) &= {}^{i,i1}\hat{v} \\
{}^{i2}\hat{\sigma}'\left({}^{i2}\hat{t}\right) &= {}^{i,i2}\hat{v} = V_{Lim}\left({}^{i2}\hat{t}\right)
\end{aligned}
\tag{80}
$$

${}^{i2}\hat{\sigma}(t)$ is generated by the procedure presented in section 5, but with the boundary conditions given by (88) and (89).

6.3. Velocity planning algorithm in dynamic environments

The comprehensive algorithm for VP in dynamic environments is summarized as follows:

1. Compute , (56) with , (77)
2. **if** lower boundary of velocity,(80) ,fails **then**
3. Compute , (84)
4. Compute , (84)
5. Compute ,(56) ,with values of 3, 4
6. Replan i-1V(s) of i-1Sg with velocity boundary conditions of (85)
7. **end if**
8. **if** boundary of deceleration, (81)fails **then**
9. Compute , (86)
10. Compute , (87)
11. Compute ,(56),with values 9, 10
12. Replan i-1V(s) of i-1Sg with velocity boundary conditions of (85)
13. **end if**
14. Shift the origin of time, (82)
15. Compute iV(s), (49)
16. Compute , (78)
17. Compute , (79)
18. Vp for i2Sg:
19. Algorithm for static environments with position boundary condition of (88) and velocity boundary conditions of (89).

7. Experimental and simulation results

The velocity planner described in previous sections is applied to the robot RAM (Ollero et al., 1993), a WMR designed for navigation with high maneuverability in indoor and outdoor industrial environments. The maximum velocity its traction motors can reach is 1.7m/s and their maximum acceleration is 10.8m/s2

The spatial paths of the robot are generated as cubic spirals, since this is curve that shows the best mechanical characteristics to be tracked by a WMR (Prado, 2000): continuity of position, orientation and curvature; bounded curvature; arbitrary selection of position and orientation at the initial and final postures; best dynamic behavior of the WMR, specifically lowest tracking errors, when it navigates with open loop control.

Fig. 4 plots the planned trajectory and the trajectory actually tracked by RAM in a static environment. The time consumed for the trajectory which was planned using the proposed algorithm is 62.2s, 24.7% longer than the time that would be consumed if RAM navigated the entire path at its maximum velocity. Results show very low tracking errors: spatial errors are lower than 0.1m for a path of 84.4m, and temporal errors are lower than 1s for a trajectory of 62.2s.

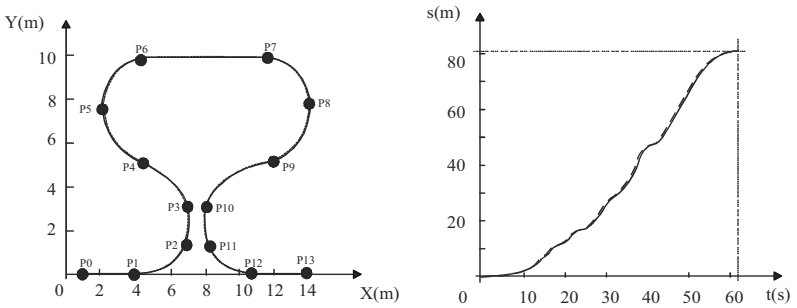


Fig. 4. Planned and tracked trajectory with maximum velocity 1.7m/s.

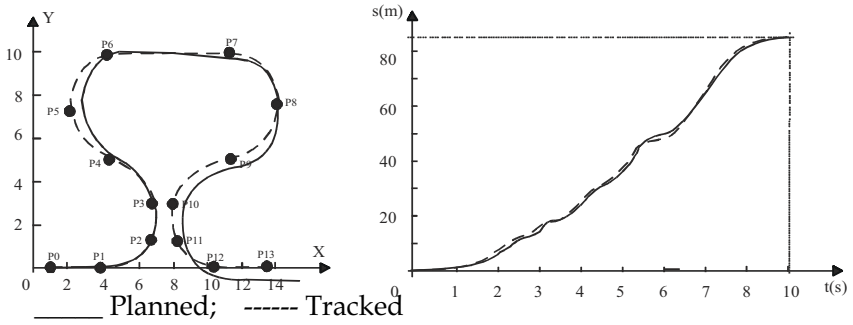


Fig. 5. Planned and tracked trajectory with maximum velocity 17m/s.

In order to test the VP algorithm in trajectories with high dynamic requirements, simulation experiments have been performed with a model of RAM (Prado et al., 2002) that increases the maximum velocity of its tractive motors up to 17m/s, also increasing its acceleration and deceleration performances by the same rate.

Fig. 5 shows the planned trajectory and the trajectory tracked by the model of RAM when executing the same path as in Fig. 4. The total time planned by the proposed algorithm is 35.5% longer than the time that would be consumed if the entire path was navigated at maximum velocity. But it is only 15.2% longer than the time that would be consumed if the velocity was set to its upper boundary function, disregarding the acceleration constraint and continuity. Moreover, the greatest difference occurs in the first segment, because the acceleration constraint causes the planned trajectory to take 3.7s (26.2% of the total time) to reach the velocity boundary function. Low errors are found again: spatial errors are always

lower than 0.80m and temporal errors are lower than 2s for a trajectory of 14.2 s.
The velocity and acceleration profiles of the trajectory of Fig. 5, along with their boundary functions are plotted in Fig. 6. It can be observed that both velocity and acceleration remain within the safety limits while still ensuring their continuity.
Similar results were found for other simulated trajectories with paths generated as cubic splines, but also as clothoids or polar splines (Prado et al. 2003). Particularly regarding velocity constraints, not one path of RAM planned as curves with continuous curvature was found for which the iteration strategies of section 5.3 were needed. Similar results can be expected for any WMR if its spatial path is planned as smooth curves.

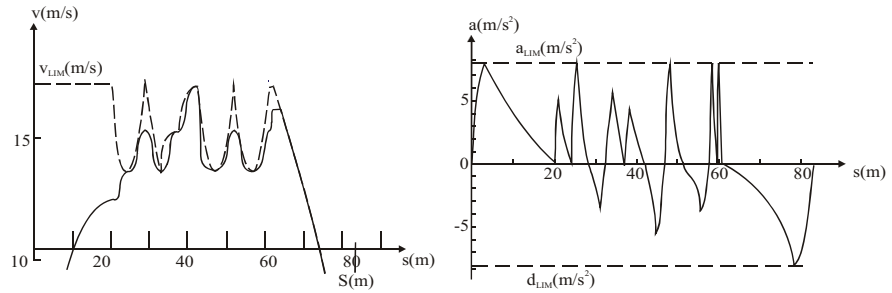
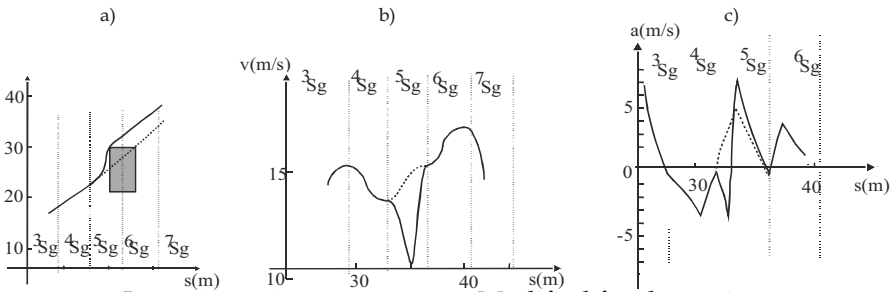


Fig. 6. a) Velocity profile of the trajectory of Fig. 5; b) acceleration profile of the trajectory of Fig. 5



_____ In static environments; ------- Modified for dynamic environments

Fig. 7. a) Planned path; b)Planned velocity profile; c) Planned acceleration profile

When a moving obstacle intersects the WMR's path, its velocity profile is slowed down, as explained in section 6. Fig. 7 shows the effect of including a forbidden region that intersects the 5th segment of the fast trajectory of Fig. 5. The maneuver involves an increase of 4.6s in navigation time. It can be observed that the obstacle is successfully avoided by only modifying the trajectory at the segment that intersects with the forbidden region. The modified trajectory is also continuous in velocity and acceleration and both functions remain within their respective bounded regions.

8. Conclusion

A method to compute velocity, acceleration and deceleration upper boundary functions for WMRs is described in this chapter. These functions confine the operating variables into safety regions in order to generate trajectories that are feasible, safe and with low tracking errors. The method first deals with issues related to the performances of steering, tractive and braking subsystems of wheeled robots, as well as mechanical properties of wheel-ground contact. Next, the motion capabilities of the robot's guide point are calculated as functions of the previous results, attending to the kinematics and dynamics of the complete robot system. Operational constraints caused by the need for fitting and synchronising the robot's motion with its environment are also defined.

The upper limits for velocity based on robot dynamics are fixed not only to avoid total vehicle slippage, as was previously done by other authors, but also to bound the spatial and temporal errors of a trajectory in which the space path generator and the trajectory tracker work under the kinematic problem solution. The definition of these boundary functions depends on either simulation outcomes of a sufficiently accurate dynamic model of the WMR, as complex as needed since it works offline, or on an appropriate number of experimental tests.

Topics involving navigation over 3D terrains and vehicles with very high velocities, such as the presence of ground irregularities, transients for non-stationary manoeuvres, aerodynamic effects, gyroscope torques, etc., are not dealt with in this chapter, though they should be taken into account for robots used in tasks such as outdoor exploration or vehicles developed to run on roads without a driver.

The resulting bounds are included in an algorithm to plan time optimal velocity profiles for a WMR on specific paths. Dealing with PP and VP simultaneously may make it possible to plan faster trajectories in some situations, but such methodologies involve more complex algorithms where it is very difficult to include a significant number of velocity and acceleration constraints that include nonintegrable functions.

The velocity planner proposed in this chapter is able to generate a trajectory with favorable tracking conditions, since:
-It confines the velocity and its first derivative into safety zones limited by functions that can consider any velocity, acceleration and deceleration constraint which can be expressed as a

function of the spatial path.

- It ensures the continuity of the velocity and acceleration over the entire path.
- The jerk of the WMR is bounded.
- It is of low computational cost. Nonetheless, iteration strategies are proposed to solve some specific situations, but such situations are not expected to occur when the spatial path is planned as smooth curves.

Initially, the problem is addressed for static environments, and later an algorithm is proposed to locally modify the velocity profile when crossing points where moving obstacles are detected. It is also an algorithm of low computational cost, thereby providing a beneficial characteristic for its possible use in local control systems.
The method was tested on the WMR RAM with paths planned as cubic spirals. Its successful performance was confirmed by experimental results for the feasible velocities of the WMR prototype and by simulation results for higher velocities.

9. References

Cherif, M. (1999). Motion planning for all-terrain vehicles: a physical modelling approach for coping with dynamic and contact interaction constraints, IEEE Transactions on Robotics and Automation, Vol. 15, No. 2, 202-218, ISSN 0882-4967

Choi, J.S & Kim, B.K. (2001). Near-time-optimal trajectory planning for wheeled mobile robots with translational and rotational sections, IEEE Transactions on Robotics and Automation, Vol. 17, No. 1, 85-90, ISSN 0882-4967.

Gill, M.A.C. & Zomaya, A.Y. (1998). Obstacle avoidance in multi-robot systems: experiments in parallel genetic algorithms. World Scientific Publishing Company, ISBN 9-8102-3423-6, New Jersey, USA.

Guarino Lo Bianco, C. & Romano, M. (2005), Bounded velocity planning for autonomous vehicles, Proceedings of 18th IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 685 – 690, ISBN 0-7803-8913-1, Edmonton, Canada, August 2005, IEEE, New York, USA.

Kiguchi, K.; Watanabe, K. & Fukuda T. (2004) Trajectory planning of mobile robots using DNA computing. Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 8, No. 3, 295-301, ISSN 1343-0130.

Krishna, K.M., Alami, R. & Simeon, T.(2006) Safe proactive plans and their execution. Robotics and Autonomous Systems, Vol. 54, No. 3, 244-255, ISSN 0921-8890.

LaValle, S.M. & Kuffner, J.J. (2001). Randomized kinodynamic planning". International Journal of Robotic Research, Vol. 20, No. 5, 278-400, ISSN 0278-3649.

Lepetic, M.; Klancar, G.; Skrjanc, I.; Matko, D. & Potocnik, B. (2003) Time optimal planning considering acceleration limits. Robotics and Autonomous Systems, Vol. 45, N0. 3-4, 199-210, 2003, ISSN 0921-8890.

Liu, G.Y. & Wu, C.J.(2001). A discrete method for time optimal motion planning a class

of mobile robots, Journal of Intelligent and Robotic Systems, Vol.32,No.1, 75-92, ISSN 09210296.

Muñoz, V.F. (1995) Trajectory planning for mobile robots, Ph.D. Thesis, University of Málaga, Spain (in Spanish)

Nearchou A. (1998). Path planning of a mobile robot using genetic heuristics, Robotica, Vol. 16, No. 5, 575-588, ISSN 0263-5747.

O´Dunlaing, C. (1987). Motion Planning with Inertial constraints, Algorithmica, Vol. 2, No. 1, 431-475, ISSN 0178-4617

Ollero, A.; Simon, A.; García F. & Torres, V. (1993). Integrated mechanical design of a new mobile robot, Proceedings of the First Symposium on Intelligent Components and Instruments for Control Applications, 461-466, Málaga, Spain, May 1992 Pergamon Press, United Kingdom

Prado, M. (2000) Dynamic modeling of mobile robots. Application to the trajectory planning, Ph.D. Thesis, University of Málaga, Spain.

Prado, M.; Simón, A. & Ezquerro, F. (2002). Velocity, acceleration and deceleration bounds for a time-optimal planner of a wheeled mobile robot, Robotica, Vol. 2, N0. 2, 181-193, ISSN 0263-5747.

Prado, M.; Simon, A. Carabias, E.; Perez, A. & Ezquerro, F. (2003). Optimal velocity planning of wheeled mobile robots on specific paths in static and dynamic environments. Journal of Robotic Systems, Vol. 20, No. 12, 737-754, ISSN 0741-2223

Reisten, D.B. & Pin, F.G. (1994). Time-optimal trajectories for mobile robots with two independent acceleration-driven wheels. International Journal of Robotic Research, Vol. 13, No. 1, 38-54, ISSN 0278-3649.

Shiller, Z. (1999). Motion Planning for Mars Rover, Proceedings of I Workshop Robot Motion and Control, pp. 257-262, ISBN 0780356551, Kiekrz, Poland, June 1999, IEEE, New York, USA.

Takeshi, A. (1994). Motion planning for multiple obstacles avoidance of autonomous mobile robot using hierarchical fuzzy rule, Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent System, pp. 265-271, ISBN 0780320727, Las Vegas, USA, October 1994, IEEE, New York, USA.

Wang, Y.; Han, L.; Li, M.; Wang, Q.; Zhou, J. & Cartmell, M. (2004). A real-time path planning approach without computation of Cspace obstacles. Robotica, Vo. 22, No. 2, 173-187, ISSN 0263-5747.

Weiguo, W.; Huitang, C. & Peng-Yung, W. (1999). Optimal Motion Planning for a Wheeled Mobile Robot, Proceedings of IEEE International Conference on Robotics and Automation, pp. 41-46, ISBN 0-7803-5180-0 , Michigan, USA, June 1999, IEEE, New York, USA.

Wong, J.Y. (2001), Theory of Ground Vehicles, John Wiley & Sons, ISBN 0-471-35461-9, New York, USA.

Zalama, E.; Gaudiano, P. & Lopez-Coronado, J. (1995). A real-time unsupervised neural network for the low level control of a mobile robot in nonstationay environment. Neural Networks, Vol. 8, No. 1, 103-1203, ISSN 0893-6080

17

Autonomous Navigation of Indoor Mobile Robot Using Global Ultrasonic System

Soo-Yeong Yi, Byoung-Wook Choi

Dept. of Electrical Engineering, Seoul National University of Technology, Republic of Korea

1. Introduction

One of distinctive features of the recent intelligent autonomous robot from the conventional industrial robot is the mobility, which makes the robot overcome the limited workspace and expands it arbitrary. For autonomous navigation in workspace, a mobile robot should be able to figure out where it is and what direction it moves towards, which is called the self-localization (Singh & Keller, 1991). The self-localization capability is the most basic requirement for mobile robots, since it is the basis of the on-line trajectory planning and control.

The trajectory error of a dead-reckoning navigation, which relies only on the internal sensor such as the odometer or the encoder, grows with time and distance. Therefore, an external sensor is necessary in order to localize the position of the robot in the workspace and to compensate it for the trajectory error. Among the several alternatives, the ultrasonic sensor is regarded as the most cost-effective external sensor and it is widely used for general purposes (Kuc & Siegel, 1987). The methods of the self-localization using the external sensors can be classified into two groups: local methods and global methods. (1) In the local method, a mobile robot makes a local object map using the relative distance data from the environmental objects and matches the local map with a global map database. As a result, the mobile robot figures out its own position in the workspace. (2) In the global method, the mobile robot computes its position directly in the global coordinates using the distances from some reference positions in the workspace.

The local method has some advantages in that collision-avoidance motion and map-reconstruction for the transformed environment are made possible by using the distance sensors as well as the self-localization system. However, this requires massive computations in terms of the local map-making and the matching processes with the global map database. In the extreme case, the robot has to stop moving momentarily in order to obtain the necessary environmental information (Leonard & Durrant-Whyte, 1992) (Ko et al., 1996). On the other hand, in the global method, the local map-making and the matching processes are avoidable and the self-localization is computationally efficient and fast (Leonard & Durrante-Whyte, 1991) (Hernandez et al., 2003).

A global ultrasonic system presented in this chapter is a kind of a pseudo-lite system with a

well-known GPS like structure (Haihang et al., 1997). For the self-localization of an indoor mobile robot, the global ultrasonic system consists of four or more ultrasonic generators fixed at reference positions in global coordinates and two receivers mounted on the mobile robot. Based on the distance measurements between the ultrasonic generators and the receivers and the appropriate data fusion algorithm for the distance data, it is possible to compute the position of the mobile robot in global coordinates.
When several ultrasonic generators are used under a system, the following problems should be taken into consideration;

(1) Cross talk between ultrasonic signals
(2) Identification of each ultrasonic signal
(3) Synchronization between each ultrasonic generator and the receiver to count the TOF(Time-Of-Flight) of ultrasonic signal

In order to solve the above problems, a small-sized RF (Radio Frequency) module is added to the ultrasonic sensors and the RF calling signal is transmitted from the ultrasonic receiver side, i.e. the mobile robot. By using this configuration, the robot is able to control triggering time and sequence of ultrasonic signal generation as well as to synchronize the ultrasonic sensors, so that to localize its own position in the global coordinates. In this chapter, we propose a global ultrasonic system and adopt the EKF (Extended Kalman Filter) algorithm designed for the self-localization. The performance of the autonomous navigation system based on the self-localization is verified through extensive experiments.

2. A global ultrasonic system

The overall structure of the global ultrasonic system is depicted in Fig. 1. The ultrasonic generators are fixed at known positions, $\boldsymbol{T_i} = [x_i,\ y_i,\ z_i]^t$, $i = 1, \cdots, 4$ in the work space, e.g. at each corner of the ceiling. Using the front and the rear ultrasonic sensors situated at $\boldsymbol{P}_f$ and $\boldsymbol{P}_r$, the mobile robot receives the ultrasonic signal and computes the distances by counting the TOF of the signal. It is conveniently assumed in Fig. 1 that the number of ultrasonic generators is four, which can be increased as needed in consideration of the workspace size and objects in the immediate environment. In order to avoid cross-talk between the ultrasonic signals and to synchronize the ultrasonic receivers with the generators, the RF receivers, $RX_1 \sim RX_4$, and the RF transmitter, TX, are added to the ultrasonic generators and the ultrasonic receivers, respectively. By using the RF channel, the mobile robot sequentially activates each one of the ultrasonic generators in successive time slots. Assuming that the delivery time for the RF calling signal is negligible, the ultrasonic signal generation occurs simultaneously with the RF calling signal transmission and it is possible to synchronize the ultrasonic generators with the receivers. In Fig. 1, $h_{f,1} \sim h_{f,4}$ denote the distance between $\boldsymbol{T_1} \sim \boldsymbol{T_4}$ and $\boldsymbol{P}_f$. The distances, $h_{r,1} \sim h_{r,4}$, between $\boldsymbol{T_1} \sim \boldsymbol{T_4}$ and $\boldsymbol{P}_r$ are omitted for brevity. The positions of the ultrasonic receivers on the robot, $\boldsymbol{P_f} = [x_f,\ y_f,\ z_f]^t$ and $\boldsymbol{P_r} = [x_r,\ y_r,\ z_r]^t$, with respect to the center position of the mobile robot, $\boldsymbol{P} = [x,\ y,\ z_c]^t$, can be described as follows:

$$\boldsymbol{P}_f = \begin{bmatrix} x + l\cos\theta \\ y + l\sin\theta \\ z_c \end{bmatrix} \quad \boldsymbol{P}_r = \begin{bmatrix} x - l\cos\theta \\ y - l\sin\theta \\ z_c \end{bmatrix} \tag{1}$$

where l represents the distance between the center position of the mobile robot and the ultrasonic receiver, and θ denotes the heading angle of the mobile robot. It is assumed that the moving surface is flat, so that the z component of the position vectors is constant as z_c in (1).

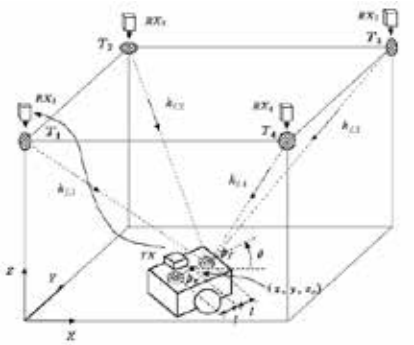


Fig. 1. A global ultrasonic system.

3. The EKF for the self-localization and the autonomous navigation algorithm

The position vector in the $x-y$ plane, $\boldsymbol{r}=[x,\ y]^t$, together with the heading angle, θ, of a mobile robot having differential wheels, follows the state equation (2) in the discrete-time domain (Fox et al., 1997):

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{cases} \begin{bmatrix} x_k + T\upsilon_k \cos\theta_k \\ y_k + T\upsilon_k \sin\theta_k \end{bmatrix} & \text{if } \omega_k = 0 \\ \begin{bmatrix} x_k + \rho_k \cos\theta_k \sin(T\omega_k) \\ \quad -\rho_k \sin\theta_k \left(1-\cos(T\omega_k)\right) \\ x_k + \rho_k \cos\theta_k \sin(T\omega_k) \\ \quad +\rho_k \sin\theta_k \left(1-\cos(T\omega_k)\right) \end{bmatrix} & \text{if } \omega_k \neq 0 \end{cases} \tag{2}$$

$$\theta_{k+1} = \theta_k + T\omega_k$$

where the subscript k is the time index, T, denotes the sampling interval, υ_k and ω_k are the linear and the angular velocities of the robot respectively, and $\rho_k = \frac{\upsilon_k}{\omega_k}$ represents the radius of rotation. The position vector and the heading angle of the mobile robot are augmented so as to become $\boldsymbol{P}=[x,\ y,\ \theta]^t$, which is referred to as the robot posture. The bold and normal symbols represent the vector and the scalar variables, respectively.

As a consequence of (1) and (2), the state equation for the ultrasonic receivers on the robot can be described as follows:

$$\boldsymbol{r}_{f,k+1} = \boldsymbol{f}_f(\boldsymbol{r}_{f,k}, \boldsymbol{u}_k, \boldsymbol{q}_k)$$
$$= \begin{cases} \begin{bmatrix} x_{f,k} + T v_k \cos\theta_k + q_{1,k} \\ y_{f,k} + T v_k \sin\theta_k + q_{2,k} \end{bmatrix} & if\ \omega_k = 0 \\ \begin{bmatrix} x_{f,k} - (l\cos\theta_k + \rho_k \sin\theta_k)(1-\cos(T\omega_k)) \\ \quad + (-l\sin\theta_k + \rho_k \cos\theta_k)\sin(T\omega_k) + q_{1,k} \\ y_{f,k} + (-l\sin\theta_k + \rho_k \cos\theta_k)(1-\cos(T\omega_k)) \\ \quad + (l\cos\theta_k + \rho_k \sin\theta_k)\sin(T\omega_k) + q_{2,k} \end{bmatrix} & if\ \omega_k \neq 0 \end{cases} \tag{3-1}$$

$$\boldsymbol{r}_{r,k+1} = \boldsymbol{f}_r(\boldsymbol{r}_{r,k}, \boldsymbol{u}_k, \boldsymbol{q}_k)$$
$$= \begin{cases} \begin{bmatrix} x_{r,k} + T v_k \cos\theta_k + q_{1,k} \\ y_{r,k} + T v_k \sin\theta_k + q_{2,k} \end{bmatrix} & if\ \omega_k = 0 \\ \begin{bmatrix} x_{r,k} - (l\cos\theta_k + \rho_k \sin\theta_k)(1-\cos(T\omega_k)) \\ \quad + (-l\sin\theta_k + \rho_k \cos\theta_k)\sin(T\omega_k) + q_{1,k} \\ y_{r,k} + (-l\sin\theta_k + \rho_k \cos\theta_k)(1-\cos(T\omega_k)) \\ \quad + (l\cos\theta_k + \rho_k \sin\theta_k)\sin(T\omega_k) + q_{2,k} \end{bmatrix} & if\ \omega_k \neq 0 \end{cases} \tag{3-2}$$

where $\boldsymbol{r_f} = [x_f, y_f]^t$ and $\boldsymbol{r_r} = [x_r, y_r]^t$ represent the positions of the front and rear ultrasonic receivers respectively and $\boldsymbol{q_k} = [q_{1,k}, q_{2,k}]^t$ is the Gaussian random noise with zero mean and $\boldsymbol{Q}$ variance. The measurement equation at the ultrasonic receivers can be modeled as follows:

$$z_{f,k} = h_{f,i}(\boldsymbol{r}_{f,k}, v_k)$$
$$= \{(x_{f,k} - x_i)^2 + (y_{f,k} - y_i)^2 + (z_c - z_i)^2\}^{1/2} + v_k \tag{4-1}$$

$$z_{r,k} = h_{r,i}(\boldsymbol{r}_{r,k}, v_k)$$
$$= \{(x_{r,k} - x_i)^2 + (y_{r,k} - y_i)^2 + (z_c - z_i)^2\}^{1/2} + v_k \tag{4-2}$$

where the measurement noise, v_k, is assumed to be Gaussian with zero mean and G variance, and the subscript, i, denotes one of the ultrasonic generators, $\boldsymbol{T_1} \sim \boldsymbol{T_4}$, which is called by the mobile robot at time k.

From the state Eq. (3-1) and the measurement Eq. (4-1), it is possible to get the following set of equations constituting the EKF estimation for the front ultrasonic receiver position:

$$\hat{\boldsymbol{r}}^-_{f,k+1} = \boldsymbol{f}_f(\hat{\boldsymbol{r}}_{f,k}, \boldsymbol{u}_k, \boldsymbol{0})$$
$$\boldsymbol{V}^-_{f,k+1} = \boldsymbol{A}_{f,k} \boldsymbol{V}_{f,k} \boldsymbol{A}^t{}_{f,k} + \boldsymbol{Q} \tag{5}$$

$$\begin{aligned} \boldsymbol{K}_{f,k} &= \boldsymbol{V}_{f,k}^{-}\boldsymbol{H}_{f,k}{}^{t}(\boldsymbol{H}_{f,k}\boldsymbol{V}_{f,k}^{-}\boldsymbol{H}_{f,k}{}^{t}+G)^{-1} \\ \boldsymbol{V}_{f,k} &= (\boldsymbol{I}-\boldsymbol{K}_{f,k}\boldsymbol{H}_{f,k})\boldsymbol{V}_{f,k}^{-} \\ \hat{\boldsymbol{r}}_{f,k} &= \hat{\boldsymbol{r}}_{f,k}^{-}+\boldsymbol{K}_{f,k}(z_{f,k}-h_{f,i}(\hat{\boldsymbol{r}}_{f,k}^{-},\boldsymbol{0})) \end{aligned} \tag{6}$$

where $\boldsymbol{K}_{f,k}$ is the kalman filter gain, $\hat{\boldsymbol{r}}_{f,k}^{-}$ and $\hat{\boldsymbol{r}}_{f,k}$ represents the *a priori* and *a posteriori* estimations for $\boldsymbol{r}_{f,k}$, respectively, and $\boldsymbol{V}_{f,k}^{-}$ and $\boldsymbol{V}_{f,k}$ represent the *a priori* and *a posteriori* error covariance matrices, respectively, as defined in (7).

$$\begin{aligned} \boldsymbol{V}_{f,k}^{-} &= \boldsymbol{E}[(\boldsymbol{r}_{f,k}-\hat{\boldsymbol{r}}_{f,k}^{-})(\boldsymbol{r}_{f,k}-\hat{\boldsymbol{r}}_{f,k}^{-})^{t}] \\ \boldsymbol{V}_{f,k} &= \boldsymbol{E}[(\boldsymbol{r}_{f,k}-\hat{\boldsymbol{r}}_{f,k})(\boldsymbol{r}_{f,k}-\hat{\boldsymbol{r}}_{f,k})^{t}] \end{aligned} \tag{7}$$

where $\boldsymbol{E}(\cdot)$ denotes the expectation of the corresponding random variables. The Jacobian matrices, $\boldsymbol{A}_{f,k}$ and $\boldsymbol{H}_{f,k}$, in (6) are given as follows:

$$\begin{aligned} \boldsymbol{A}_{f,k} &= \frac{\partial \boldsymbol{f}_{f}}{\partial \boldsymbol{r}_{f,k}}(\hat{\boldsymbol{r}}_{f,k},\boldsymbol{u}_{k},\boldsymbol{0}) \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{8}$$

$$\begin{aligned} \boldsymbol{H}_{f,k} &= \frac{\partial h_{f,i}}{\partial \boldsymbol{r}_{f,k}}(\hat{\boldsymbol{r}}_{f,k},\boldsymbol{0}) \\ &= \begin{bmatrix} \dfrac{x_{f,k}-x_i}{D_{f,i}} & \dfrac{y_{f,k}-y_i}{D_{f,i}} \end{bmatrix} \end{aligned} \tag{9}$$

where $D_{f,i}$ is defined by the following Eq. (10).

$$D_{f,i}=\{(x_{f,k}-x_i)^2+(y_{f,k}-y_i)^2+(z_C-z_i)^2\}^{1/2} \tag{10}$$

The EKF estimation, $\hat{\boldsymbol{r}}_{r,k}$, for the rear ultrasonic receiver position is similar and omitted here for the sake of brevity.
From $\hat{\boldsymbol{r}}_{f,k}$ and $\hat{\boldsymbol{r}}_{r,k}$, the posture estimation for the mobile robot can be described as follows:

$$\begin{aligned} \hat{x}_k &= \frac{\hat{x}_{f,k}+\hat{x}_{r,k}}{2} \\ \hat{y}_k &= \frac{\hat{y}_{f,k}+\hat{y}_{r,k}}{2} \\ \hat{\theta}_k &= \tan^{-1}\frac{\hat{y}_{f,k}-\hat{y}_{r,k}}{\hat{x}_{f,k}-\hat{x}_{r,k}} \end{aligned} \tag{11}$$

Assuming that the estimation error covariances for positions of the front and the rear ultrasonic receiver are the same, the error covariances of the posture estimation are given in (12) as shown in Fig. 2.

$$\begin{aligned} \boldsymbol{V}_{p,k} &= \boldsymbol{E}\,[(\boldsymbol{r}-\hat{\boldsymbol{r}}_k)(\boldsymbol{r}-\hat{\boldsymbol{r}}_k)^2] \\ &= \boldsymbol{V}_{f,k}\ (=\boldsymbol{V}_{r,k}) \\ \boldsymbol{V}_{\theta,k} &= E\,[(\theta-\theta_k)^2] \\ &\approx \tan^{-1}\frac{\boldsymbol{V}_{f,k}}{l} \end{aligned} \tag{12}$$

Eq. (12) implies that the estimation for heading angle becomes more accurate according to the distance between the two ultrasonic receivers. Based on the self-localization given in (11), a simple control input, υ_k and ω_k, to drive the mobile robot toward the given goal position, can be written as (13).

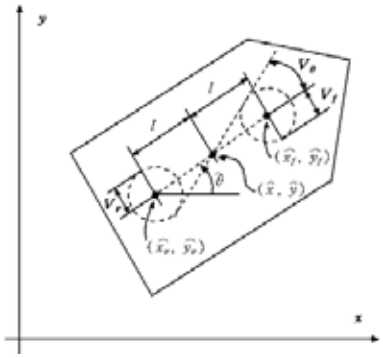


Fig. 2. Error covariances of the posture estimation.

$$\begin{aligned} &\upsilon_k = c \\ &\omega_k = k_\theta(\theta_{d,k}-\theta_k), \quad \theta_{d,k} = \tan^{-1}\frac{y_g-\hat{y}_k}{x_g-\hat{x}_k} \end{aligned} \tag{13}$$

where c and k_θ are positive constants. The mobile robot adjusts its heading angle toward the intended position and moves with the constant velocity as depicted in Fig. 3.

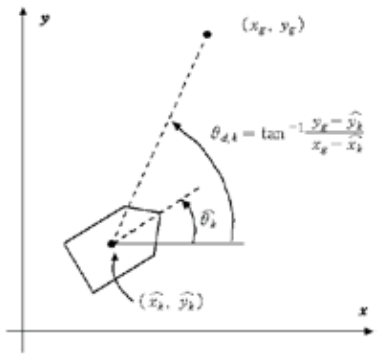


Fig. 3. Navigation control.

4. Experiments and Discussions

In order to verify the performance of the EKF based self-localization and autonomous navigation system using the global ultrasonic system, a simple experimental set-up was established as shown in Fig. 4, which has dimension of 1,500 *mm* and 1,500 *mm* in width and length respectively and 2,500 *mm* in height. The ultrasonic generators installed with the RF receivers are fixed near the four corners of the ceiling and their positions are described in (14).

Fig. 4. Experimental set-up.

$$\begin{aligned} \boldsymbol{T}_1 &= [10.0, 10.0, 2360.0]^t \\ \boldsymbol{T}_2 &= [1427.0, 5.0, 2370.0]^t \\ \boldsymbol{T}_3 &= [1423.0, 1445.0, 2357.0]^t \\ \boldsymbol{T}_4 &= [0.0, 1380.0, 2370.0]^t \end{aligned} \tag{14}$$

At first, a preliminary experiment was carried out for the ultrasonic calibration and the result is presented in Fig. 5.

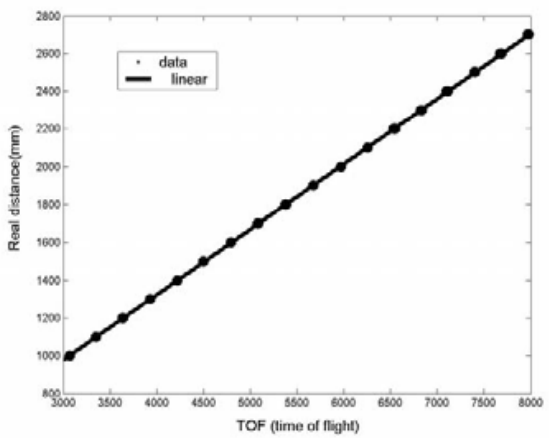


Fig. 5. Real distance with respect to ultrasonic TOF.

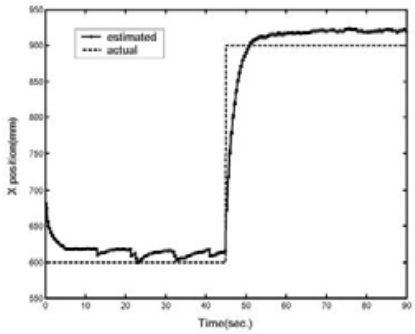
The linear equation relating the ultrasonic TOF to the real distance is given in (15), as obtained from the least-square method, and the variance of the measurement noise is specified as (16).

$$D = 0.34533 \cdot T - 57.224 \tag{15}$$

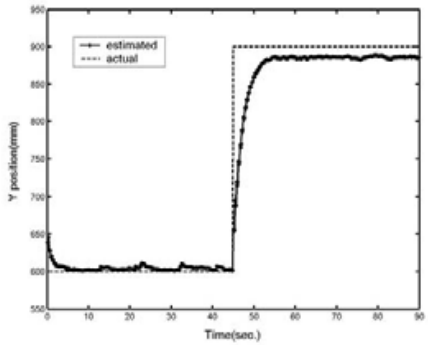
$$G = 1.8 \tag{16}$$

where $D(mm)$ represents the real distance between the ultrasonic generator and the receiver and $T(\mu\text{sec})$ is the ultrasonic TOF.

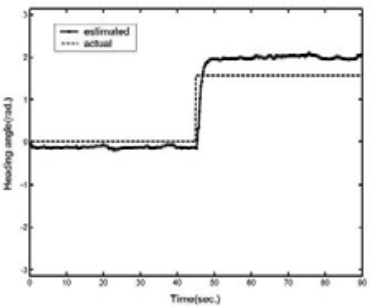
Fig. 6 shows the results of the self-localization experiment, in which the robot is moved manually from the initial posture, $(x, y, \theta) = (600, 600, 0)$ to the goal posture, $(900, 900, \pi/2)$ at 45 *sec*. The initial value of the posture estimation is set arbitrarily as $(650, 650, 0)$. The distance and the heading angle are described by mm and $rad.$, respectively. As shown in Fig. 6, the position errors in the x and y axes are less than 25 mm in the steady-state. Since the distance between the center position of the robot and the ultrasonic receiver is designed as $l = 75\,mm$, the estimation error of the heading angle in (12) becomes $\tan^{-1}(25/75) \approx 0.32\,rad.$ as shown in Fig. 6 (c).



(a) Position estimation in x axis.



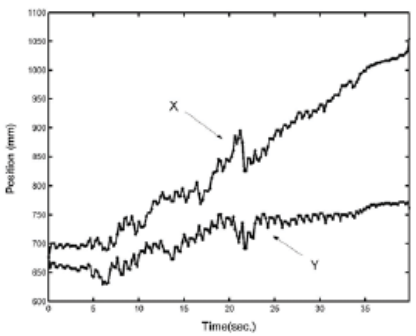
(b) Position estimation in y axis.



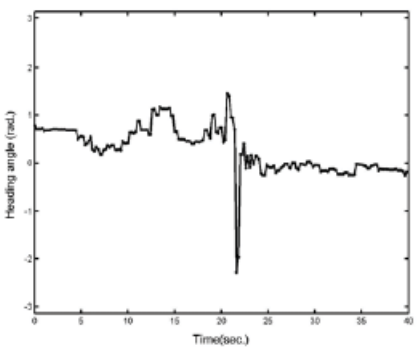
(c) Estimation for heading $_{\theta}$ angle.

Fig. 6. The self-localization of the mobile robot.

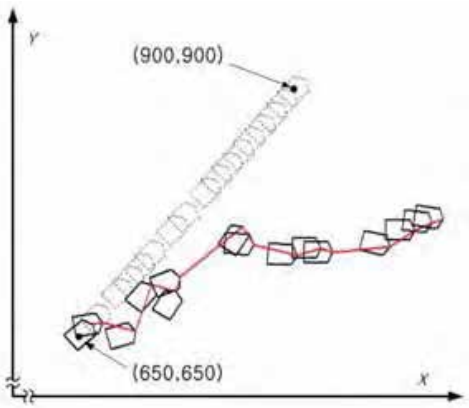
The autonomous navigation system using the global ultrasonic system is compared to the dead-reckoning navigation system on the straight line connecting the initial posture, $(650, 650, \pi/4)$, and the goal posture, $(900, 900, \pi/4)$, in the workspace.



(a) Position in x and y axis



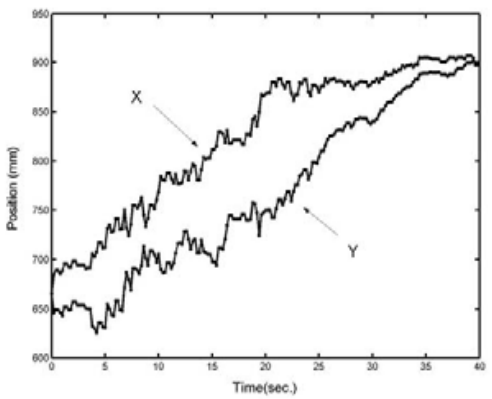
(b) Heading angle



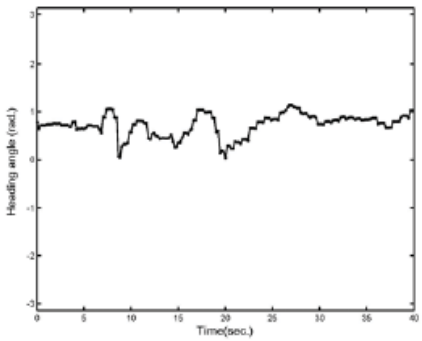
(c) Trajectory in $x-y$ plane

Fig. 7. The dead-reckoning navigation.

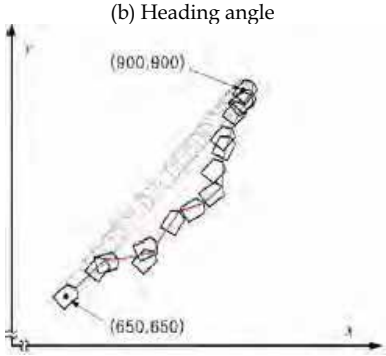
Fig. 7 shows the results in the case of the dead-reckoning navigation, in which the mobile robot cannot reach its goal posture, due to the uncertainties in the state equation. In Fig. 7 (c), the dotted polygons represent the desired postures of the mobile robot with respect to time. The results of the autonomous navigation system based on the self-localization using the global ultrasonic system are presented in Fig. 8 for the same initial and goal postures. As shown in this figure, the mobile robot reaches the goal posture, overcoming the uncertainties in the state equation, and the heading angle at the final position is around $\frac{\pi}{4}$ as desired. It should be noted that the posture data in Figs. 7 and 8 are obtained by using the global ultrasonic system also, thus these values may be different from the actual postures to some degree.



(a) Position in x and y axis



(b) Heading angle



(c) Trajectory in $x-y$ plane

Fig. 8. Navigation with global ultrasonic system.

The size of the ultrasonic region in the work space is dependant on the beam-width of the ultrasonic generator. In the case of a general ultrasonic ranging system, in which both the signal generator and the receiver are lumped together, an ultrasonic generator with a narrow beam-width is preferable in order to avoid the ambiguity and to enhance the measurement accuracy. On the other hand, the proposed global ultrasonic system, which has a distributed signal generator, requires the use of a wide beam-width generator, in order to expand the ultrasonic region in the work space.

5. Conclusions

In this chapter, the global ultrasonic system with an EKF algorithm is presented for the self-localization of an indoor mobile robot. Also, the performance of the autonomous navigation based on the self-localization system is thus verified through various experiments. The global ultrasonic system consists of four or more ultrasonic generators fixed at known positions in the workspace, two receivers mounted on the mobile robot, and RF modules added to the ultrasonic sensors. By controlling the ultrasonic signal generation through the RF channel, the robot can synchronize and measure the distance between the ultrasonic generators and receivers, thereby estimating its own position and heading angle. It is shown

through experiments that the estimation errors are less than 25 *mm* in terms of the position and less than $0.32\,rad.$ in terms of the heading angle. Since the estimation error of the heading angle is dependant on the distance between the two ultrasonic receivers on the robot, it is possible to obtain a more accurate estimation for the heading angle by increasing this distance.

The global ultrasonic system has the following salient features: (1) simple and efficient state estimation since the process of local map-making and matching with the global map database is avoidable due to the GPS-like nature of the system, (2) active cuing of the ultrasonic generation time and sequence through the RF channel, and (3) robustness against signal noise, since the ultrasonic receiver on the mobile robot processes the signal received directly from the generator, instead of through an indirect reflected signal.

In this chapter, it is assumed that an ideal environment exists without any objects in the workspace. Environmental objects may result in an area of relative obscurity, which the ultrasonic signals cannot reach. It is possible to overcome the problems associated with environments containing obstacles by increasing the number of ultrasonic generators in the work space as needed. This enhancement is currently being studied.

6. References

Fox, D.; Burgard, W. & Thrun, S. (1997). The dynamic window approach to collision avoidance, *IEEE Robotics and Automation Magazine,* Vol.4, No.1, March, pp.23-33, ISSN:1070-9932

Haihang, S; Muhe, G. & Kezhong, H. (1997). An integrated GPS/CEPS position estimation system for outdoor mobile robot, *Proceedings of IEEE International Conference on Intelligent Processing Systems*, Beijing, China, October, pp.28-31

Hernandez, S.; Torres, J.M, Morales, C.A. & Acosta, L. (2003). A new low cost system for autonomous robot heading and position localization in a closed area, *Autonomous Robots*, Vol. 15, pp. 99-110, ISSN:0929-5593

Kleeman, L. (1992). Optimal Estimation of Position and Heading for Mobile Robots Using Ultrasonic Beacons and Dead-reckoning, *Proceedings of IEEE Conference on Robotics and Automations*, Nice, France, May, pp.2582-2587

Ko, J.; Kim, W. & Chung, M. (1996). A Method of Acoustic Landmark Extraction for Mobile Robot Navigation, *IEEE Transaction on Robotics and Automation*, Vol.12, No.6, pp.478-485, ISSN:1552-3098

Leonard, J.; Durrant-Whyte, H. (1991). Mobile Robot Localization by Tracking Geometric Beacons, *IEEE Transaction on Robotics and Automation*, Vol.7, No.3, pp.376-382, ISSN:1552-3098

Leonard, J. & Durrant-Whyte, H. (1992). Directed sonar sensing for mobile robot navigation, *Kluwer Academic Publishers,* ISBN:0792392426

R. Kuc & Siegel, M.W. (1987). Physically based simulation model for acoustic sensor robot navigation. *IEEE Transaction on Pattern Analysis and. Machine Intelligence*, Vol.9, No.6, pp.766-777, ISSN:0162-8828

Singh, S. & Keller, P. (1991). Obstacle detection for high speed autonomous navigation. *Proceedings of IEEE International Conference on Robotics and Automation*, pp.2798-2805

18

Distance Feedback Travel Aid Haptic Display Design

Hideyasu SUMIYA
IBARAKI University
Japan

1. Introduction

This chapter introduces approaches of electronic travel (walking) aid (**ETA**) interface for visual information deficit and gives discussions on the high integrity design concept under restrictions of visual- tactile sensational characteristics in substitution process. Here, we start from the concept formulation of ETA. The concept of ETA human interface is based on the sensory substitution between visual and tactile sensation. If human has lost the visual information caused by some sort of environment interference or physical troubles, this ETA assist the subjects to obstacle avoidance self-walking with transferring some environment information (depth, image, object and so on).
Since the first prototype ETA model of TVSS (Tactile Vision Substitution System) in early 1960's, enormous number of research and commercial instruments are developed with visual sensory substitution. Some of these models are still available in markets with improvements (e.g. sonic torch/guide, MOWAT sensor and so on.).
The user with visually impaired using these devices claimed on the difficult understanding at complex environment like in crowds and '**sensory overload'** in use as complexity of understanding, inhibition to other sensory function.

Fig.1. Haptic Travel Aid Image

Applying sensory	**Transfer target**	**Transfer method**	**Display device**	**Transfer part**
Auditory Sense	Distance	*Sound modulation *Voice Guide/Alarm *osseous conduction	Headphone (speaker)	Drum membrane
	Obstacle	Voice guide/Alarm	Headphone (speaker)	Drum membrane
Tactile Sense	Distance	*Mechanical Vibration *Sound modulation *Voice Guide/Alarm *Electromagnetic relay *PZT actuator *micro vibro-Motor	*Electric cane *Vibro Handheld device *Tactile/haptic Display	*forehead, *back, *forearm, *palm, *finger, *fingertip, *tongue
	2Dimage	*Pin stroke *Voltage impression	*2D Electric-driven Braille display *2D Electrode Array	*Fingertip *Back
	Letter/ Texture	*Pin stroke *Voltage Impression	*2D Electric-driven Braille display *2D Electrode Array	*Fingertip *Back)
	3Dfigure	*Pin stroke *Sequential 2D depth contour *Touch /grasp object *Force Feedback	*2D Electric-driven Braille display *Pneumatic pressure *transform object (deforming object , balloon, actuator) *Haptic display	*Fingertip, *Palm, *Tongue)
Baresthesia (Pressure sense)	Distance	*Hydrostatic / pneumatic pressure	*Water /Air injection valve	*Palm)
Electric sense?	2Dimage	*voltage impression	*2D Electrode Array	*Fingertip *Back Retina, Cortex
Thermal /Chemical sense	N.A	N.A	N.A	N.A

Table 1. Visual Sensory Substitution Method (N.A: not available)

For the user-friendly ETA interface design, we should consider more direct operational method and transfer device. From the transfer target classification, ETA type is classified into 3 categories as (A) (edge operation processed) environment image, (B) Distance Information of surrounding obstacles, and (C) combination of (A) and (B). By comparison with ETA, other applications of vision-tactile sensory substitution are listed in character display with Braille display, 2-dimensional image display, Pseudo-3D object figure transfer, and surrounding state guide. (Shinohara et al., 1995), (Shimojo et al., 1997) From the aspect of using sensory classification types, they are listed as following: a) artificial vision with

surgery operation to implant electrode array on retina, b) transfer camera image to implant ed electrode on visual cortex (needs surgery operation), c) make use of auditory sensation with sound modulation or beep or voice announce correspond to depth or object, d) use tactile sense to display visual image or extracted information with tactile/haptic display device.
Furthermore, from the visual-tactile sense conversion method classification, representative ETA method are listed in 1) 2D pin/electrode image display (including pre-processed operation, difference, edge, so on) , 2) low frequency vibro-tactile stimulation based on depth, and 3) selective part haptic display are representative methods.
As mentioned above, current objective is to realize the (none sensory overload) user-friendly ETA interface and to give design guide. This chapter takes up the simple scheme distance feedback ETA using selective stimulation haptic depth display, which possess advantage in fast depth recognition in comparison to existing 2D tactile display type ETA and doesn't need **heavy** surgery operation and concentrates to discuss the adequate design guide for haptic sensational restrictions.
Following background of ETA section, basic characteristics and restrictions of tactile/haptic sensation are discussed, which are important for user-friendly haptic ETA design. Based on this consideration, we introduce a concept of selective skin part stimulation distance feedback ETA interface system and continue to the discussion of user-friendly and effective distance-tactile stimulation conversion and device design from the aspect of avoidance walk and environment recognition.

2. Background and History of ETA

The background and history of ETA are shown in Table 2. In 1960s, TVSS(Tactile Vision substitution System) are studied at Smith-Ketlewell Labs. L.KAY' s Sonic Torch is produced as the first practical ETA device and continues in following famous commercial models, MOWAT sensor, Laser Cane, and so on. These ETA devices are basically surrounding distance transfer device, which gives distance information along pointed direction back to user with converted tone, sound modulation or mechanical vibrations. In addition, not only portable device, there exists travel guidance system in building as functional welfare facility, which gives voice announce about the important location and attribute information to the visually impaired by detecting sensor under the floor or street with electric cane. Beyond portable ETA concept, Guide Dog Robot, which scans the environment image and street line and precedes and guide subjects, has been developed in 1978 (TACHI, 1978)
For Image transfer, 2D electric driving pin array (electric Braille display, OPTACON) are developed and investigated on the static 2D image recognition of character and/or figures. Human's character and image recognition with millimetric order electric pin-array and electrode array 2D display recognition characteristics are investigated not only from physical aspect but also from the psychological one. The phantom effect and adequate display rate and method are summarized (Shimizu 1997).
For user-friendly ETA device design, Tactile Display Glove and Line Type Haptic Display, which project distance to selective skin part, was proposed and shown direct operational performance (SUMIYA et al, 2000)(SUMIYA, 2005)

Year	Device Name	Transfer Target	Transfer Method	Implementer/Planner
1960s	TVSS (Tactile Vision Substitution System) /The voice	Gray level camera Image	400 millimeter Solenoid activator array/ Soundscape	Smith-Kettlewell Institute (USA) Carter Collins Peter Meijer
(1965)	SonicTorch,	distance	Tonal pattern	Leslie KAY(UK)
	GuideCane			Johann Borenstein (USA)
1978	SonicGuide (KASPA)	distance	Tone	Leslie KAY(UK)
1978	Guide Dog Robot (MELDOG MARK I)	Camera Image, distance	Voice Annouce (precede subject)	Susumu TACHI, Kazuo TANIE,Yuji HOSODA, Minoru ABE (JPN)
1973	MOWAT Sensor		Vibration, Tone	MOWAT G,C, (USA)
1980s	Trisensor			Leslie KAY(UK)
	Radar on a chip			Lawrence Livermore Labs(USA)
	LaserCane (Polaron, Wheelchair Pathfinder)		Vibration, sound(+audible warning signal)	Nurion-Raycal (USA)
	Lindsey Russell Pathsounder	Obstacle detection	Audible Signal /silent vibration	Lindsey Russell (USA)
	Sensory 6		Tone pitch	Brytech Corporation
	(Proto-type)	Camera image	2D Electrode array (20*20 condensor discharge electrode,150Hz)	National Institute of Bioscience and Human- Technology (JAPAN)
	Miniguide			Greg Phillips (Australia)
(1984)	Sonic Pathfinder (Nottingham Obstacle Detector)		stereophonic	Tony Heye (UK)
1996		Cortical implant		Schmidt et al (GERMANY)
	(Dobelle Artificial Vision System)			(Dobelle Institute(USA))
1997	Artificial Retina	Implant on Retina		Ito, N. et al (JPN)

Table 2. ETA(Electronic Travel Aid) Development History

Artificial vision with implanting surgical operation technique have started in 1990s, the 1st type of artificial vision is implanting electrode on retina and connect with neuron to cortex

(Ito et al, 1997)(Rizzo et al, 2001). The 2nd type is 2D edge operation processed image information is applied to 2D electrode array implanted on the visual cortex. They are still in clinical testing, but reports the recovery of some part of visual function so that they can follow child and grasp objects.
As represented by the work of Weber E. H., Fechner G.T., Von Frey, Weinstein S., Schmidt R., and Verrillo, R. t. et al, enormous number of Tactile Sense analysis and brain projection are investigated. (WEBER 1978) These investigated result are closely linked to user-friendly ETA design and quoted in next section.

3. Problems of the Visual-Tactile Sensory Sybstitution

This section gives significant characteristics and restrictions on tactile sense.

3.1 Static Characteristics of Tactile Sense Recognition

Static Tactile Recognition Characteristics are investigated and derived numerical values by the achievements of our predecessors as follows.
(1) 2 Points Discrimination Threshold
E.H.WEBER has measured the 2 point discrimination threshold. Including this result, he summarized and published his famous book, 'The sense of Touch'.

Part	Threshold(mm)	Part	Threshold
Forehead	22.5	Dorsal hand	31.5
Apex of tongue	1.0	Fingertip	2.3
Lip	4.5	Dorsum of finger	7.0
Front of forearm	40.5	Anterior tibial(shin)	67.5

Table 3. (Statical Pressure) 2 Points Discrimination Threshold on Human Skin(Average).

(2) WEBER-FECHNER Law
In 'The sense of Touch', he wrote the concept of **WEBER-FECHNER Law**. The rate of sensitivity resolution vs. applied range takes constant value. If E is sensing event, S is caused sense.(sensitivity)

$$\Delta E / E = cons\tan t \tag{1}$$

If the variation of output sense takes constant for the variation of given event.

$$\Delta S = \Delta E / E \tag{2}$$

Solve this difference equation as differential equation, then sensitivity is expressed as next equation.

$$S - A\log_{10} E + B \tag{3}$$

(Here, B is an offset value.)

(2) Baresthesia (Static Pressure Sensing Limit)
Frey, V., M. has measured the human's static pressure sensing limit on skin part.(Frey. 1896)

Sensing Part	Sensing Limit (g/mm^2)	Sensing Part	Sensing Limit(g/mm^2)
Apex of tongue	2	Abdominal Area	26
Dorsum of antebrachium (backside of forearm)	33	Lumber Division (pars lumbalis)	48
Front of forearm	8	Dorsal hand	12
Fingertip	3	Sura (calf)	16
Dorsum of finger	5	Planta pedis (sole)	

Table 4. Static Pressure Sensing Limit on Human's Skin Surface

(3) Spatial Positioning Error

Spatial Positioning Error is the error between the actual stimulating point on skin surface and the subject's recognized point. (Weinstein, 1968) (Schmidt et al., 1989) (Schmidt, 2001)

Part	Error(mm)	Part	Error(mm)
Forearm	9	Forehead	4
Upperarm	11	Abdomen	9
Shoulder	9~10	Distal thigh	11
Fingertip	2	Calf	11
forehead	4	Sole	7~8
Finger	2	Toe	2

Table 5. Spatial Positioning Error

(4) Sensory Representation in The Cerebral Cortex

For further work on brain function connected to tactile sense, the tactile sense projection map on the cerebral cortex studied by Penfield W. and Rasmussen T are the milestone in this research field and the projected area and relative position gives many hint on next issue. Even Penfield's Homunculus image still gives strong impact for every viewer. (Penfield & Boldrey, , 1937)(Rasmussen et al., 1947)

3.2 Dynamic Characteristics of Tactile Sense Recognition

(1) Dynamic Range/Sensitivity

Tactile sense frequency sensitivity and discrimination performance shows the different characteristics from stimulating contactor dimensional size. Bolanoski et al(Bolanoski et al., 1988) and Verrillo(Verrilo 1968,1969) investigated tactile sensitivity for vibro-stimuli with diameter rod with $2.9cm^2$ and $0.005cm^2$ correspondingly. The frequency discrimination result shows U-curve and highest sensitivity at 250Hz. Lav. Levänen and Hamforf studied the frequency discrimination value for the deaf subjects and the hearing subjects, and showed the smallest frequency difference at palm and finger are 21±3Hz and 28±4 in 160-250Hz (1s duration, 600 stimuli), correspondingly.

Sumiya et al reported the tactile vibro-stimuli recognition rate follows the WEBER-FECHNER Law in recognition for quick random frequency presentation, as seen for ETA sensing and the resolution is at most 20% of total area at a point on forearm. This means the resolution in high speed sensing at most 5 partition of the searching range. For the case of linear partition at 10m searching area and projected to frequency, the resolution segment is at most 2m or smooth recognition (Sumiya et al., 2000). That is also considerable to

introduce log partition in the aspect of WEBER-FECHNER Law, but in late section, Linear partition shows higher performance for ETA (blindfolded walk) with the reason of easy association of environment map.

Kyung has studied perceptual tactile frequency discrimination characteristics on finger with 0.7mm diameter small contactor, as popular size for Braille display. The subject's sensitivity shows almost 100% recognition rate at the frequency bands of 1-3Hz and 18-32Hz. As frequency increases up to 500Hz around, the frequency discrimination performance decreases gradually to 85% on abdominal finger. On palm, the discrimination characteristic shows flat curve through 1 to 560Hz at the value of 85±5Hz.(Kyung et al., 2005)

(2) Learning Effect

Learning effect of frequency discrimination at 20hz around is reported by Imai. (Imai et al., 2003). Learning Effect after daily training shows rapid gains within 2 weeks, and later within 4 weeks the improvement of learning effect shows still raise in a measure , but shows conversion.

Learning effect for distance recognition with distance-selective skin part stimulation ETA device has tested for the several distance partition methods. The result shows the linear partition shows best learning effect for blindfolded walk time performance. (Sumiya 2005)

(3) Fatigue effect/ Saturation from repetitive stimulation

Fatigue effect and Saturation of tactile sense has not tested precisely. For steady use as ETA device, this should be investigated.

4. Specific Requirement for Travel Aid Interface

Compared with other tactile/haptic image recognition device, ETA should satisfy the next factor.

(1) Fast response

Slow response brings interruption to user in operation and understanding. For human's reaction, response time should satisfy 15Hz or higher for mobility.

(2) Accordance between operational direction and spatial direction

Easy-to-use needs this coincidence to help user's intuitive operation for environmental grasping. Left-Right direction, close-far direction , rotation direction should match to operator-centered intuition. This also help fast and high adaptability. In addition, this is also important for learning effect when the first handling is not easy.

(3) Transfer environmetal attribute information

ETA user request the detected object's attribute information. For example, the detected object is whether person or still object. Color, Material, Moving Direction, Hardness, … and functional meaning. In introducing next research, using the stereo image and Nueral net scheme, several pattern has specified and send to user (human, stairs, rectangular form obstacle) to the assigned stimulating point. The combination of 2D tactile display and selective part stimulating haptic ETA device would cover this problem. Even single 2D tactile display, Shimojo et al proposed the unique approach for 3D information transfer with time sequence depth contour display.(shimojo et al. 1999) .

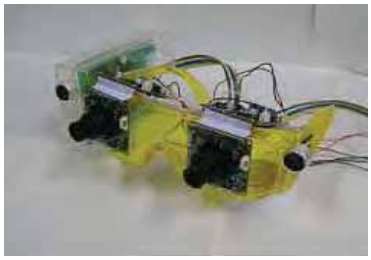
(4) reconstruction of environmental state (spatial relative position, forgetting factor)

It is important that the user 's mental image is close to real environment. Author has tried to questionnaire sheet to draw the obstacle position and ETA user 's image map after

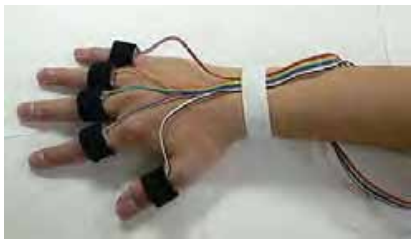
blindfolded walk with ETA. Without knowing global image around subjective, normally it is hard to grasp the global location of subject with simple one directional distance sensing. Gibson has proposed '**The theory of Affordance**' that subjects will sense envirionmental information and in accordance with self-moving.(Gibson) , That is to say, subject's motion it self help the grasp of environment. Simultaneously, however, this captured information fades away from forgetting. The questionnaire method can not respond to the real time image reconstruction. Although the mental (recognized) space is hard to estimate, but it will give some hint to know the projected map on visual cortex. PET: positron emission tomography, fMRI: functional magnetic resonance imaging, MEG: magnet-encephalography are impossible to monitor moving subject's inner brain activity from their structure. From the report that retina image is projected to primary visual cerebral cortex (Fellerman and Essen, 1991), it could be possible to brain activity of moving subjects with '**Optical Topography',** which will monitor the cerebral cortex activity, if the resolution will increase. (Watanabe er al. 1996) (Plichta et al. 1997)

5. Current Approach

The concept of walking aid human interface for visually impaired is based on the sensory substitution (visual sensation to tactile sensation conversion). A variety of electronic travel aid system (ETA) or sensory substitution system(SSS) have been developed, and some of these products are casted in commercial market :(Lenay et al. 1997).



(a) Environmental State Detection Goggle (ESDG)



(b) Selective Stimulation Haptic Display 1

(c) Selective Stimulation Haptic Display 2

Fig. 2. Selective Stimulation Haptic ETAInterface

The distance display method of these systems are classified as distance-sound modulation (mono-, stereophonic), distance-tactile vibration (frequency modulation), distance-tactile pressure, distance-selective stimulation part mapping using mechanic vibration or electronic stimulation and so on. Recently, DOBELLE System and Artificial Retina System are developed and broadcasted in several media, but they need surgical operation and still cost high denomination. Simultaneously, this is the critical point, the tactile sensation would rather suit to 2 Dimensional sense with low resolution of force, frequency sensitivity. Therefore, vision to tactile sense substitution studies are still exploring the design of 3 dimensional depth display that vision transfers as seen in interesting literature: (Shimojo et al., 1999). Our main concern is to develop affordable price and direct operational feel walking aid human interface without surgical operation and long learning process. This study concentrates on the distance-selective skin part stimulation mapping method that known as high performance in absolute repetitive cognition : (Shinoda et al., 1998), (Sumiya et al., 2000), (Nakatani et al., 2003), (Kobayashi & Ohta, 1999).

First, we show the concept of our distance-selective skin part mapping type tactile walking aid interface. Secondly, this paper discusses the basic concept of the stimulation point number and design of the selective skin part mapping interface with the consideration of tactile sensation characteristics and restriction of human being. At the third stage, we propose different types of distance display interface that takes a count of absolute distance value priority display method and relative distance priority display method. Through the blindfolded walking experiment, we inspected the performance in spatial perception and direct operation factor for these proposed interfaces with their direct recognition accuracy and learning Effect.

5.1. System Concept and Configuration

(1) Distance Display Walking Aid Tactile Interface

This paper concentrates on The walking aid system with the distance-selective skin part stimulating. User wears the Environmental State Detection Goggle(ESDG) .

This sensing unit installs one ultrasonic distance detection sensor unit and stereo camera unit. This plural detected signal are sent to the signal processing unit in personal computer to produce target pointed distance and 3D-depth map for further surroundings state and object information feedback with surround state transfer tactile display through Surrounding Attribute estimate Neural network Scheme[4]. This detected target-pointed distance information in the user-facing direction is converted into depth transfer tactile display signal. Briefly, the detected distance information is classified into some range because of the selective stimulation part number restriction. In the case of tactile Display Glove installs 5 selective stimulating point on root of each finger. If detected distance would classified in range i, then i-th finger's stimulator activated and applies vibration.

Then user acquires the information of the detected distance range in facing direction. With the danger priority consideration, closest range mapped to first finger stimulator and mapped each finger to corresponding distance range in upwards. The issues are distance-selective points mapping.

(2) Distance Display Walking Aid Tactile Interface

Then user gets the information of the detected distance range in facing direction.

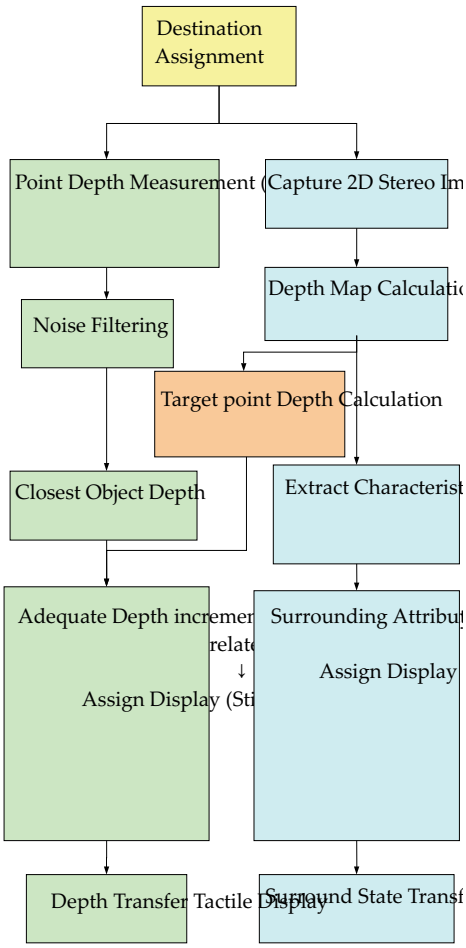


Fig. 3. Haptic ETA operation Flow

(3) Distance Display Method for spatial perception
Then user gets the information of the detected distance range in facing direction.
(4) Consideration on Selective Stimulation Part Number
Humans' discriminative resolution on spacial perception, especially along depth direction, are not constant.(WEBER-FECHNER Law). For the walking circumstance, the mainly influencing distance range could be assumed from 0 to 10m (or less 5m in slow exploring walk in complex circumstance) for path-finding purpose. In this range, as seen in space cognitive resolution characteristics, we can assume that the distance cognitive resolution also possesses linear characteristics with depth(target distance).

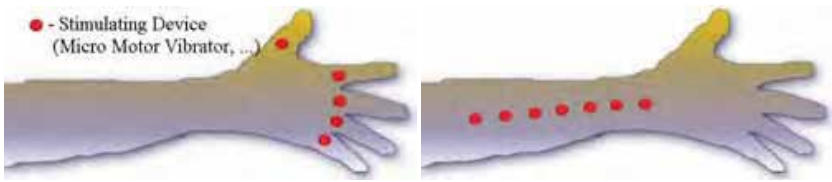


Fig. 4. Selective Stimulation Contactor Alignment

If we assume the distance resolution as Δy =1cm at y = 1m depth, from the above linear resolution hypothesis, the resolution in all travel of target range keeps Δy/y= 0.01. If we set the starting detection distance and the ending distance as ds, de correspondingly. The necessary number of selective stimulator number is calculated as follows.
The first stimulator's sensing depth d(0) is

$$d(0) = d_s \tag{4}$$

If we assume the incremental value right to reference point would proportional to resolution, then the mapped distance to the n-th stimulator as described in eq.(2)

$$d(n) = d_s(1 + \Delta y / y)^{n-1} \tag{5}$$

Necessary number of stimulator should satisfy next eq.(6).

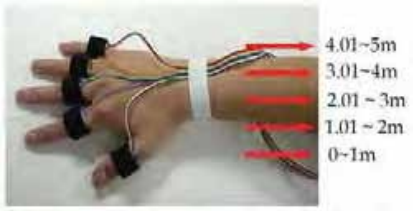
$$n > d_e / (d_s \log(1 + \Delta y / y)) \tag{6}$$

This is the realization of Weber-Fechner law in suit to logarithmic sensitivity.
In latter section, the performance comparison among linear distance partition and other partition method projection method are discussed.

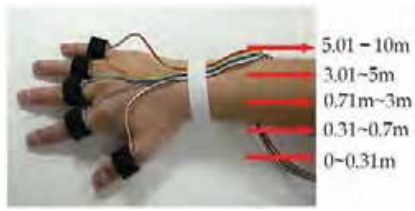
5.2. Distance Transfer Tactile Display

(1) Linear Mapping
Distance-Selective Stimulation Mapping using A-TDG
Mapping as written in section 1-3-2, a detected signal converts corresponding selective skin part stimulation. In this section, we tried 3 types of linear discrete depth range division and distance transfer display using mapping into corresponding selective finger part stimulation.
(2) Personal Space Based Mapping
Personal Space is psychological concept that human being keeps their own social territory in other people's communication. range1:closest relation as holding each other as family touch each shoulder to 30cm, range2:friend relation holding each hands to make a familiar greeting closer one reach 70cm, range3: acquaintance region to exist situation in street until both opened reach 1.5m, unknown person that if other person enters room where main person stays, then should feel attention on each other, room size,5m or more. This range is mapped to each finger stimulator. This conversion methods is psychological factor priority conversion
(3) Stride Based Mapping
As every person has experienced in growing process, the different feeling of space perception in childhood and after grown up sometimes cause the ponder of the scale-based

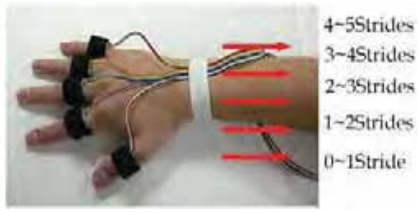
space perception idea. Stride Based Mapping is the humans' physical factor priority mapping based on the stride length as a representative physical dimension. This Base mapping is linear mapping but is not divided by constant Number for each user. Each user takes one depth division taken from their own stride size. The issue is whether this method will fit each user and cause the smooth walk to erase the unfit feeling caused by the personal physical size factor.



1)Equidistance Mapping



2)Personal Space based Mapping
(Mental Factor Priority)



3) Stride-based Mapping

Fig. 5 Distance - Selective Part Mapping

5.3 Blindfolded Walking Experiment

(1) Comparison Between Linear Discrete Mapping

Distance-Selective Stimulation Mapping using A-TDG Mapping as written in section 1-3-2, a detected signal converts corresponding selective skin part stimulation. In this section, we tried 3 types of linear discrete depth range division and distance transfer display using mapping into corresponding selective finger part stimulation. While blindfolded walking experiment, subjects walking motion is recorded in video camera. Subject location is derived from the captured video image. The obsctacle alignment position is changed for each trial . Afer walking experiment, subject draw the target image on right-questionnaire map.

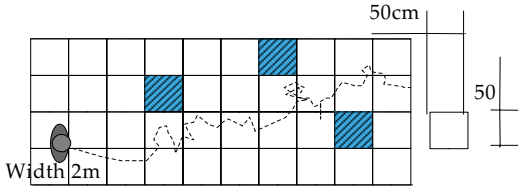
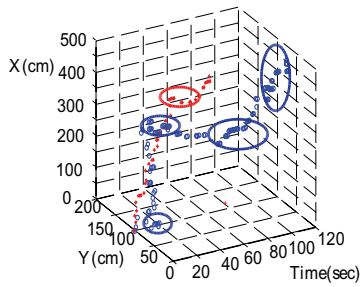
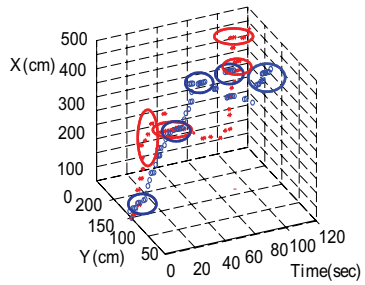


Fig. 6. Blindfolded Walking Experiment

5.4. Walking Aid Interface Performance

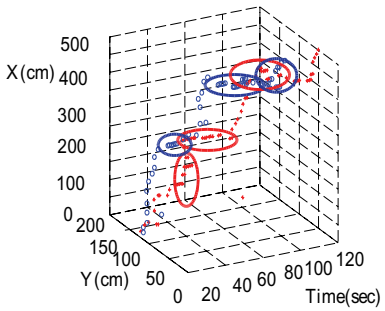


(a) Linear Mapping



(b) Personal Space Based Mapping –Mental Factor

o –first trial
* - second trial



(a) Stride Based Mapping - Physical Factor

Fig. 7. Walking 3D Trajectory in time-line

Walk Time(sec)	No1	No2	No3	No4	No5	Average
Relative Indicator	65.4	77.9	135.7	103.8	56.1	88.8
Relative Indicator (sensitivity prior.)	116	152.3	135.7	107.8	61.8	118.2

Table 6. Walk Through Performance (Absolute Indicator Value as 100(%))

(1) Spatial Reconstruction after Walking Performance
Mapping as written in 5.2, a detected signal converts corresponding selective skin part stimulation. In this section, we tried 3 types of linear discrete depth range division and distance transfer display using mapping into corresponding selective finger part stimulation.
(2) Learning Effect
From the walking performance time performance, Learning Effect is explicitly detected only for the linear (equidistance) mapping. (This signal conversion may already too simple to activate the humans' ability. This method still does not have enough transfer function about fast 3-Dimensional space perception.

6. Environmental Attribute Transfer

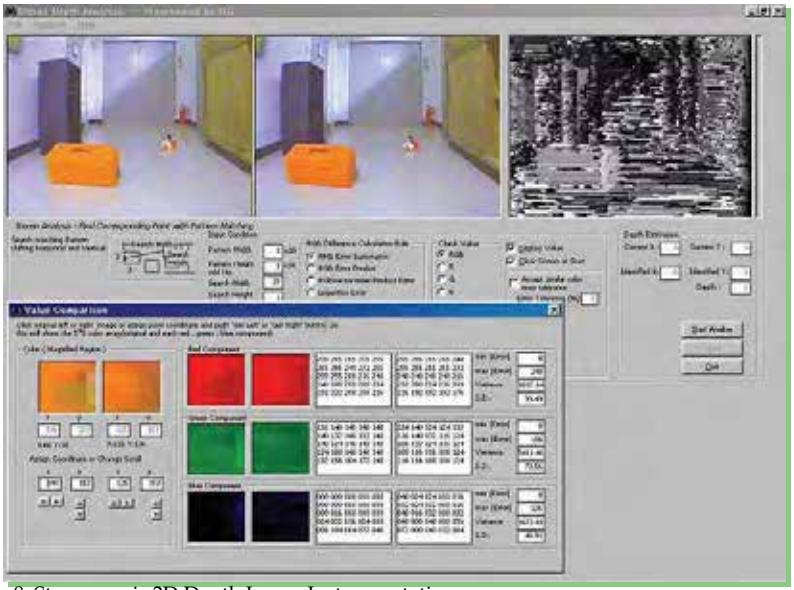
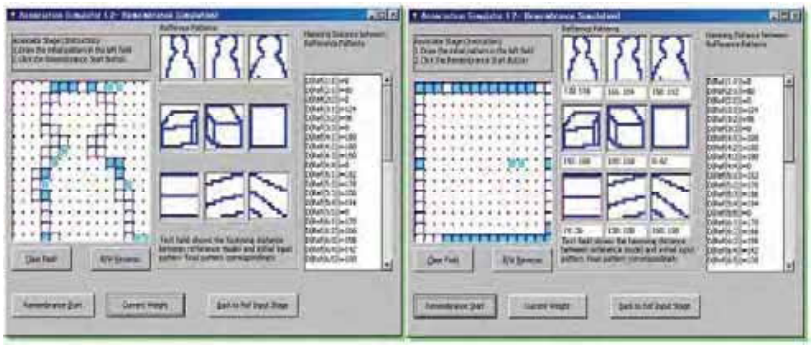


Fig. 8. Stereoscopic 2D Depth Image Instrumentation

This system generates the 2D distance (depth) map in VGA size using L-R image pattern matching with continuity of surface. The inner product vector similarity is proposed for the L-R pattern shift calculation and additional algorithm improvement is processed for Speed-up. Generated depth map information is also used to estimate detected object and state. Extracted outline with the consideration of depth and colors pattern is normalized as fit to 16*16 bit image, then set to the inter-connected Neural Net scheme. For the estimation accuracy improvement, this system adopt combinational learning and estimating algorithm. See (sumiya, 2005) for more details.



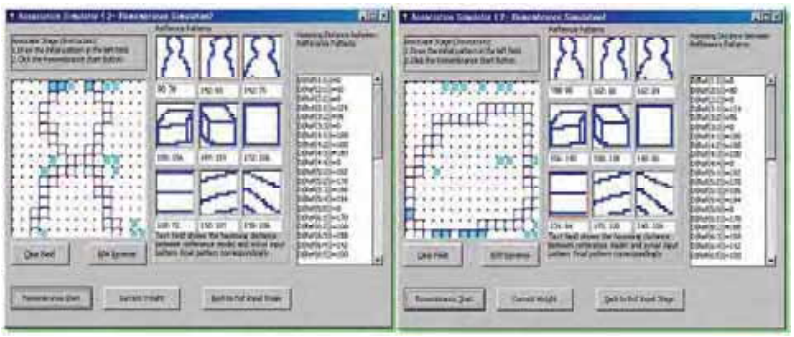


Fig. 9. Estimated Searched Object and State Attribution

7. Conclusion

This chapter aims at the user-friendly ETA design. As a preliminary information, several tactile characteristics including sensing range, sensitivity, dynamic factors are introduced, which are critical to design tactile/haptic device. In the latter half of this chapter, an example model of recent ETA design is introduced along the user-friendly real-time operation. Current ETA system are still on the process to gives satisfactory level of environmental reconstruction for user. Even under restriction of tactile charcterists, this design concept will give some hint to create new device to activate human's sensitivity. (e.g. magnification of sensing resolution, extending human sense). Recent studies of sensory substitution system has another aspect to extend human original sense. These approach would be called as 'Hyper sense'. Currently, the most popular user of ETA is the visually impaired. But this applicable area is more extensive. ETA technique will take the certain role in 'Hyper Sense' technique.

8. Consideration (Current Unsolved Problem)

As discussed in section 4., visualization of constructed mental map /image is next interesing issue. Especially for human in motion, the portable brain activity monitoring system should be introduced. If starting with the cortex neighbor activity monitoring , Optical topography technique is a possible candidate , if resolution will increase up to mm order.

9. Acknowledgement

This study is one of the outcome that is supported by the Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science in 2005. We would like to take this cite to thank this encouraging support and to extend our appreciation to JSPS for providing this opportunity to present our activities and research environments.

10. Refernces

Bach-y-Rita, Paul; Collins, Carter C.(1971), Sensory Substitution Systems Using the Skin for the Input to the Brain, *AES Volume 19 Number 5 pp. 427-429; May*

Bolanoski, S.J., Jr., Gescheider, G.A., Verrillo, R.T., & Checkosky, C.M. (1988), "Four Channels Mediate the Mechanical Aspects of Touch", Journal of the Acoustical Society of America, Vol. 84, pp.1680-1694.

Cowey A, Stoerig P (1995), Blindsight in monkeys. Nature 373, 247-249

Diller, T.T., Schloerb, D., and Srinivasan, M.A.,(2001) "Frequency response of human skin in vivo to mechanical stimulation", RLE Technical Report No.648, MIT

Fechner, G. T. (1964). Elemente der Psychophysik [Elements of psychophysics]. Amsterdam: E. J. Bonset. (Original work published 1860).

Fechner, G. T. (1966), Elements of psychophysics (Vol. 1), (H. E. Adler, Trans.), New York: Holt, Rinehart & Winston.(Original work published 1860).

Fellerman and Van Essen (1991), Cerebral Cortex, vol 1, 1-47

Frey, V. M. (1894) : Beiträge zur Physiologie des Schmerzsinns (2 Mitt.). Berichte über die Verhandlungen der Königlich Sächsischen Gesellschaft der Wissenschaften; 46: 283–297.

Frey, V. M(1896): Untersuchungen über die Sinnesfunktionen der menschlichen Haut. Erste Abhandlung:Druckempfi ndung und Schmerz.Abhandlungen der mathematisch-physischen Klasse der Königlich Sächsischen Gesellschaft der Wissenschaften; 23: 208–217.

Frey, V. M. (1922): Versuche über schmerzerregende Reize. Z Biol ; 76: 1–24.

Gibson J.J.(1979), "The Theory of Affordances," The Ecological Approach to Visual Perception, Lawrence Erlbaum, Hillsdale

Imai et al. (2003), Learning of Tactile Frequency Discrimination in Humans, Human Brain Mapping 18:260–271

Ito, N., Shirahata, A., Yagi, T., Matsushima, T., Kawase, K., Watanabe, M., Uchikawa, Y., (1997) Development of Artificial Retina using Cultured Neural Cells and Photoelectric Device: A Study on Electric Current with Membrane Model, Proceedings of the 4th International Conf. on Neural Information Processing (ICONIP'97), pp.124-127

Ki-Uk Kyung, Minseung Ahn, Dong-Soo Kwon, Mandayam A. Srinivasan (2005), Perceptual and Biomechanical Frequency Response of Human Skin: Implication for Design of Tactile Display, First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'05) pp. 96-101

Kobayashi, M., Ohta, M.(1999), Walking Guide System for the Visually Impaired by using Three-dimensionalSound, Proceeding of IEEE SMC , WP02-1

Kyung, K., Ahn, M., Kwon, D., Srinivasan, M. A. (1978), Perceptual and Biomechanical Frequency Response of Human Skin: Implication for Design of Tactile Display, First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'05) pp. 96-101Weber E. H.: The Sense of Touch, Ernst Heinrich Weber, Academic Pr.,(1978/12)

Lenay, C., Canu, S. & Villon, P. (1997). Technology and Perception : the Contribution of Sensory Substitution Systems. In Proceedings of the Second International Conference on Cognitive Technology, Aizu, Japan (pp. 44-53). Los Alamitos: IEEE.

Levänen S., Hamdorf D. (2001), Feeling vibrations: enhanced tactile sensitivity in congenitally deaf humans, Neuroscience Letters 301, 75-77

Minsky, M., Ouh-Young, M., Steele, O., Brooks, F., Behensky, M.(1990), "Feeling and seeing: issues in force display". Proceedings of the Symposium on 3D Real-Time Interactive Graphics, ACM

Müller J(1838) : Handbuch der Physiologie des Menschen für Vorlesungen. Coblenz, J. Hölscher, 1835–1840 (two volumes translated into English); in Baly W: Elements of Physiology. London, Taylor & Walton, 1838–1843.

Mowat, G. C. (1973). Guiding Devices. U.S. Patent #3,718,896 issued Feb 27

Nakatani. M., Kajimoto, H., Sekiguchi, D., Kawakami, N. and Tachi,S.(2003), "3D form display with shape memory alloy," in Proc. of 13th International Conference on Artificial reality and Telexistence (ICAT), pp. 179-184

Nelson, Felleman, Kaas, J. (1980), comp Neurol. 192, 611-644

Penfield WG, Boldrey E (1937), Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation, Br4ain 60, 389

Penfield, W & Rasmussen, T. (1950), The cerebral cortex of man, New York: Macmillan

Plichta MM, Herrmann MJ, Baehne CG, Ehlis AC, Richter MM, Pauli P, Fallgatter AJ (2006). [Event -related functional near-infrared spectroscopy (fNIRS): Are the measurements reliable?], Department of Psychiatry and Psychotherapy, Laboratory for Psychophysiology and Functional Imaging, University Hospital Wuerzburg, Neuroimage., Jan 27; [Epub ahead of print]

Pressey N.(1977), "Mowat Sensor." Focus, Vol. 11, No. 3, pp. 35-39.

Rasmussen T, Penfield W (1947), Further studies of sensory and motor cerebral cortex of man, Fed. Proc. 6, 452

Rizzo, Wyatt, Humayun, Liu, Chow, Eckmiller, Zrenner, Yagi, Abrams (2001), Retinal Prosthesis: An Encouraging First Decade with Major Challenges Ahead, Ophthalmology, 108, 1

Robert F. Schmidt (1989), Human Physiology, Springer-Verlag Berlin and Heidelberg GmbH & Co. K (1989/12/31)

Robert F. Schmidt (2001), Physiologie kompakt, Springer-Verlag GmbH (2001/04)

Schmidt R, Schmelz M, Ringkamp M, Handwerker HO, Torebjörk HE (1997): Innervation territories of mechanically activated C nociceptor units in human skin. J Neurophysiol ; 78: 2641–2648.

Shimizu Y., (1997)

http://www.tsukuba-tech.ac.jp/info/treky:8080/index.html/kaken/home.htm,

Shimojo M., Shinohara M., Fukui Y. (1997): "Shape Recognition Performance Depending on Pin-Matrix Density for 3D Tactile Display", Trans. EIC, Vol.J80-D-II, No.5, pp.1202-1208

Shimojo, M., Shinohara, M.and Fukui, Y.(1999),Human Shape Recognition Performance for 3-D Tactile Display, IEEE Trans. on Sys., Man, and Cyb. Part A: Sys. and Hum., vol. 29, pp. 637-644.

Shinoda, H., Asamura N., and Tomori, N. (1998). Tactile Feeling Display Based on Selective Stimulation to Skin Mechanoreceptors. Proc. 1998 IEEE Int. Conf. Robotics and Automation, Vol.1, pp. 680-686.

Shinohara M. , Shimizu Y. and Mochizuki A. (1995): "Development of a 3-dimensional tactile display for the blind", Proceedings of European Conference on the Advancement of Rehabilitation Technology-ECART, pp.404-405

Sumiya, H., Yonekura, T. & Shiraishi, M.(2000), Walking guide tactile interface for the visually impaired using plural self-calibrating sensors, 6th Intl. conference on VSMM2000

Sumiya, H. (2005), A travel guide human interface using depth and surround state transfer tactile display, Proceedings of 11th international conoference on virtual systemsand multimedia

Sumiya, H., Gotoh, Y., Shiraishi, M. (2005), Walking Aid Human Interface for The Visually Impaired Using A-TDG/R-TDI Interface, Proceedings of 36th International Symposium on Robotics, ISR 2005

Tachi, S., Komiyama, K., Hosoda, Y., and Abe, M., (1978) : "Study on Guide Dog (Seeing-Eye) Robot(I)", Bulletin of Mechanical Engineering Laboratory, No. 32, pp. 1-14, (1978.4)

Tachi, S., Tanie, K., Komiyama, K., Hosoda Y., and Abe, M. (1981), : Guide Dog Robot ~ Its basic plan and some experiments with MELDOG MARK I, Mechanism and Machine Theory, Vol.16, No.1, pp. 21-29

Verrillo, R.T. (1968), A Duplex Mechanism of Mechanoreception, in D. R. Kenshalo(Ed.), The Skin Sense (Proceeding of the First International Symposium on the Skin Senses), Thomas, Springfield, Illinois, pp.139-159

Verrillo, R.T., Fraoli, A.J., & Smith R.L.(1969) "Sensation magnitude of vibrotactile stimuli", Perception and Psychophysics, Vol. 7, pp.366-372

Weinstein, S. (1968), "Intensive and extensive aspects of tactile sensitivity as a function of body part, sex and laterality", in D. R. Kenshalo (Ed.), The Skin Sense (Proceeding of the First International Symposium on the Skin Senses), Thomas, Springfield, Illinois, pp.195-222

Watanabe E, Yamashita Y, Maki A, Ito Y, Koizumi H. (1996),[Non-invasive functional mapping with multi-channel near infra-red spectroscopic topography in humans.],Neurosci Lett. 1996 Feb 16;205(1):41-4. PMID: 8867016

Watanabe et al.(1996), http://www.hitachi-medical.co.jp/info/opt-e/index.html

19

Efficient Data Association Approach to Simultaneous Localization and Map Building

Sen Zhang, Lihua Xie & Martin David Adams
Nanyang Technological University
Singapore

1. Introduction

A feature based approach to simultaneous localization and map building (SLAM) is to use the information obtained by sensors mounted on a vehicle to build and update a map of the environment and compute the vehicle location in that map. One of the critical problems in obtaining a robust SLAM solution is data association, i.e. relating sensor measurements to features in the map that has been built thus far (Guivant, 2000). SLAM relies on correct correspondence between data obtained from the robot sensors and the data currently stored in the map.

There have been numerous approaches to data association. In stochastic mapping, the simplest method is the NN algorithm which is a classical technique in tracking problems (Bar-shalom & Fortmann, 1988). The great advantage of NN is its $o(mn)$ computational complexity in addition to its conceptual simplicity (Here m is the number of sensor measurements and n is the number of existing features in the map). It performs satisfactorily when clutter density is low and sensor accuracy is high. However, during the process of SLAM, especially in complex outdoor environments, clutter level is usually high and the innovations in matching different observations obtained from the same vehicle position are correlated. In this situation, the NN algorithm may accept a wrong matching, which leads to divergence in state estimation.

In order to improve the robustness of data association, Neira and Tardos (Neira & Tardos, 2001) presented an approach using a joint compatibility test based on the branch and bound search with a high computational cost. Juan Nieto et al. (Nieto et al., 2003) give a fast SLAM algorithm for data association by applying the multiple hypotheses tracking method in a variety of outdoor environments. The experimental complexity estimates show that if the number of features in one scan is large, these algorithms will not be fast enough for real time implementation. In other approaches, Bailey et al. consider relative distances and angles between points and lines in two laser scans and use graph theory to find the largest number of compatible pairings between the measurements and existing features (Bailey et al., 2000). The work of Lim and Leonard (Leonard & Lim, 2000) applies a hypotheses test to implement data association of the relocation in SLAM using geometric constraints. Castellanos and Tardos (Castellanos et al., 1999) use binary constraints to localize the robot with an a priori map using an interpretation tree. In these methods, geometric constraints among features are used to obtain hypotheses with pairwise compatible parings. However, pairwise compatibility doesn't guarantee joint compatibility, and additional validations are required.

Data association based on a 0-1 integer programming (IP) problem for multi-sensor multi-target tracking was firstly proposed in (Morefield, 1977) and later improved in (Poore & Robertson, 1995; Poore & Robertson, 1997; Poore & Robertson, 1994). In these articles, the data association and tracking problem was formulated as a multi-dimensional assignment problem.

By extending the multi-sensor and multi-target tracking problem, the data association in SLAM is formulated as a 0-1 integer programming problem in (Zhang et al., 2004*b*; Perera et al., 2003; Perera et al., 2004). In (Perera et al., 2003; Perera et al., 2004), the data association in SLAM is first formulated as a three dimensional assignment problem where two frames of scan data are used to establish the correspondence between the measurements and the features stored in SLAM. This algorithm can track features very reliably especially in high clutter density environments. However, three dimensional assignment problem is NP-hard (Hochbaum, 1997) and only heuristic optimization algorithms are available for an approximation solution because of the computational complexity. In (Zhang et al., 2004*b*), a parallel development of data association in SLAM is formulated as a two dimensional assignment problem. The chapter reports the detail of this work.

In this chapter we present an efficient integer programming (IP) based data association approach to SLAM. In this approach, the feature based SLAM data association problem is formulated as a 0-1 IP problem that considers only 1 frame of scan data. Therefore, it is formulated as a two dimensional assignment problem for which many optimization algorithms such as those in (Poor & Robertson, 1994; Poor & Robertson, 1997; Miller & Franz, 1993; Miller & Franz, 1996; Storms & Spieksma, 2003) can be applied. The IP problem is approached by first solving a relaxed linear programming (LP) problem. In order to reduce the computational burden, a validation gate is applied to reduce the size of the solution space. An iterative heuristic greedy rounding (IHGR) process based on linear programming techniques (Miller & Franz, 1993; Miller & Franz, 1996; Storms & Spieksma, 2003) is then proposed to obtain a suboptimal solution to the integer programming problem. The algorithm has moderate computational requirements.

Detailed simulation and experimental results show that the proposed method gives a much higher success rate of data association for environments of high density features than the NN algorithm while the cost of computation is moderately higher than the latter. Further, experiments in a real outdoor environments show that the NN algorithm leads to a diverged vehicle pose estimate whereas the proposed algorithm performs satisfactorily. As compared to other existing methods such as the JCBB algorithm, our approach has a lower computational complexity and provides a good trade-off between accuracy and computational cost.

The chapter is organized as follows: Section 2 is devoted to an IP formulation for data association in SLAM. Section 3 presents an iterative heuristic greedy rounding algorithm for the IP problem. Section 4 shows some simulation and experimental results. Some conclusions are drawn in Section 5.

2. Problem Formulation

In this section we formulate the data association of SLAM as a 0-1 integer programming problem similar to (Perera et al., 2003; Perera et al., 2004). A mathematical framework of SLAM which is based on the extended Kalman filter (EKF) will be applied.

Data association in SLAM is the decision process of associating measurements (observations) with existing features in the stochastic map. It should be noted that the term

"measurements" (observations) in this chapter refers to the observed features after feature extraction rather than the raw sensor measurements. Generally, the number of the measurements obtained in each scan is not equal to the number of features whose positions are estimated by the EKF. Each measurement may either (1) belong to a previously known geometric feature or (2) be a new geometric feature or (3) be a spurious measurement (also called a false alarm). On the other hand, there also exist features that do not have associated measurements in the current scan. A dummy element is applied to denote the case of a false alarm or a new feature or a feature that does not have an associated measurement here.

Assume that there are M measurements from the latest scan which are to be assigned to N existing features in the map built based on the previous scans. It is also assumed that the measurements are independent. Typically, $M \neq N$. We define the binary assignment variable

$$x_{nm} = \begin{cases} 1 & if \quad measurement\ m\ is\ assigned\ to\ feature\ n \\ 0 & otherwise. \end{cases} \tag{1}$$

Note that $x_{n0} = 1$ implies that the feature N has no associated measurement in the current scan, and $x_{0m} = 1$ implies that the measurement m is not assigned to any of the existing N features, but instead, assigned to a dummy feature–false alarm or newly initialized feature. In the data association process, we make the reasonable assumption that one measurement originates from at most one feature, and one feature can produce at most one measurement. Therefore, the following constraints can be imposed to the association variables:

$$\sum_{m=0}^{M} x_{nm} = 1,\ n = 1, 2, \cdots, N, \tag{2}$$

$$\sum_{n=0}^{N} x_{nm} = 1,\ m = 1, 2, \cdots, M. \tag{3}$$

Our goal is to match the sensor's observations with the features by providing estimates of the features' positions relative to the vehicle pose at the time of the current scan. In order to formulate the 2-D assignment problem, a generalized likelihood ratio which involves feature state estimates for the candidate associations is used to assi5ogn a cost to each association. Similarly to the multi-target tracking problem, we maximize a likelihood function LH as follows:

$$LH = \prod_{\{n,m \in E_{nm}\}} \Lambda(z_{m,} f_n), \tag{4}$$

$$\Lambda(z_{m,} f_n) = \frac{1}{2\pi |S|^{1/2}} \exp\{-\frac{1}{2}[z_m - \hat{z}_n]^T S^{-1}[z_m - \hat{z}_n]\}, \tag{5}$$

where z_m is the m-th measurement of the scan, $\hat{z}_n$ is the predicted relative position of the n-th feature by the EKF, S is the covariance matrix of $z_m - \hat{z}_n$, and E_{nm} is the set of all possible assignment pairs. The likelihood ratio $\Lambda(z_m, f_n)$ is in fact the probability that the mth measurement matches the n-th feature in the current sensor scan. In order to constitute a 2D-assignment optimization problem, instead of maximizing the product of matching probabilities, we can minimize the negative log-likelihood ratio. To this end, we define:

$$c_{nm} = -\ln \Lambda(z_m, f_n). \tag{6}$$

Then, an equivalent cost function for equation (4) can be written as follows:

$$\min \sum_{\{n,m \in E_{nm}\}} c_{nm} x_{nm},$$

wher. Thus, data association in SLAM can be formulated as the following 0-1 integer programming problem:

$$\min \sum_{\{n,m \in E_{nm}\}} c_{nm} x_{nm} \tag{7}$$

subject to

$$\sum_{m=0}^{M} x_{nm} = 1,\ n = 1, 2, \cdots, N, \tag{8}$$

$$\sum_{n=0}^{N} x_{nm} = 1,\ m = 1, 2, \cdots, M, \tag{9}$$

where

$$x_{nm} \in \{0,1\} \quad \text{and} \tag{10}$$

$$c_{nm} = \begin{cases} 0 & if\ m = 0\ or\ n = 0, \\ -\ln \Lambda(z_m, f_n) & otherwise. \end{cases}$$

In this algorithm, two points should be noted:

- If a measurement does not fall into the 3σ region of any of the existing N features (see the gating process in Subsection 3.1), we assume that it is a new feature and add it to the SLAM map directly. It will no longer be considered in the above 2D-assignment.
- If an existing feature does not have any measurement that falls into its 3σ region, we consider that this feature is not observed in the current scan. No further matching will be carried out for this feature

3. The IHGR Based Data Association Algorithm

In this section, we apply an LP based algorithm to solve the data association problem formulated in the last section. The method is a combined IHGR and LP algorithm. In order to reduce the computational burden, a validation gate is applied first to reduce the above global association to several local associations.

3.1 Gating

In order to reduce the solution space, gating is first applied. Only measurements that are close enough to the predicted state of an existing feature are considered possible candidates of association with the feature. The criterion of gating is given by:

where

$$\tau_{ij} = \nu_{ij}^T s^{-1} \nu_{ij} \le \varepsilon \quad i = 1, 2, \cdots N;\ j = 1, 2, \cdots, M,$$

$$\nu_{ij}(k+1) = z_j(k+1) - \hat{z}_i(k+1|k),$$

and S_{ij} is the covariance of the innovation ν_{ij}. In this equation, $\hat{z}_i(k+1|k)$ can be calculated from an EKF and $z_j(k+1)$ means the jth measurement in the current scan.

Note that since ν_{ij} is a Gaussian random variable, τ_{ij} is a random variable following the χ^2 distribution. Thus, a validation gate, ε, is used to decide whether the measurement

$z_j(k+1)$ is a close enough match to the predicted feature position. From the χ^2 distribution table, we know that with a probability of 0.99 for a variable of two degrees of freedom. Here we set $\varepsilon = 6.63$.

3.2 Iterative Heuristic Greedy Rounding

3.2.1 Relaxation and Rounding Technique

Many combinatorial optimization problems can be attacked with approximation algorithms that yield a feasible solution in polynomial time with cost close enough to the optimal one (Hochbaum, 1997). Such approximation algorithms can be loosely categorized as combinatorial approximation algorithms and LP based approximation algorithms. Here, we are interested in the latter category. In order to solve the IP problem, we often firstly relax it into a LP problem (Cormen et al., 2001). In general, the relaxation refers to the action of relaxing the integer requirement of a linear IP to turn it into an LP. However, the optimal solution to the LP problem in general does not coincide with the solution to the initial IP problem. One of the basic techniques which is widely exploited to derive a LP based approximation algorithm is LP-based rounding. It refers to how to construct a feasible solution for the IP from the LP (Cormen et al., 2001; Parker & Rardin 1988). Thus, the LP-based approximation algorithm first relaxes the IP problem to a LP problem, solves the LP problem and then converts the fractional optimal solution of LP to an integer solution. The heuristic algorithm applied here is based on the relaxation and rounding algorithms as described in (Miller & Franz, 1993; Miller & Franz, 1996; Storms & Spieksma, 2003). These algorithms are used for other applications. It is noted that the two dimensional assignment problem can also be solved by other optimization techniques.

3.2.2 IHGR Procedure

By changing the integer constraint $x_{nm} \in \{0,1\}$ to $0 \le x_{nm} \le 1$, the IP problem is relaxed to a LP one. The LP problem can be solved by basic LP algorithms, such as the Simplex algorithm (Vajda, 1981). If the optimal solution x_{op} of the LP-relaxation is fully integer-valued (in this case all decision variables will have the value of either 0 or 1) then the solution x_{op} is optimal for the 0-1 IP problem in Equation (7) (Parker & Rardin, 1988). Otherwise, we apply the IHGR procedure (see, e.g. (Miller & Franz, 1996)). Observe that the larger the decision variable x_{nm}, the higher the probability that the m-th measurement associates with the n-th feature. Hence, the algorithm starts by setting the maximum decision variable (with a value close to 1) to 1 and all other entries in the same row and column to zero to meet the constraints (8) and (9). Then, solve the LP problem for the remaining assignment matrix and repeat the IHGR procedure to decide the next pairing of measurements and features. The process is continued until all measurements have been assigned. In this manner, a feasible (but not necessarily optimal) solution for the original IP problem is constructed.

In the IHGR procedure, when x_{nm} is set to 1, all variables in the column and row associated with the specific set in E_{nm} must be set to 0. Once a variable is forced to a certain value, it is not allowed to change any more. To achieve this, all rounded variables and all implicated variables are discarded from the IHGR procedure. In this way, the IHGR will never set the

value of a variable twice. This deletion of variables also applies to the initial LP solution, i.e. all variables with value 1 and all zero-valued variables implicated by them, are removed. The IHGR algorithm repeats the actions of selection, rounding and deletion until there are no variables left. The outcome will then be a feasible solution to (7).
The algorithm described above can be summarized in the following 4 steps:
Step 1: Relax the IP problem to a LP problem and solve the LP problem. If the solution is fully integer valued, then stop. Otherwise, go to Step 2.

- Step 2: Set the maximum decision variable to 1 and other entries in the same row and column to zero.
- Step 3: Delete the elements that have been decided, go back to step 1 for the rest of the assignment matrix.
- Step 4: Repeat the above relaxation, selection, rounding, and deletion steps until all the elements are assigned.

Observe that the IHGR can be implemented efficiently, but it is clear that there is no guarantee that this heuristic procedure yields the optimal solution to the IP problem. However, the simulations and experiments to be discussed in the following section show that the IHGR does result in acceptable feature-measurement assignments of which the achieved cost is close to the optimal cost.

3.3. Algorithm Complexity

Due to the application of the gating process that is affected by random factors, we cannot give an exact description of the complexity. However, we know that in any fixed dimension, LP can be solved in polynomial linear time (linear in the input size) (Karmarkar, 1984). For our case, the input size is $M \times N$. Thus, we can roughly know the worst-case complexity of the proposed algorithm is

$$O(MN + (M-1)\times(N-1) + \cdots + (|M-N|+1)\times 1)$$

Neira and Tardos (Neira & Tardos, 2001) presented a data association approach–JCBB. JCBB performs incremental construction and search of an interpretation tree of joint association hypotheses. The gating determines acceptable hypotheses and performs branch and bound pruning of the search space. The discussion in (Neira & Tardos, 2001) does not provide any theoretical bound, but gives an empirical complexity estimate of $O(1.53^N)$, where N is the number of observed features. When N is large, the algorithm will have a high computational complexity. For example, when $N = 30$, $1.53^{30} = 347330$. For the algorithm presented here, $MN+(M-1)(N-1)+\ldots+(|M-N|+1)=9455$ where M =30 that is the worst case. Therefore, when the observed feature number is large (such as more than 30), our algorithm is faster than JCBB. In the simulation, we also found that when N is large, for example $N \geq 30$, JCBB is too slow to work while the algorithm proposed can work well (The example can be seen in the section of experimental results).

4. Simulation and Experimental Results

The algorithm presented was tested in two different environments, an artificial environment and a real outdoor environment. In these two environments, SLAM was implemented by using the data association algorithm proposed in the last section.

4.1 Results of Simulation Environment
4.1.1 Simulation Example 1

The first test environment is established by randomly generating some features and assuming that the vehicle's trajectory is a circle whose radius is 62 meters. The robot moves at a constant speed and the heading angle changes 1 degree at each sampling instant. The environments involve 105 features which are randomly distributed in a region of 120 meters by 120 meters. The standard deviations of the simulated laser sensor are taken as $\sigma_r = 0.01$ meters and $\sigma_\theta = 0.0005$ radians. When the vehicle moves, some of the features are observed. We assume that the features' positions are unknown which is the case in SLAM and match the features with the observations by using the NN and the IHGR algorithms, respectively.
In order to compare the successful rates of these two data association algorithms, we change feature positions except one feature 200 times to implement the SLAM. The features' positions are randomly generated each time with uniform distribution in space. We apply the NN data association algorithm and the IHGR data association method to perform the SLAM processes, respectively. In the simulation, we only fix the position of one feature and investigate the successful rate of the matching of this feature with its observation in one scan (scan 15) during the SLAM. The data association successful rate for this feature when using IHGR algorithm is 96.5% (193/200) while it is only 81% (162/200) for the NN algorithm.

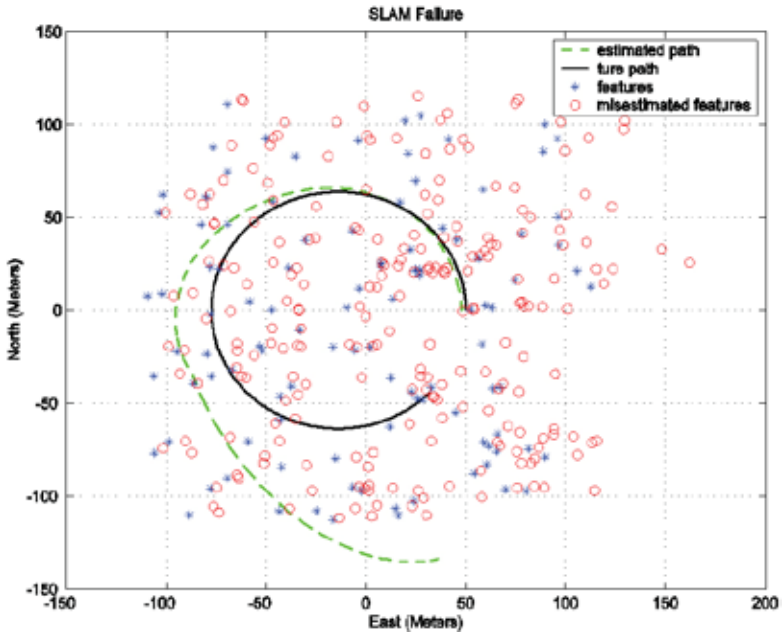


Fig. 1. Unsuccessful mapping when applying the NN data association algorithm in the SLAM process.

When the features are closely located, the NN algorithm fails. Figures 1 and 2 show the SLAM results where the NN algorithm fails whereas the proposed IHGR algorithm performs well. In this case, the positions of 3 features in the environment are fixed and they are located at (27, 20.5), (26, 19.5) and (26.5, 19), respectively. The remaining features are randomly distributed. It can be observed from Figure 1 that the NN algorithm leads to diverged estimates of vehicle pose and feature positions. On the other hand, our IHGR method performs very well as observed in Figure 2. In fact, the vehicle's true path is almost overlapped with the estimated one. The global error between the estimated vehicle's position and the ground truth can be seen in Figure 3.

A comparison on execution time between the NN algorithm and the IHGR based data association algorithm versus the number of features observed in one scan is shown in Figure 4 (the algorithms are run on Pentium IV PC, 1.7GHz) for the cases when the NN algorithm is able to give successful SLAMs. In the figure, the mean execution time means the average CPU time used for 50 Monte Carlo runs for data association algorithm. In the simulation, we extract data at each scan under different feature number and assume that the number of measurements is the same as that of the existing features (this is the most time consuming case). The result shows that our IHGR based method has moderate computational requirement and is implementable in real-time applications.

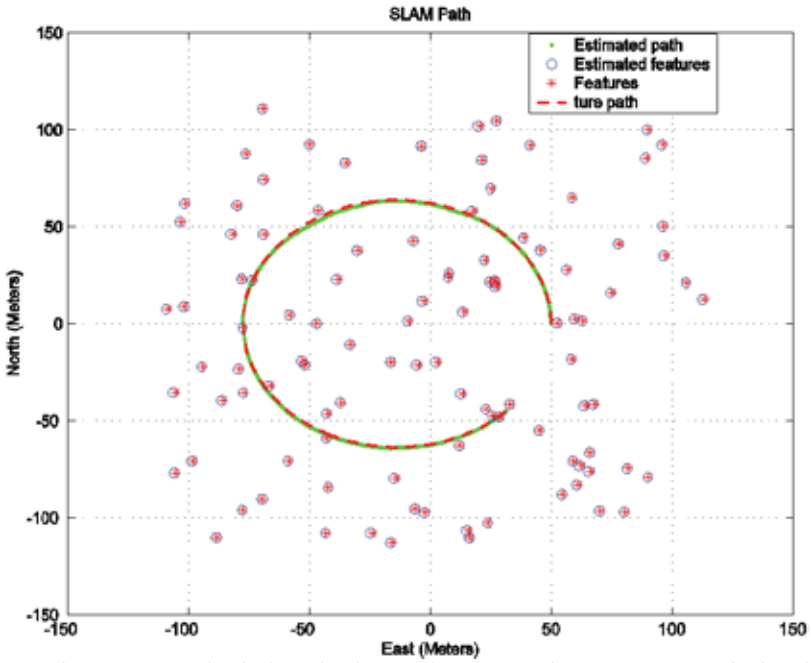


Fig. 2. The mapping and vehicle path when applying IHGR data association method in the SLAM process.

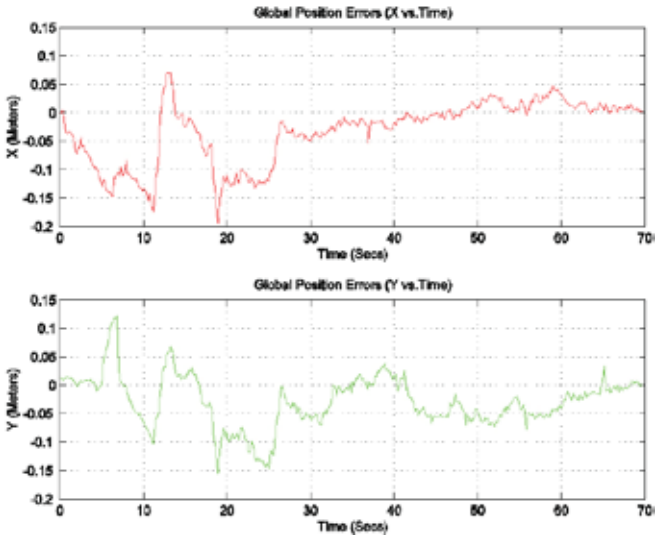


Fig. 3. The vehicle's position errors in global coordinates between the ground truth (generated data) and the estimated position using IHGR data association algorithm in the SLAM process.

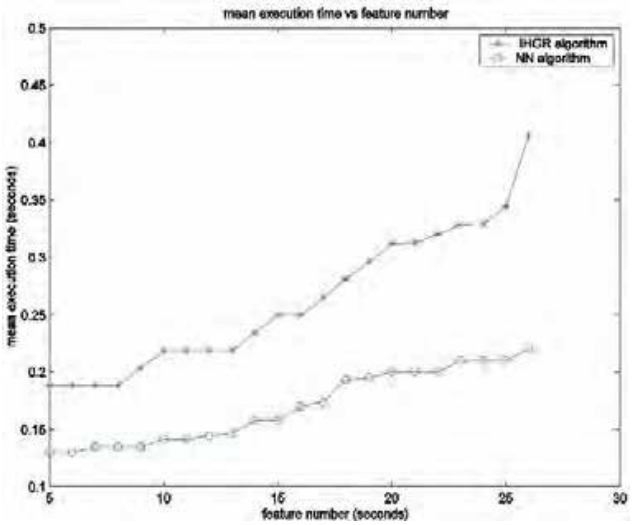


Fig. 4. The mean execution time of the IHGR algorithm and the NN algorithm. The mean execution time means the average CPU time used of 50 Monte Carlo runs for the process of the data association algorithm. The mean execution time of IHGR is nearly linear with respect to the observed feature number by repeated experiments.

	Errors(Meters)	
M	X direction error	Y direction error
2	0.2374	0.2518
5	0.1737	0.1946
8	0.1278	0.1469
10	0.1008	0.1348
13	0.0792	0.0986
15	0.0771	0.0910
18	0.0631	0.0874
20	0.0558	0.0806
25	0.0501	0.0645
30	0.0354	0.0439
35	0.0323	0.0357

Table 1. Comparison of the SLAM performance for various number of observations (M).

In Table 1, we show the average global errors of the vehicle's position estimate versus the number of features observed (observations) in each scan in the same environment mentioned above. We fixed the number of observations in each SLAM process which lasted for a few thousand steps but changed the number of observations for different SLAM processes. In the table, the errors are calculated as follows:

$$\text{X direction error}=\frac{1}{K}\sum_{i=1}^{K}\left|E_{x_i}\right|, \tag{11}$$

$$\text{Y direction error}=\frac{1}{K}\sum_{i=1}^{K}\left|E_{y_i}\right|, \tag{12}$$

where $\left|E_{x_i}\right|$ and $\left|E_{y_i}\right|$ are the absolute values of the i th time step vehicle position errors in the X and Y directions, respectively, and K is the total number of steps during SLAM.

It is found that the errors decrease when the number of observations increases. When the number of observations is more than 35, we found that the errors change very little. Therefore, we only show the results when the number of observations is smaller than 35.

In order to examine the robustness of the IHGR based algorithm with respect to sensor noise, we also carried out the simulations under various sensor range variances. From (Zhang et al., 2004a; Zhang et al., 2003; Adams et al., 2004), we know that for any range finder, the range noise covariance can vary depending on the received signal amplitude while the angular uncertainty is relatively very small and little changed compared to the range noise. In the simulation, we fix $\sigma_\theta = 0.0005$ radian. The sensor used here is a LADAR (Laser Detection and Ranging) sensor, therefore, the standard deviation of the range noise σ_r can be typically from 0.01 meters to 0.25 meters (Adams, 1999) for different sensors. Table 2 shows the

average performance of 10 SLAM processes for the same environments mentioned earlier with 12 observations.

In Subsection 3.3 we have compared the computational burden of our method with that of the JCBB algorithm (Neira & Tardos, 2001). Here we shall compare the accuracy of these two methods. Note that the JCBB algorithm which uses the branch and bound enumeration algorithm gives an optimal solution to the data association problem. In this simulation, we use 30 features in the large scale map because the speed of the JCBB algorithm becomes quite slow when the feature number is too large.

	Errors(Meters)	
σ_r	X direction error	Y direction error
0.01	0.0815	0.0979
0.02	0.0989	0.1135
0.03	0.1247	0.1389
0.05	0.2049	0.2209
0.08	0.2975	0.3108
0.1	0.3564	0.4532
0.15	0.4753	0.6786
0.2	0.6785	0.9860
0.25	0.9076	1.0062

Table. 2. SLAM performance versus the standard deviation of the sensor range noise.

	JCBB method		IHGR method	
M	$Error_x$	$Error_y$	$Error_x$	$Error_y$
5	0.1802	0.1899	0.1802	0.1899
8	0.1346	0.1573	0.1389	0.1623
10	0.1025	0.1389	0.1101	0.1475
13	0.0834	0.1003	0.0924	0.1120
15	0.0799	0.0915	0.0856	0.1079
18	0.0702	0.0832	0.0786	0.0886
20	0.0628	0.0784	0.0703	0.0826
25	0.0591	0.0748	0.0651	0.0799
30	0.0528	0.0537	0.0580	0.0722

Table 3. The comparison of the SLAM performance for the JCBB algorithm and IHGR algorithm under different number of features. The unit for the errors is meter.

From Table 3 one can see that with the IHGR based data association method a near optimal solution to SLAM has been achieved. Observe that when the number of features is 5, the same performance as the JCBB algorithm which gives the optimal solution is obtained in this experiment.

4.2 Real Outdoor Environment
4.2.1 SLAM with Artificial Beacon

In order to implement the IHGR data association algorithm during SLAM in a real environment, we firstly use the experimental data set from (Guivant & Nebot, 2003) obtained by Guivant and Nebot. The testing site is a car park at Sydney University. The vehicle is equipped with a GPS, a laser sensor and wheel encoders. A kinematic GPS system of 2 cm accuracy was used to evaluate the ground truth. Thus, the true navigation map was available for comparison purposes. Wheel encoders give an odometric measurement of the vehicle location. The dead reckoning sensors and laser range sensor are combined together to predict the vehicle's trajectory using the extended Kalman filter and to build up the map at the same time. In this experiment, the features used are artificial beacons. The feature detection was done by using a geometric analysis of the range measurement to obtain the most likely centers of tree trunks using intensity information. The laser scans are processed using Guivant's algorithm (Guivant & Nebot, 2000) to detect tree trunks' centers and estimate their radii.

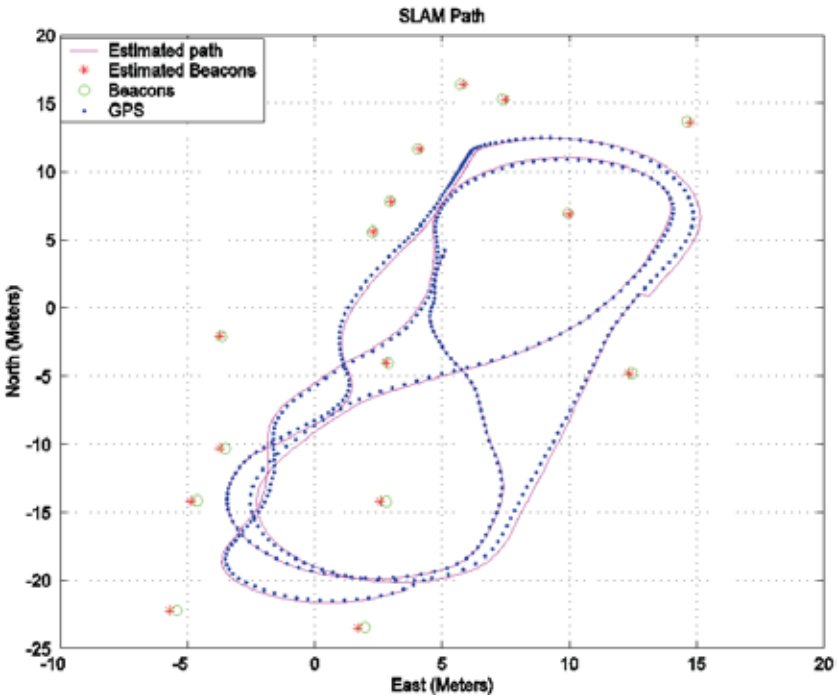


Fig. 5. The SLAM path and the feature map during the SLAM process with IHGR data association.

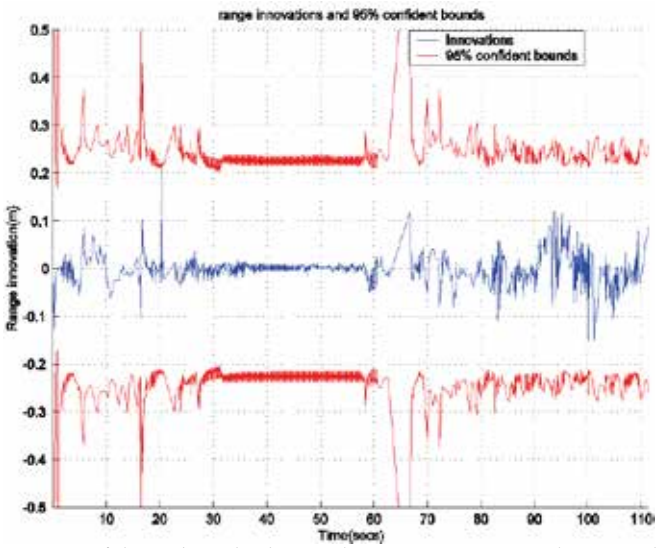


Fig. 6. The 2σ confidence bounds during the SLAM process with IHGR data range innovation and its association.

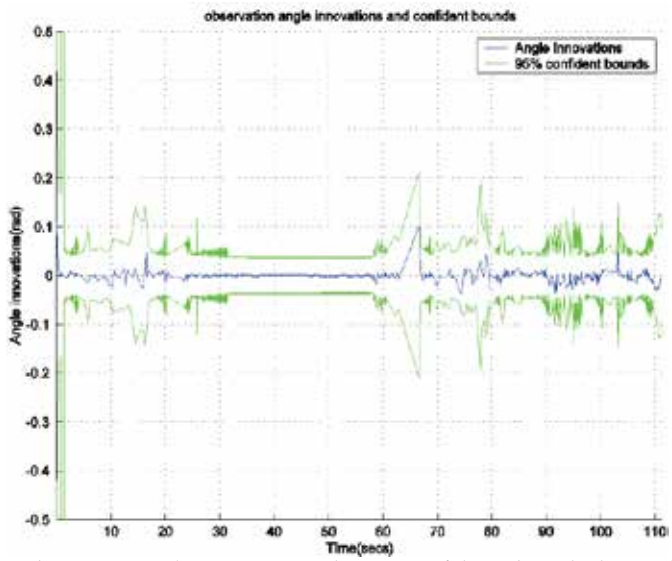


Fig. 7. The observation angle innovation and its 2σ confidence bounds during the SLAM process with IHGR data association.

We ran continuous SLAM for more than 5000 time steps and obtained the map shown in Figure 5 using our proposed IHGR data association method. It can be seen that the IHGR method performs well in real time. Figures 6 and 7 give the measurement (range and angle) innovation and their 95% confidence bounds during the SLAM process, respectively. The results show that the IHGR based algorithm works well during the SLAM process.
In order to check the effectiveness of the IHGR data association method, we randomly choose two scans (scan 68 and scan 87) and show the matching matrix x_{nm} after the IP problem is solved. In scan 68, the laser sensor obtained 2 measurements and the existing feature number has accumulated to 11. As mentioned, the term "measurement" means the extracted features. As seen in Table 4, measurement 1 is associated with feature 3 and measurement 2 is associated with feature 2. The rest of the features are all undetected in this scan. In Table 5, for scan 87, measurement 1 is matched with a dummy element which means this is a new feature or false alarm. Since the probability of false alarm for the laser sensor is low, we regard the measurement as a new feature. The other measurements and features are similar to those in scan 68 which can be seen in Table 4.

	Existed feature number and dummy element												
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**0**
scan 68	**1**	0	0	1	0	0	0	0	0	0	0	0	0
	2	0	1	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	1	1	1	1	1	1	1	1	0

Table 4. The matching matrix in scan 68.

		Existed feature number and dummy element														
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**0**
scan 87	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	0	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0

Table 5. The matching matrix in scan 87.

In this environment, the features are sparsely distributed. By checking the experimental data, we know that the number of observations per step ranges from 1 to 5. Therefore, the environment is relatively simple as far as data association is concerned. In this case, the NN algorithm also works well. We ran SLAM with the NN algorithm and IHGR algorithm respectively and found that the computational cost is similar although the NN algorithm is slightly faster than our method. For NN data association, the whole SLAM process (involving more than 5000 time steps), where the vehicle travelled several hundred meters, took 40.9680 seconds while the IHGR algorithm took 58.1420 seconds on the same computer (the algorithms were run on Pentium IV PC, 1.7GHz, and we calculated the CPU time for each algorithm).

4.2.2 SLAM with Natural Features in a real Environment

In order to verify the effectiveness of the IHGR algorithm, a more complex real environment is explored using our car-like Cycab vehicle (see Figure 9). In this case, SLAM is

implemented with natural features rather than artificial beacons in the last experiment. The features extracted from the complex campus environment are obtained by using the feature extraction algorithm given in (Zhang et al. 2004*a*). The experimental data used was obtained from a 2-D scanning range laser (SICK LMS200) mounted on the front of the vehicle. The laser returns a 180 degree planar sweep of range measurements in 0.5 degree intervals (i.e., 361 range values in an anti-clockwise order) with a range resolution of 50mm. Similar to the last experiment, the vehicle is equipped with a D-GPS, a laser sensor and encoders. The D-GPS system is used to evaluate the ground truth. Thus, the true navigation map is available for comparison purposes. The testing site is a long walk way around Hall 7 at Nanyang Technological University of Singapore. The real campus map, including this road, is shown in Figure 8. The dead reckoning sensors and laser range sensor are combined together to predict the vehicle's trajectory using the extended Kalman filter and to build up the map at the same time. During SLAM, in order to improve our map accuracy, we attempted to detect two types of features, namely the point features and circular features and used the IHGR algorithm to carry out data association.

Fig. 8. The map of our experiment site. The blue curve represents the vehicle's trajectory; The starting and ending of the trajectory are indicated by red dots.

Figure 9 shows part of the experimental environment. Figure 10 gives a typical laser scan during the run and shows the main features that are extracted from the scan data. Figure 11 shows the experimental results of SLAM. In this figure, there is a break in the GPS data. This is because the vehicle is under a building which blocks the GPS signal (the GPS data coordinate is vacant roughly between (-10, -120) and (-30, -115)). That is, no ground truth is available in this segment of road. From this figure, we can see that the estimated path is close to the GPS reading except the part of the road that is near to the break because the GPS reading is not accurate for this segment of the trajectory. Figures 12 and 13 indicate the observation innovations and their 95 % confidence bounds during the whole process which lasted for 450 seconds. These figures show that the SLAM process is consistent using the natural features. It is also important to mention that the average positional error of the entire SLAM process except the segment where the ground truth is unavailable is smaller than 0.5 meter. We also

carried out the SLAM process with the NN data association algorithm. Unfortunately, the NN algorithm results in a diverged vehicle path estimate as shown in Figure 14, due to the complex environment with features of high density. In Figure 14, point A is the starting point of the SLAM process as in the previous experiment with the IHGR algorithm.

Fig. 9. The environment and the vehicle in our experiment.

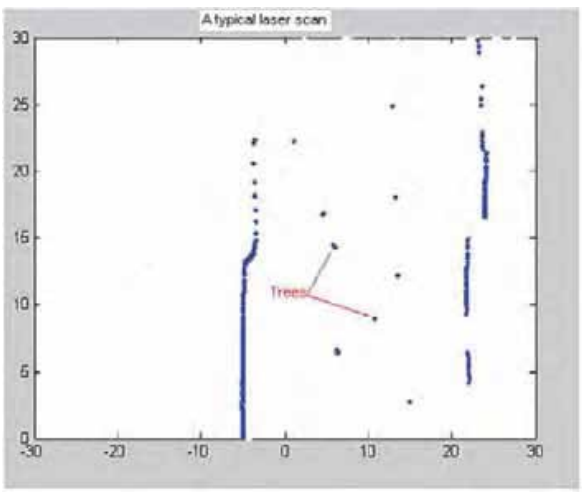


Fig. 10. A typical laser scan of the experimental environment.

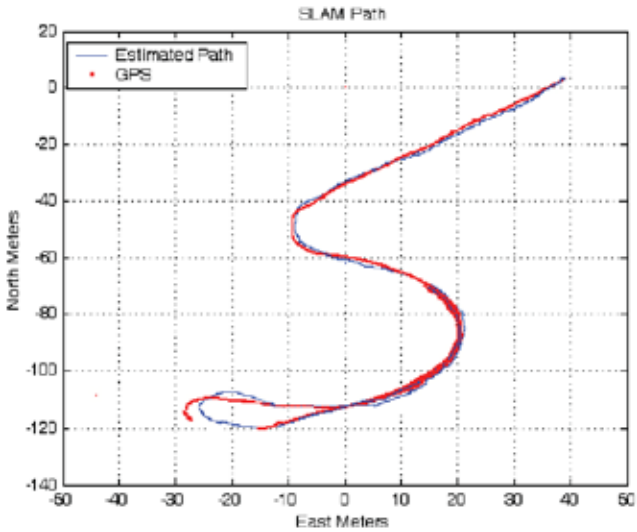


Fig. 11. A comparison between the estimated path and the path estimated by the D-GPS (used as ground truth) during SLAM.

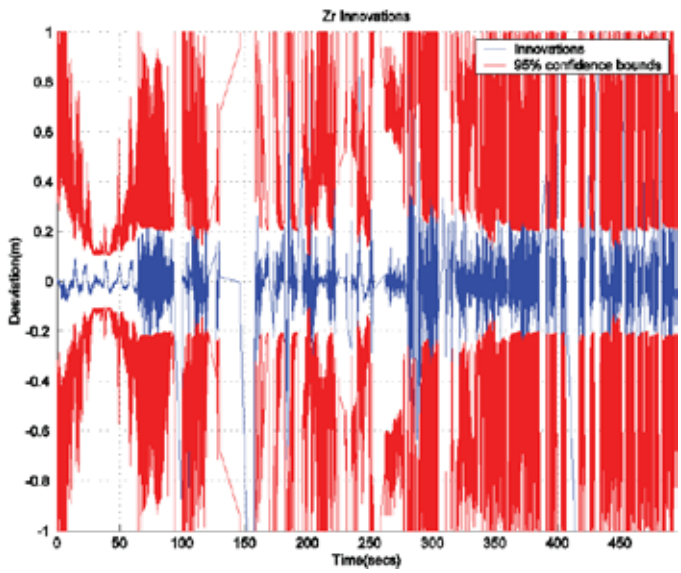


Fig. 12. The range observation's innovation and its 95% confidence bounds in the SLAM experiment.

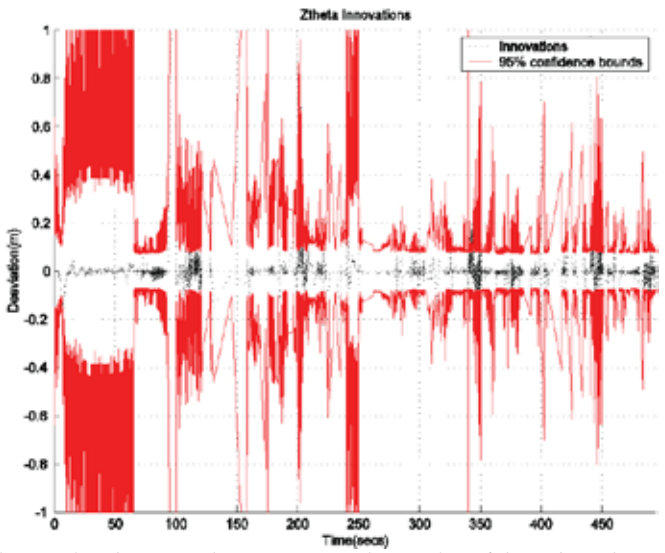


Fig. 13. The angular observation's innovation and its 95% confidence bounds in the SLAM experiment.

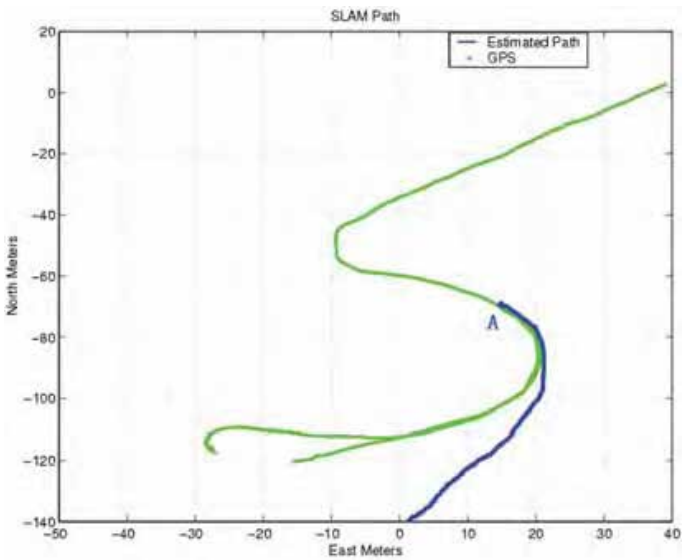


Fig. 14. The unsuccessful path estimation of this experiment when NN algorithm is applied. The vehicle started from point A in our experiment, see Figure 8.

5. Conclusions

This chapter presented a data association algorithm for SLAM which offers a good trade-off between accuracy and computational requirements. We first formulated the data association problem in SLAM as a two dimensional assignment problem. In our work, only 1 frame of scan data is considered. The data association problem in SLAM is formulated as a two dimensional assignment problem rather than a three dimensional one which is an NP hard problem and is computationally more efficient. Further, since only one step prediction is involved, the effect of the vehicle model uncertainty is smaller as compared to the data association methods using two frame scan data. In order to obtain a fast solution, the 0-1 IP problem was firstly relaxed to an LP problem. Then we proposed to use the IHGR procedure in conjunction with basic LP algorithms to obtain a feasible solution of the data association problem. Both the simulation and experiment results demonstrated that the proposed algorithm is implementable and gives a better performance (higher successful rate) than the commonly used NN algorithm for complex (outdoor) environments with high density of features. Compared to the optimal JCBB algorithm, the proposed algorithm has lower computational complexity and is more suitable for real-time implementation.

References

Adams, M. D. (1999). *Sensor Modeling, Design and Data Processing for Autonomous Navigation*, World Scientific, Singapore.

Adams, M. D., Zhang, S., and Xie, L. (2004). Paticle filter based outdoor robot localization using natural features extracted from laser scanner, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 854–859, New Orleans, USA, April, 2004.

Bailey, T., Nebot, E. M., Rosenblatt, J. K., and Durrant-Whyte, H. F. (2000). Data association for mobile robot navigation: a graph theoretic approach, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2512–2517, San Francisco, USA, May, 2000.

Bar-shalom, Y. and Fortmann, T. E. (1988). *Tracking and data association*, Acdamic Press.

Castellanos, J. A., Montiel, J. M., Neira, J., and Tard**o**s, J. D. (1999). The spmap: a probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation,* Vol. 15, No. 5, pp 948–953.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*, The MIT Press, London.

Franz, L. S., and Miller, J. L. (1993). Scheduling medical residents to rotations: solving the large scale multiperiod staff assignment problem. *Operations Research,* Vol. 41, No. 2, pp. 269–279.

Guivant, J., and Nebot, E. M. (2003). *ACFR, Experimental outdoor dataset*. http://www.acfr.usyd.edu.au /homepages/academic /enebot/dataset.htm.

Guivant, J., Nebot, E. M., and Durrant-Whyte, H. F. (2000). Simultaneous localization and map building using natural features in outdoor environments, *Proceedings of the IEEE International Conference on Intelligent Autonomous Systems*, Venice, Italy, pp. 581–588. Hochbaum, D. S. (1997). *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company, Boston.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica,* Vol. 4, No. 4, pp. 373–395.

Lim, J. H., and Leonard, J. J. (2000). Mobile robot relocation from echolocation constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 22, No. 9, pp. 1035–1041.

Miller, J. L., and Franz, L. S. (1996). A binary rounding heuristic for multi-period variable-task duration assignment problems. *Computers and Operations Research*, Vol. 23, No. 8, pp. 819–828.

Morefield, C. L. (1977) Application of 0-1 integer programming to multitarget tracking problems, *IEEE Transactions on Automatic Control*, Vol. AC-22, No. 3, pp. 302-312.

Neira, J., and Tard**o**s, J. D. (2001). Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 6, pp. 890–897.

Nieto, J., Guivant, J., Nebot, E., and Thrun, S. (2003). Real time data association for fastslam, *Proceedings of the IEEE International Conference on Robotics and Automation*. pp. 2512–2517, Taiwan, May, 2003.

Parker, R. G., and Rardin, R. L. (1988). *Discrete optimization*, Acdamic Press, New York.

Poore, A. B. (1994). Multidimensional assignment formulation of data association problems arising from multitarget tracking and multisensor data fusion. *Computational Optimization and Applications,* July 1994, Vol. 3, pp.27-57.

Perera, D. L., Wijesoma, S., and Adams, M. D. (2003). Multi-dimensional assignment based data association for simultaneous localization and mapping for autonomous vehicles, *Proceedings of CIRAS*, pp. 1450–1456, Singapore, December, 2003

Perera, D. L., Wijesoma, S., and Adams, M. D. (2004). On multidimensional assignment data association for simultaneous localization and mapping, *Proc. of IEEE Int. Conf. Robot. Automat.,* pp. 860–865, New Orleans, USA, April, 2004.

Poore, A. B., and Robertson, A. J. (1995). A new multidimensional data association algorithms for class of multisensor multitarget tracking, *Proceedings of SPIE,* pp. 448-459, July, 1995.

Poore, A. B., and Robertson, A. J. (1997). A new class of langrangian relaxation based algorithms for a class of multidimensional assignment problems. *Computational Optimization and Applications*, May, 1997, Vol. 8, No. 2, pp. 129–150.

Storms, P. P. A., and Spieksma, F. C. R. (2003). An lp-based algorithm for the data association problem in multitarget tracking. *Computers and Operation Research,* Vol. 30, No. 7, pp.1067–1085.

Vajda, S. 1981. *Linear programming: algorithms and applications*, Chapman and Hall, USA.

Zhang, S., Xie, L., and Adams, M. D. (2003). Geometrical feature extraction using 2d range scanner, *Proceedings of the Fourth International Conference on Control and Automation*. pp. 901–905, Montreal, Canada, June 2003.

Zhang, S., Xie, L., and Adams, M. D. (2004*a*). Gradient model based feature extraction for simultaneous localization and mapping in outdoor applications, *Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004)*, pp.431-436, Kunming, China, Dec. 2004.

Zhang, S., Xie, L., and Adams, M. D. (2004*b*). An efficient data association approach to simultaneous localization and map building, *Proc. of IEEE Int. Conf. Robot. Automat.*, pp. 1493–1498, New Orleans, USA, April 2004.

20

A Generalized Robot Path Planning Approach Without The Cspace Calculation

Yongji Wang[1], Matthew Cartmell[2] , QingWang[1] , Qiuming Tao[1]
[1]*State Key Laboratory of Computer Science and Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing, China*
[2]*Department of Mechanical Engineering, University of Glasgow, Glasgow, G12 8QQ, U.K.*

1. Introduction

One of the ultimate goals of robotics is to create autonomous robots. Such robots could accept high level instructions and carry out tasks without further human supervision or intervention. High level input commands would specify a desired task and the autonomous robot would compute how to complete the task itself. Progress towards autonomous robots is of great research and practical interest, with possible applications in any environment hostile to humans. Examples are underwater work, space exploration, waste management and bomb disposal among many others. One of the key technical issues towards such an autonomous robot is the Path Planning Problem (PPP): How can a robot decide what paths to follow to achieve its task. The PPP can be described as follows: given a robot with an initial configuration, a goal configuration, its shape and a set of obstacles located in the workspace, find a collision-free path from the initial configuration to the goal configuration for it.

PPP has been an active field during the past thirty years. Although seemingly trivial, it has proved notoriously difficult to find techniques which work efficiently, especially in the presence of multiple obstacles. A significant and varied effort has been made on this complicated problem (Wang et al., 2005; Wang et al., 2004; Wang & Lane, 2000; Wang & Lane, 1997; Wang, 1995; Wang, 1997; Wang et al., 2000; Wang & Cartmell, 1998c; Wang & Cartmell, 1998a; Wang & Cartmell, 1998b; Wang & Linnett, 1995; Wang et al., 1994a; Wang et al., 1994b; Petillot et al., 1998; Petillot et al., 2001; Park et al., 2002; Ruiz et al., 1999; Trucco et al., 2000; Brooks & Lozano-Perez, 1985; Conn & Kam, 1997; Connolly, 1997; Hu et al., 1993; Huang & Lee, 1992; Hwang & Ahuja, 1992; Khatib, 1986; Latombe, 1991; Lozano-Perez, 1983; Lu & Yeh, 2002; Lumelsky, 1991; Oriolo et al., 1998; Xu & Ma, 1999; Zhang & Valavanis, 1997). Various methods for dealing with the basic find-path problem and its extensions, such as Vgraph, Voronoi diagram, exact cell decomposition, approximate cell decomposition, potential field approach, and optimization-based approach have been developed. A systematic discussion on these old methods can be found in references (Wang et al., 2005; Wang, 1995; Latombe, 1991).

In any robot path planning method, robot and obstacle representation is the first thing to be

considered. PPP essentially deals with how to find a collision_free path for a 3 Dimensional (3D) object (robot) moving among another set of 3D objects (obstacles), satisfying various constraints (Wang et al., 2005; Wang et al., 2004; Wang & Lane, 2000; Wang & Lane, 1997; Wang, 1995; Wang, 1997; Wang et al., 2000). There are many reasons for having so many different approaches developed. For example, assumptions made on both the shapes of the robot and obstacles and the constraints imposed by the mechanical structure of the robot contribute to them (Wang, 1997). The important thing for judging the reality of an approach is whether the realistic constraints have been considered.

An important concept proposed in the early stage of robot path planning field is the shrinking of the robot to a point and meanwhile the expanding of the obstacles in the workspace as a set of new obstacles. The resulting grown obstacles are called the Configuration Space (Cspace) obstacles. The find-path problem is then transformed into that of finding a collision-free path for a point robot among the Cspace obstacles. This idea was first popularized by Lozano-Perez (Lozano-Perez, 1983) in the Artificial Intelligence Laboratory, MIT as a basis of the spatial planning approach for the find-path and find-place problems, and then extended by Latombe (Latombe, 1991) as a basis for all motion planning approaches suitable for a point robot. However, the research experiences obtained so far have shown that the calculation of Cspace obstacles is very hard in 2D when the following situations occur. 1. Both the robot and obstacles are not polygons; and 2. The robot is allowed to rotate. The situation gets even worse when the robot and obstacles are 3D objects with various shapes (Ricci, 1973; Blechschmidt & Nagasuru, 1990; Barr, 1981; Chiyokura, 1988). For this reason, direct path planning approaches without the Cspace calculation is quite useful and expected.

The objective of this chapter is to present a new approach to the PPP without the Cspace calculation. The chapter is based on our previous work (Wang et al., 2005; Wang et al., 2004; Wang & Lane, 2000; Wang & Lane, 1997), and in the following we will present the background of the new method to show its principle.

Historically the Constrained Optimization and Constructive Solid Geometry (COCSG) method is first proposed in (Wang & Lane, 1997), and two assumptions made in it are that: 1. The Cspace obstacles in the workspace can be approximately represented by inequalities; and 2. The robot can be treated as a point. The mathematical foundations for the Constructive Solid Geometry (CSG), the Boolean operations, and the approximation techniques are developed to represent the free space of the robot as a set of inequalities (Ricci, 1973; Wang & Lane, 1997). The fundamental ideas used include: 1. The free Cspace of the robot is represented as a set of inequality constraints using configuration variables; 2. The goal configuration is designed as the unique global minimum point of the objective function, and the initial configuration is treated as the start point for the spatial search; and 3. The numerical algorithm developed for solving nonlinear programming problem is applied to solve the robot motion planning problem and every immediate point generated in this way guarantees that it is in the free space, and therefore is collision free. The contribution of the above paper is that for the first time, the idea of inequality is introduced to represent objects and the optimization technique is used for the efficient search.

However, we can still observe that two issues arise from the above problem formulation. One is how to exactly rather than approximately deal with the shapes of both the robot and the obstacles, and the other is how to calculate the Cspace obstacles. In reference (Wang &

Lane, 2000), we further investigate the effect of obstacle shapes on the problem formulation, and introduce the new concept of the first and second kinds of obstacles. When the second kind of obstacles is considered, the PPP leads to a generalized constrained optimization problem (GCOP) with both logic AND and OR relationships, which is totally different from the traditional standard constrained optimization problem with only logic AND relationship among the constraints. A mathematical transformation technique is developed to solve the GCOP. The original contributions of this paper include threefold: First, from the viewpoint of optimization theory, it is the first one to propose such a GCOP; Second, a method is developed to solve such a GCOP; Third, from the viewpoint of PPP, this paper inherits the advantage of the previous method in (Wang & Lane, 1997) and further generalizes its ability to deal with various shapes of obstacles.

The issue that has not been addressed by the above two papers is the calculation of the Cspace obstacles. We deal with the PPP with the first kind of obstacles in (Wang et al., 2004) and the second kind of obstacles in (Wang et al., 2005) respectively, without the need to calculate the Cspace obstacles. A sufficient and necessary condition for a collision free path for the robot and the obstacles is then derived in the form of a set of inequalities that lead to the use of efficient search algorithms. The principle is that the points outside the obstacles in the 3D workspace are represented by implicit inequalities, the points on the boundary of a 3D robot are expressed in the form of a parametric function, and the PPP is formulated as a semi-infinite constrained optimization problem with the help of the mathematical transformation. To show its merits, simulation results with different shapes of robot and obstacles in 3D space are presented.

In this chapter we will present a comprehensive introduction to the principle of the PPP without the Cspace calculation, including the mathematical background, robot and obstacle representation, sufficient and necessary condition for collision-free path, algorithm efficiency, and the simulation results. Particularly, we will also discuss the constraints that must be considered in the future work and explain mathematically the reason why these constraints can lead to more difficulties in this area.

The rest of the chapter is organized as follows. Section 2 gives a brief description of inequality constraints and the formulations for optimization theory. In particular, a previously-developed, generalized constrained optimization and the mathematical translation needed for its solution are also presented in this section. In Section 3, obstacle and robot presentation method is presented. The implicit function inequalities for representing the outside of the obstacles and the parametric function equalities for representing the surface points of 3D robot are developed. In Section 4, we investigate how to convert the robot path planning problem into a semi-infinite constrained optimization problem. Simulation results are presented in Section 5. Finally conclusions are given in Section 6.

2. Mathematical Background

In this section we will give a brief introduction to various optimization problems, i.e. the standard constrained optimization problem (SCO), generalized constrained optimization problem (GCO), and semi-infinite constrained problem (SICO). The essential part of the mathematical transformation which can transfer a set of inequalities with logic OR operations into one inequality is also introduced in subsection 2.4. Details of the nonlinear

programming theory can be found in (Fletcher, 1987; Gill et al., 1981; Luenberger, 1984; Polak & Mayne, 1984; Rao, 1984; Tanak et al., 1988).

2.1 Optimization Problems

Standard optimization theory (SOT) concerns the minimization or maximization of a function subject to different types of constraints (equality or inequality) (Fletcher, 1987; Gill et al., 1981). There are mainly four different types of optimization problem: *Linear Programming*, *Unconstrained Problems*, *Constrained Problems* and *Semi-infinite Constrained Problems*, as listed in Table 1. The last three parts together comprise the subject of *Non-linear Programming*.

TYPE	NOTATION
Unconstrained scalar	min $f(\boldsymbol{x})$
Unconstrained	min $f(\boldsymbol{x})$
Constrained	min $f(\boldsymbol{x})$ such that $g(\boldsymbol{x}) \le 0$
Goal	min $\boldsymbol{\gamma}$ such that $f(\boldsymbol{x})$-$\boldsymbol{x}^{\boldsymbol{\gamma}} \le \boldsymbol{Goal}$
Minmax	min{max $f(\boldsymbol{x})$} such that $g(\boldsymbol{x}) \le 0$
Nonlinear least squares	min $\sum\{ f(\boldsymbol{x})^* f(\boldsymbol{x})\}$
Nonlinear equations	$f(\boldsymbol{x})=0$
Semi-infinite constrained	min $f(\boldsymbol{x})$ such that $g(\boldsymbol{x}) \le 0$ & $\Phi(\boldsymbol{x},\boldsymbol{w}) \le 0$ for all $\boldsymbol{w} \in \Re^2$

Table 1. Types of nonlinear minimization problems.

2.2 Standard Constrained Optimization (SCO)

The standard constrained optimization problem can be described as follows: find an optimal point $\boldsymbol{x}^*$ which minimizes the function:

$$f(\boldsymbol{x}) \tag{1}$$

subject to:

$$g_i(\boldsymbol{x}) = 0,\ i = 1, 2, \ldots, t$$
$$g_j(\boldsymbol{x}) \le 0,\ j = t+1, t+2, \ldots, s$$
$$\boldsymbol{x}_L \le \boldsymbol{x} \le \boldsymbol{x}_U \tag{2}$$

where t and s are positive integers and $s \ge t$, $\boldsymbol{x}$ is an n-dimensional vector of the unknowns $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, and f, g_i $(i = 1, 2, \ldots, t)$ and g_j $(j = t+1, t+2, \ldots, s)$ are real-valued functions of the variables $(x_1, x_2, \ldots, x_n)$. $\boldsymbol{x}_L = (L_1, L_2, \ldots, L_n)$ and $\boldsymbol{x}_U = (U_1, U_2, \ldots, U_n)$ are the lower and upper bounds of $\boldsymbol{x}$, respectively. The function f is the objective function, and the equations and inequalities of (2) are constraints.

It is important to note that although not explicitly stated in the literature available, the logic relationship among the constraints (equalities and inequalities) in (2) are ***logic AND*** (denoted by "$\wedge$"). That is, constraints in (2) can be presented explicitly as:

$$g_1(\boldsymbol{x})=0 \wedge g_2(\boldsymbol{x})=0 \wedge \ldots \wedge g_t(\boldsymbol{x})=0$$
$$\wedge\, g_{t+1}(\boldsymbol{x}) \le 0 \wedge g_{t+2}(\boldsymbol{x}) \le 0 \wedge \ldots \wedge g_s(\boldsymbol{x}) \le 0 \tag{3}$$
$$\wedge\, L_1 \le x_1 \le U_1 \wedge L_2 \le x_2 \le U_2 \wedge \ldots \wedge L_n \le x_n \le U_n.$$

Problem described by (1) and (2) is named as the standard constrained optimization problem (SCOP).

2.3 Generalized Constrained Optimization (GCO)

The work reported in (Wang & Lane, 2000) has shown that some realistic problem can be cast as a generalized constrained optimization problem of the following form:
Find an optimal point $\boldsymbol{x}^*$ which minimizes the function

$$f(\boldsymbol{x}) \tag{4}$$

subject to:

$$\begin{gathered} g_1(\boldsymbol{x})=0 \wedge g_2(\boldsymbol{x})=0 \wedge \ldots \wedge g_t(\boldsymbol{x})=0 \\ \wedge g_{t+1}(\boldsymbol{x})\le 0 \wedge g_{t+2}(\boldsymbol{x})\le 0 \ \ldots \wedge g_s(\boldsymbol{x})\le 0 \\ \wedge (h_{1,1}(\boldsymbol{x})\le 0 \vee h_{1,2}(\boldsymbol{x})\le 0 \vee \ldots \vee h_{1,k_1}(\boldsymbol{x})\le 0) \\ \wedge (h_{2,1}(\boldsymbol{x})\le 0 \vee h_{2,2}(\boldsymbol{x})\le 0 \vee \ldots \vee h_{2,k_2}(\boldsymbol{x})\le 0) \\ \wedge \ldots \\ \wedge (h_{m,1}(\boldsymbol{x})\le 0 \vee h_{m,2}(\boldsymbol{x})\le 0 \vee \ldots \vee h_{m,k_m}(\boldsymbol{x})\le 0) \\ \wedge L_1\le x_1\le U_1 \wedge L_2\le x_2\le U_2 \wedge \ldots \wedge L_n\le x_n\le U_n \end{gathered} \tag{5}$$

where, the symbol "$\vee$" denotes the ***logic OR*** relationship, $t, s, m, k_1, k_2, \ldots, k_m$ are all positive integers, and $h_{i,j}(\boldsymbol{x})$, $(i=1,2,\ldots m;\ j=1,2,\ldots k_i)$, are real-valued functions of the variables $\boldsymbol{x}$. The problem described by (4) and (5) is named as the generalized constrained optimization problem (GCOP) because the constraints have both ***logic AND*** and ***logic OR*** relationships.
The development of an algorithm for the solution to GCOP is important. There are two ways to deal with the difficulty. The first is to develop some new algorithms which can directly deal with the GCOP rather than adopting the algorithms available for the SCOP. The second way is based on the idea of devising a mathematical transformation which is able to convert each constraint: $h_{i,1}(\boldsymbol{x})\le 0 \vee h_{i,2}(\boldsymbol{x})\le 0 \vee \ldots \vee h_{i,k_i}(\boldsymbol{x})\le 0$ ($i=1, 2, \ldots, m$) into *one* new inequality $H_i(\boldsymbol{x})\le 0$, $i=1, 2, \ldots, m$, for any point $\boldsymbol{x}$. As a result, the algorithms developed for the SCOP can be directly applied to the GCOP.

2.4 A Mathematical Solution to Converting a Set of Inequalities with Logic OR Relation into One Inequality

Here, we present a mathematical transformation which is able to realize the second idea in subsection 2.3. Suppose there are m inequalities $h_i(\boldsymbol{x})<0$, $i=1,2,\ldots,m$, with ***Logic AND*** defined as set A in (6). From a mathematical viewpoint, set A represents the point set of the inside for a generalized n dimensional object, and its complement $\overline{A}$ represents the point set of the outside and boundary of the object. In a 3D space, set definition (6) may be explained as representing the set of all the inside points for an object whose surface is mathematically represented by m continuous equations $h_i(\boldsymbol{x})=0$ $(i=1, 2,\ldots, m)$.

$$A = \{\, \boldsymbol{x} \mid h_1(\boldsymbol{x})<0 \wedge \ldots\ldots \wedge h_m(\boldsymbol{x})<0 \,\} \tag{6}$$

$$\overline{A} = \{\, \boldsymbol{x} \mid h_1(\boldsymbol{x})\ge 0 \vee h_2(\boldsymbol{x})\ge 0 \vee \ldots \vee h_m(\boldsymbol{x})\ge 0 \,\} \tag{7}$$

For each function $h_i(\boldsymbol{x})$, $(i=1, 2, \ldots, m)$, a new function of the following form is constructed ($\boldsymbol{x}$ is omitted for simplicity):

$$v_i=(h_i^2+t^2)^{1/2}+h_i \qquad i=1, 2, \ldots, m \tag{8}$$

where t is a small, positive real number and satisfies $t<<1$. Note that v_i is the function of a point $\boldsymbol{x}$ and t. For the whole object, a function V of the following form is also constructed:

$$V = v_1 + v_2 + \ldots + v_m = \sum_{i=1}^{m} v_i \tag{9}$$

Now let us examine the properties of the two transformations from h_i to v_i and from v_i to V. First, function v_i is **always positive** for any point $\boldsymbol{x}$ and any constant t, i.e., $v_i>0$ always holds, and second, it is an **increasing function** of h_i, which suggests that the value of v_i at the points where $h_i>0$ is much larger than the value at the points where $h_i<0$. If $t<<1$, v_i can be approximately represented as

$$v_i = \begin{cases} 2h_i + O(t^2) >> t > 0, & h_i > 0; \\ \approx t, & h_i = 0; \\ \approx O(t^2), & h_i < 0. \end{cases} \quad i=1, 2, \ldots, m \tag{10}$$

where $O(t^2)$ represents a very small positive number with the order of t^2 for $t<<1$. (10) indicates that except for the points located at the vicinity of the surface $h_i=0$, v_i is large compared with t when $h_i>0$, and small compared with t when $h_i<0$.

From Fig. 1 we can see that for the points located inside the object and in the vicinity of $h_i=0$, the value of all other functions h_j $(j=1,2,\ldots,m \text{ and } j\neq i)$ is less than zero. This leads to v_i in the order of $O(t^2)$. Substituting (10) into (9) gives

$$V = \begin{cases} >> t, & for \quad (h_1 > 0) \vee (h_2 > 0) \vee \ldots \vee (h_m > 0); \\ \approx t + O(t^2), & for \quad (h_1 = 0 \wedge h_2 \leq 0 \wedge h_3 \leq 0 \wedge \ldots \wedge h_m \leq 0) \vee \\ & \quad (h_2 = 0 \wedge h_1 \leq 0 \wedge h_3 \leq 0 \wedge \ldots \wedge h_m \leq 0) \vee \\ & \quad \ldots \vee (h_m = 0 \wedge h_1 \leq 0 \wedge h_2 \leq 0 \wedge \ldots \wedge h_{m-1} \leq 0); \\ \approx O(t^2), & for \quad (h_1 < 0) \wedge (h_2 < 0) \wedge \ldots \wedge (h_m < 0). \end{cases} \tag{11}$$

Consequently, from (11), (6), and (7) we can observe that: function V is small, $\approx O(t^2)$, compared with t, when all the h_i are sufficiently negative, i.e. at those points which are inside the object; $V>>t+O(t^2)$ at the set of outside points of the object where at least one of the h_i is greater than t; and $V\approx t$ in the vicinity of the boundaries of the object. Fig. 1 illustrates this situation.

Now let us consider the situation when $t\to 0$ to have a better understanding why construction functions (8) and (9) are used as the mathematical transformation. As $t\to 0$, v_i tends to be

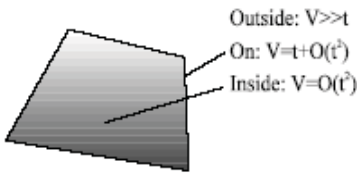


Fig. 1. Illustration of V as a function of point $\boldsymbol{x}$ in n-dimensional space.

$$v_i \begin{cases} > 0, & h_i > 0; \\ = 0, & h_i \le 0. \end{cases} \qquad i=1, 2,\ldots, m \tag{12}$$

It is well-known that the sum of two positive values is positive, the sum of a positive value and a zero is positive, and the sum of two zeroes is zero. Thus the addition operation of v_i in (9) corresponds to the ***Logic OR*** if we treat a positive value as a logic value "1" and a zero as a logic value "0". Thus we have

$$V \begin{cases} > 0, & h_i > 0, \quad for \quad some \quad i \in \{1,2,...,m\}; \\ = 0, & h_i \le 0, \quad for \quad all \quad i \in \{1,2,...,m\}. \end{cases} \tag{13}$$

This property indicates that when t=0 the **sufficient and necessary condition** for a point $\boldsymbol{x}$ which falls into the outside of the object is that

$$V = \sum_{i=1}^{m} v_i > 0 \tag{14}$$

Note that the standard form for constraints in optimization problem (1) and (2) is **less than or equal to** zero. Note that although (14) may change to the form (15), it does not allow the condition "equal to zero".

$$-V = -\sum_{i=1}^{m} v_i < 0 \tag{15}$$

In fact, the "equal to zero" case means the point lies on the boundary of the object. However, it is not desirable for robot path planning to have the path too close to the obstacles. Thus a small positive value Δv can be introduced to control the distance of the feasible path to the obstacle. If the following inequality is satisfied by a point $\boldsymbol{x}$

$$V = \sum_{i=1}^{m} v_i \ge \Delta v \quad \text{or} \quad \Delta v - \sum_{i=1}^{m} v_i \le 0 \tag{16}$$

then this point must be outside the obstacle determined by (6). If $\Delta v \to 0$, the boundary determined by $\Delta v - \sum_{i=1}^{m} v_i \le 0$ tends to be the surface of the obstacle.

In summary, we have the following result.

Theorem 1: *If the outside and the surface of an object is determined by* ($h_1 \ge 0 \vee h_2 \ge 0 \vee \ldots \vee h_m \ge 0$), *then its outside and surface can also be determined by the inequality* $\Delta v - \sum_{i=1}^{m} v_i \le 0$ *as the small positive value* $\Delta v \to 0$. *In other words, the satisfaction of the inequality* $\Delta v - \sum_{i=1}^{m} v_i \le 0$ *for a point* $\boldsymbol{x}$ *guarantees that this point falls outside the object.*

A direct conclusion drawn from **Theorem 1** is that a GCO problem can be converted into an SCO problem by the transformations (8) and (9).

2.5 Semi-Infinite Constrained Optimization (SICO)

The semi-infinite constrained optimization problem is to find the minimum of a semi-infinitely constrained scalar function of several variables $\boldsymbol{x}$ starting at an initial estimate $\boldsymbol{x}_s$. This problem is mathematically stated as:

Minimize

$$f(\boldsymbol{x}),\ \boldsymbol{x} \in \Re^n, \tag{17}$$

subject to:

$$g_i(\boldsymbol{x}) = 0,\ i = 1, 2, \ldots, t$$
$$g_j(\boldsymbol{x}) \le 0,\ j = t+1, t+2, \ldots, s$$
$$\Phi_k(\boldsymbol{x}, \boldsymbol{v}) \le 0,\ k = 1, 2, \ldots, r$$
$$\boldsymbol{x}_L \le \boldsymbol{x} \le \boldsymbol{x}_U, \text{ for all } \boldsymbol{v} \in \Re^2 \tag{18}$$

where $\Phi_k(\boldsymbol{x}, \boldsymbol{v})$ is a continuous function of both $\boldsymbol{x}$ and an additional set of variables $\boldsymbol{v}$. The variables $\boldsymbol{v}$ are vectors of at most length two. The aim is to minimize $f(\boldsymbol{x})$ so that the constraints hold for all possible values of $\Phi_k(\boldsymbol{x}, \boldsymbol{v})$. Since it is impossible to calculate all possible values of $\Phi_k(\boldsymbol{x}, \boldsymbol{v})$, a region, over which to calculate an appropriately sampled set of values, must be chosen for $\boldsymbol{v}$. $\boldsymbol{x}$ is referred to as the unknown variable and $\boldsymbol{v}$ as the independent variables.

The procedure for solving such an SICO with nonlinear constraints is as follows:

(a) Assign an initial point for $\boldsymbol{x}$ and a region for $\boldsymbol{v}$;

(b) Apply a search algorithm to find the optimum solution $\boldsymbol{x}^*$ and the corresponding minimum objective function $f(\boldsymbol{x}^*)$.

In the subsequent sections we will gradually illustrate that the 3D path planning problem without the calculation of Cspace obstacles can be converted into a standard semi-infinite constrained optimization problem.

3. Obstacle and Robot Representations

For robot path planning, the first thing is to give each of the objects a mathematical representation, including obstacles and robot in the workspace. Modeling and manipulation of objects is the research task of Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), and Computer Graphics (CG) (Ricci, 1973; Blechschmidt & Nagasuru, 1990; Barr, 1981; Chiyokura, 1988; Hall & Warren, 1990; Berger, 1986; Comba, 1968; Franklin & Barr, 1981). A solid model should contain an informationally complete description of the geometry and topology of a 3D object (Blechschmidt & Nagasuru, 1990). A successful modeling system, in addition to many other features, must be capable of representing the object's surface and be able to unambiguously determine whether a point is in the "inside" or "outside" of the object. In CAD, CAM, and CG, there are three traditional categories of solid modeling systems, namely boundary representation (B-rep), spatial decomposition, and constructive solid geometry (CSG) (Chiyokura, 1988). In our method, two different categories of obstacles are distinguished, and CSG together with an approximation approach are used to represent the various objects in real world in form of inequality constraints.

3.1 General Representation of Obstacle and Classification

A 3D object $\boldsymbol{S}$ divides the 3D Euclidean space $\boldsymbol{E}^3$ into three parts: the **inside** of the object (denoted by $\boldsymbol{I}$), the **outside** of the object (denoted by $\boldsymbol{T}$), and the **boundary** (denoted by $\boldsymbol{B}$), with

$$\boldsymbol{I} \cup \boldsymbol{B} \cup \boldsymbol{T} = \boldsymbol{E}^3 \tag{19}$$

$$\boldsymbol{I} \cap \boldsymbol{B} = \boldsymbol{B} \cap \boldsymbol{T} = \boldsymbol{I} \cap \mathrm{T} = \Phi \tag{20}$$

Let $\boldsymbol{x}=(x, y, z) \in \boldsymbol{E}^3$ denote a **point** in 3D space. An obstacle can be described as a set of all those points that fall into the inside of the obstacle, that is, an obstacle **A** can be described as:

$$A = \{\, \boldsymbol{x} \mid \boldsymbol{x} \text{ falls into the } \textbf{inside} \text{ of } \textbf{A} \,\} \tag{21}$$

Based on this set-formed representation, we can define an important concept "**free space of an obstacle**" and get a basic condition for a collision-free point.

Definition 1: Free space of an obstacle: The set of points on or outside of the surface of a 3D obstacle is defined as its free space. That is, the **free space** of an obstacle A (set-formed representation) is just $\overline{A}$, i.e., the complement of set A.

Proposition 1: *The necessary and sufficient condition for a point* $\boldsymbol{x}$ *to be collision-free from an obstacle A is that the point* $\boldsymbol{x}$ *must fall into the free space of A, that is,* $\boldsymbol{x} \in \overline{A}$.

The inside of a 3D object can be mathematically represented by one or several united implicit function inequalities. According to the number of the inequalities, we categorize 3D obstacles into two groups.

Definition 2: First group of obstacles: If the inside of a obstacle can be represented by only ***one*** implicit function inequality, the obstacle is said to be in the first group. That is, if an obstacle A can be represented as:

$$A=\{\, \boldsymbol{x} \mid h(\boldsymbol{x}) < 0 \,\} \tag{22}$$

where $h(\boldsymbol{x})$ is an implicit function of $\boldsymbol{x}$, then A belongs to the first group of obstacles. Obviously the free space of A can be represented as the following:

$$\overline{A} = \{\, \boldsymbol{x} \mid h(\boldsymbol{x}) \geq 0 \,\} \tag{23}$$

Examples of the obstacles in the first group include spheres, ellipsoids, torus, superellipsoids and so on (Ricci, 1973; Blechschmidt & Nagasuru, 1990; Barr, 1981; Berger, 1986; Franklin & Barr, 1981; Wang & Lane, 1997). A simple example of the first-group obstacles is illustrated in Fig. 2.

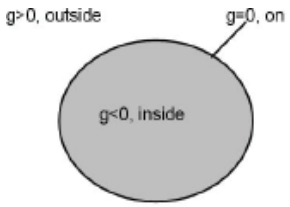


Fig. 2. First group of obstacles: inside and outside of an obstacle.

Definition 3: Second group of obstacles: If the inside of an obstacle must be represented by ***more than one*** united implicit function inequalities, the obstacle is said to be in the second group. That is, if an obstacle A can be represented as:

$$A =\{\, \boldsymbol{x} \mid h_1(\boldsymbol{x})<0 \wedge h_2(\boldsymbol{x})<0 \wedge \ldots \wedge h_m(\boldsymbol{x})<0 \,\} \tag{24}$$

where $h_i(\boldsymbol{x})$, $i=1, 2, \ldots, m$, are all implicit functions of $\boldsymbol{x}$ and m is more than one, then A belongs to the second group of obstacles.

For the second-group obstacle A, the free space can be represented as the following:

$$\overline{A} = \{\, \boldsymbol{x} \mid h_1(\boldsymbol{x}) \geq 0 \vee h_2(\boldsymbol{x}) \geq 0 \vee \ldots \vee h_m(\boldsymbol{x}) \geq 0 \,\} \tag{25}$$

For example, in Fig. 3, the 2-demensioned obstacle is a rectangle whose inside is surrounded by four lines $h_1 = x\text{-}a = 0$, $h_2 = -x\text{-}a = 0$, $h_3 = y\text{-}b = 0$, and $h_4 = -y\text{-}b = 0$, where a and b are positive values. The inside of the obstacle (denoted as A) is the intersection of regions A_i, $i=1, 2, 3, 4$, where each A_i is defined as $A_i = \{\, (x, y) \mid h_i < 0 \,\}$, that is:

$$\begin{aligned} A &= A_1 \cap A_2 \cap A_3 \cap A_4 \\ &= \{\, (x, y) \mid h_1 < 0 \,\} \cap \{\, (x, y) \mid h_2 < 0 \,\} \cap \{\, (x, y) \mid h_3 < 0 \,\} \cap \{\, (x, y) \mid h_4 < 0 \,\} \\ &= \{\, (x, y) \mid h_1 < 0 \wedge h_2 < 0 \wedge h_3 < 0 \wedge h_4 < 0 \,\} \end{aligned} \tag{26}$$

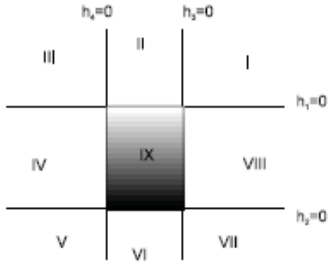


Fig. 3. Second group of obstacles: inside and outside of an obstacle described by **more than one** implicit function inequalities.

where "$\cap$" denotes intersection operation of a set and "$\wedge$" denotes ***logic AND***. The free space of the obstacle (just the union of region I, region II, …, region VIII, and the boundary of the rectangle), can be represented as:

$$\begin{aligned} \overline{A} &= \overline{(A_1 \cap A_2 \cap A_3 \cap A_4)} = \overline{A_1} \cup \overline{A_2} \cup \overline{A_3} \cup \overline{A_4} \\ &= \{\, (x, y) \mid h_1 \geq 0 \,\} \cup \{\, (x, y) \mid h_2 \geq 0 \,\} \cup \{\, (x, y) \mid h_3 \geq 0 \,\} \cup \{\, (x, y) \mid h_4 \geq 0 \,\} \\ &= \{\, (x, y) \mid h_1 \geq 0 \vee h_2 \geq 0 \vee h_3 \geq 0 \vee h_4 \geq 0 \,\} \end{aligned} \tag{27}$$

where "$\cup$" denotes union operation of a set and "$\vee$" denotes ***logic OR***.

3.2 Construction of Object's Defining Inequality

According to section 3.1 we know the inequality "$h(x) < 0$" is a general form to define a representation of an object. We name it as ***defining inequality***. How to construct defining inequality for specific objects in real world? Here we present an approximated method.

To represent an object, another form equivalent to "$h(x) < 0$" is "$f(\boldsymbol{x}) < 1$". The latter form can be easily transformed into "$h(x) < 0$", and is more applicable and convenient for constructing defining inequalities of complex objects from those of the simple objects. In "$f(\boldsymbol{x}) < 1$", the function $f(\boldsymbol{x})$ is named as ***defining function***.

Definition 4: Defining Function: For an object $\boldsymbol{S}$ with its inside $\boldsymbol{I}$, outside $\boldsymbol{T}$, and boundary $\boldsymbol{B}$, a continuous and positive function $f(\boldsymbol{x})$ is called the ***defining function*** of $\boldsymbol{S}$ if for any $\boldsymbol{x} = (x, y, z) \in \boldsymbol{E}^3$, the following hold:

$$f(\boldsymbol{x}) = 1 \quad \Leftrightarrow \quad \boldsymbol{x} \in \boldsymbol{B}$$

$$0< f(\boldsymbol{x}) < 1 \Leftrightarrow \boldsymbol{x} \in \boldsymbol{I} \qquad (28)$$
$$f(\boldsymbol{x}) > 1 \quad \Leftrightarrow \boldsymbol{x} \in \boldsymbol{T}.$$

For example, a defining function for a sphere with radius R and its centre at the origin of the coordinate system is

$$f(\boldsymbol{x}) = (x/R)^2 + (y/R)^2 + (z/R)^2, \qquad (29)$$

and equality $(x/R)^2 + (y/R)^2 + (z/R)^2 = 1$ defines the surface of the sphere.

There are many categories of basic defining functions for object representation (called "primitive solids") such as Quadrics, Superquadrics, and Blobby functions (Berger, 1986).

a. Quadrics. A frequently used class of objects are the quadric surfaces, which are described with second-degree equations (quadratics). They include spheres, ellipsoids, tori, paraboloids, and hyperboloids. Quadric surfaces, particularly spheres and ellipsoids, are common elements of CAD and Graphics, and are often available in CAD and graphics packages as primitives from which more complex objects can be constructed.

Sphere: In Cartesian coordinates, a spherical surface with radius r centered on the coordinate origin is defined as the points (x, y, z) that satisfy the equation

$$x^2+y^2+z^2 = r^2 \qquad (30)$$

Ellipsoid: An ellipsoidal surface can be described as an extension of a spherical surface, where the radii in the three mutually perpendicular directions can have different values. The Cartesian representation for points over the surface of an ellipsoid centered on the origin is

$$(x/r_x)^2 + (y/r_y)^2 + (z/r_z)^2 = 1 \qquad (31)$$

Slabs, i.e. region bounded by two parallel planes with the expression of $(x/a)^2 = 1$, $(y/b)^2 = 1$, $(z/c)^2 = 1$ and circular or elliptical cylinder with the expression of $(x/a)^2 + (y/h)^2 = 1$ are special cases of this general ellipsoid, here a, b, c are positive real numbers.

Torus: A torus is a doughnut-shaped object. It can be generated by rotating a circle or other conic about a specified axis. The Cartesian representation for points over the surface of a torus can be written in the form

$$\left[r - \sqrt{(x/r_x)^2 + (y/r_y)^2}\right]^2 + (z/r_z)^2 = 1 \qquad (32)$$

where r is any given offset value.

b. Superquadrics. This class of objects is a generalization of the quadric representations and provides more flexibility to describe objects (Franklin & Barr, 1981). Superquadrics are formed by incorporating additional parameters into the quadric equations to provide increased flexibility for adjusting object shapes. The number of additional parameters used is equal to the dimension of the object: one parameter for curves and two parameters for surfaces. The most useful one for CSG is superellipsoid.

Superellipsoid. A Cartesian representation for points over the surface of a superellipsoid is obtained from the equation for an ellipsoid by incorporating two exponent parameters:

$$\left[\left(\frac{x-x_o}{r_x}\right)^{\frac{2}{s_2}} + \left(\frac{y-y_o}{r_y}\right)^{\frac{2}{s_2}}\right]^{\frac{s_2}{s_1}} + \left(\frac{z-z_o}{r_z}\right)^{\frac{2}{s_1}} = 1 \qquad (33)$$

where parameters s_1 and s_2 can be assigned any positive real value. For $s_1=s_2=1$, we get an ordinary ellipsoid. Super-ellipsoid is also used to represent the robot in 3D space. We will describe this in more detail in the next section.

c. Blobby functions. Blobby function has been used in computer graphics for representing molecular structure, water droplets and other liquid effects, melting objects, and muscle shapes in the human body. In robotics, it is also useful for describing obstacles. Several models have been developed for representing blobby objects as distribution functions over a region of space. One way to do this is to model objects as combinations of Gaussion density functions, or "bumps". A surface function is then defined as

$$\sum_k b_k e^{-a_k r_k^2} = T \tag{34}$$

where $r_k^2 = \sqrt{x_k^2 + y_k^2 + z_k^2}$, parameter T is a specified threshold, and parameters a_k and b_k are used to adjust the amount of blobbiness of the individual objects. Negative values for parameter b_k can be used to produce dents instead of bumps.

The defining functions (30) to (34) describe the solids in standard positions and orientations. It is usually necessary to translate and rotate the objects to the desired configurations. The rigid body transformations are invertible. Thus, the original inside-outside function can be used after a function inversion. For example, substituting the translation $x=(x'-a)$ into the defining function $(x/a)^2 = 1$ leads to a new defining function $[(x'-a)/a]^2=1$, which describes the surface of an infinite slab centered at $x'=a$ and with the same thickness of $2a$. More generally, let $M \in R^{3\times3}$ denote the desired rotation matrix and $B=[b_1,b_2,b_3]$ denote the translation vector. Then the translated and rotated solid S is given by:

$$x' = Mx + B \tag{35}$$

and the new inside-outside defining function is calculated by inverting the translation and substituting into the old inside-outside function; i.e.

$$f'(x', y', z') = f(x, y, z) \tag{36}$$

where

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = M^{-1} \begin{pmatrix} x'-b_1 \\ y'-b_2 \\ z'-b_3 \end{pmatrix} \tag{37}$$

M^{-1} is the inverse of the rotation matrix. Because the rotation matrix is always orthogonal, its inverse is the same as its transpose, i.e. $M^{-1}=M^T$.

It's easy to give out defining functions for basic object such as sphere and ellipsoids, directly using primitive solids. But objects in real world are usually complex and cannot be directly represented by primitive solids. A natural method to overcome this difficulty is constructing complex objects from simple objects via set operations (union, intersection, and difference).

Given n defining functions $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$, and $f_n(\boldsymbol{x})$ for n objects, respectively, the defining function for the intersection of the n objects is given by

$$f^I(\boldsymbol{x}) = \max(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) \tag{38}$$

and the surface equation of the intersection of the n objects is given by

$$\max(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) = 1 \tag{39}$$

Similarly, the defining function for the union of the n objects is given by

$$f^U(\boldsymbol{x}) = \min(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})). \tag{40}$$

and the surface equation of the union of the n objects is given by

$$\min(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})) = 1 \tag{41}$$

For example, the intersection of the three infinite slabs with defining functions: $f_1(\boldsymbol{x})= (x/r)^2$, $f_2(\boldsymbol{x})= (y/r)^2$, and $f_3(\boldsymbol{x})= (z/r)^2$ has the following surface equation $\max((x/r)^2, (y/r)^2, (z/r)^2) =1$, which represent the surface of a cube.

Although equations (39) and (41) represent the exact surfaces of the intersection and union of the n objects, they are not readily manipulated and computed. To realize a smooth blending of the n objects into a final one, equations (39) and (41) must be approximated by means of suitable functions. A certain degree of smoothing has been obtained in a particular technique for the detection of intersections of 3D objects (Wang & Cartmell, 1998a), but this method does not apply to non-convex objects. A currently wide-used method is the one reported in (Ricci, 1973). We use that method here. The intersection and union can be smoothly approximated as:

$$f^I(\boldsymbol{x}) = (f_1{}^m(\boldsymbol{x}) + f_2{}^m(\boldsymbol{x}) + f_n{}^m(\boldsymbol{x}))^{1/m} \tag{42}$$

$$f^U(\boldsymbol{x}) = (f_1{}^{-m}(\boldsymbol{x}) + f_2{}^{-m}(\boldsymbol{x}) + f_n{}^{-m}(\boldsymbol{x}))^{-1/m} \tag{43}$$

The resulting approximations of the surfaces for the intersection and union of the n objects are respectively represented as:

$$(f_1{}^m(\boldsymbol{x}) + f_2{}^m(\boldsymbol{x}) + f_n{}^m(\boldsymbol{x}))^{1/m} = 1 \tag{44}$$

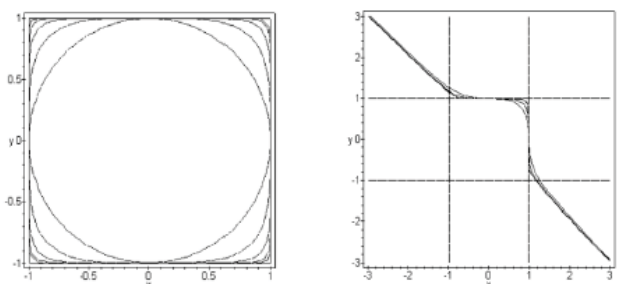
$$(f_1{}^{-m}(\boldsymbol{x}) + f_2{}^{-m}(\boldsymbol{x}) + f_n{}^{-m}(\boldsymbol{x}))^{-1/m} = 1 \tag{45}$$

where m is a positive real number. m is used to control the accuracy of the smoothing approximation and thus is called the control parameter. A larger m produces blending surfaces that cling more closely to the primitive objects. Ricci (Ricci, 1973) proved that when $m\to\infty$, the approximations (42) and (43) give the exact description of the intersection and union respectively. These approximations have the following advantages:

(i) Blending effects are primarily noticeable near surface intersections.

(ii) $f^I(\boldsymbol{x})$ and $f^U(\boldsymbol{x})$ are differentiable which may avoid the possible difficulties in computation due to the differentiability of the max and min functions.

One problem that has been investigated in the previous literature (Wang & Lane, 1997) is the choice of the control parameter m in Equations (44) and (45). Although Ricci (Ricci, 1973) suggested that any positive real number may be chosen as the candidate, our experience has shown that when using slabs as the basic primitives, some care must be taken. In this case, m must be an integer, which leads to $2m$ as an even number. Some examples are given below.



(a) (b)

(a) m = 1, 2, 4, 8, and 16 from inside to outside; (b) m = 3/2, 9/2, and 33/2 from inside to outside.

Fig. 4. Illustration of intersection $f_1 \cap f_2$. $f_1 = x^2$, $f_2 = y^2$, $((x^2)^m + (y^2)^m)^{1/m} = 1$.

Figures 4(a) and 4(b) show two examples, using two slabs $f_1(\boldsymbol{x})= x^2$, $f_2(\boldsymbol{x})= y^2$ as basic primitives to construct a rounded square with different orders (control parameters). In Figure 4(a), m has been chosen to be an integer. When using m=1 to approximate the intersection of $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$, the result is a circle. As m increases the approximation to the intersection of $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$, a square with length and width being 1, get better. It can be seen that when m = 8 or 16, the approximation is very close to the square. In Figure 4(b), the values of m are fractions rather than integers so that $2m$ is an odd number. The resulting approximate implicit function is not, as could be expected, a closed curve. Closed curve here means that the number of real circuits is limited to one and that the circuit does not extend to infinity. Only in the first quadrant it is a good approximation of the intersection of $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$. Furthermore, if m is chosen as a decimal so that $2m$ is not an integer, then the resulting approximate implicit function is of real value only in the first quadrant: see the figures given in (Franklin & Barr, 1981). The above discussion also applies to the union operation.
If, on the other hand, a circle or an ellipse is used as the primitive to construct a new object, any positive number m able to keep the closeness of the intersection and union operation. Figures 5 and 6 give two examples. Similarly as m increases, the approximation gets better.

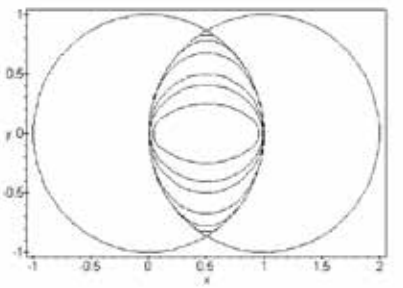


Fig. 5. Illustration of intersection $f_1 \cap f_2$. $f_1 = x^2+y^2$, $f_2 = (x-1)^2+y^2$. m = 0.6, 0.8, 1, 2, 5, and 25 from inside to outside.

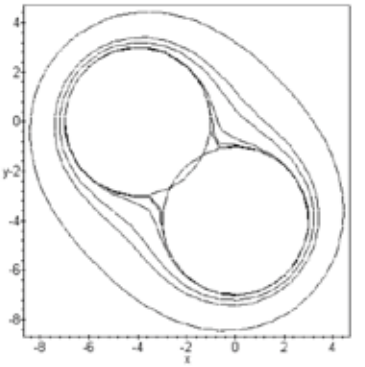


Fig. 6. Illustration of union $f_1 \cup f_2$. $f_1 = [(x+4)^2+y^2]/3^2$, $f_2 = [x^2+(y+4)^2]/3^2$. m= 8, 4, 2, 1, 0.8, and 0.5 from inside to outside.

3.3 Robot Representation

Since robot is a movable and rotatable object in the workspace, to clearly model and dynamically manipulate a robot in 3D space, we must be capable of representing not only its shape and size, but also its spatial location and orientation. Robot representation means the expression of a point, denoted by $\boldsymbol{x}=(x, y, z)$, on the boundary of the robot as the function of its spatial position and orientation variables. Normally there are two mathematical ways to describe the boundary of a robot. The first is the implicit function which takes the form of $g(x, y, z) = 0$ for its boundary expression. The second is the parametric form, in which the x, y, and z are expressed as functions of two auxiliary parameters $\boldsymbol{v}=(t_1, t_2)$, so that $x=\boldsymbol{x}(\boldsymbol{v})=\boldsymbol{x}(t_1, t_2)$, $y=\boldsymbol{y}(\boldsymbol{v})=\boldsymbol{y}(t_1, t_2)$, and $z=\boldsymbol{z}(\boldsymbol{v})=\boldsymbol{z}(t_1, t_2)$. In the following context, we will use both the implicit and the parametric forms to formulate a robot.

Let the geometric centre point $\boldsymbol{O}$ of the robot, denoted by $\boldsymbol{O}(x_o, y_o, z_o)$, be chosen as the position parameters and let a set of three orientation angles, denoted by $\boldsymbol{\Theta}(\theta_1, \theta_2, \theta_3)$, be chosen as the orientation parameters. Then a robot can be represented as:

$$\{ (x, y, z) \mid g(x, y, z, x_o, y_o, z_o, \theta_1, \theta_2, \theta_3) = 0 \} \tag{46}$$

and its equivalent parametric form can be expressed as:

$$\begin{aligned} \{ (x, y, z) \mid & x=\boldsymbol{x}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}), \\ & y=\boldsymbol{y}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}), \\ & z=\boldsymbol{z}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, \boldsymbol{v}) \end{aligned} \tag{47}$$

x_o, y_o, z_o together with $\theta_1, \theta_2, \theta_3$ are responsible for determining the position and orientation of a robot. We call $(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3)$ as the robot's ***space configuration variables***.

The implicit function we use to describe the robot here is the superellipsoid proposed in references (Barr, 1981; Berger, 1986), which is a Constructive Solid Geometry (CSG) primitive for a broad family of robot and obstacles. The superellipsoid is defined as:

$$\left[\left(\frac{x-x_o}{r_x} \right)^{\frac{2}{s_2}} + \left(\frac{y-y_o}{r_y} \right)^{\frac{2}{s_2}} \right]^{\frac{s_2}{s_1}} + \left(\frac{z-z_o}{r_z} \right)^{\frac{2}{s_1}} = 1 \tag{48}$$

Its parametric form is:

$$\begin{aligned} & bx= r_x \cos^{s_1}(t_1) \cos^{s_2}(t_2)+ x_o ; \\ & y= r_y \cos^{s_1}(t_1) \sin^{s_2}(t_2)+ y_o; \quad -\pi/2 \le t_1 \le \pi/2; \ 0 \le t_2 \le 2\pi \\ & z= r_z \sin^{s_1}(t_1)+ z_o. \end{aligned} \tag{49}$$

where rx , ry , rz define the geometric extent, s1 and s2 specify the shape properties (s1 is the squareness parameter in the north-south direction; s2 is the squareness parameter in the east-west direction), and xo, yo, zo describe the spatial location. The superellipsoid can be constructed from the basic slabs. Some superellipsoid shapes produced by the choice of different values for parameters s1 and s2 are shown in Fig. 7 when rx=ry=rz.

From (Barr, 1981; Berger, 1986), we know that most kinds of robots can be simulated by the broad family of easily defined superellipsoid primitives. In addition to the superspherical shapes that can be generated using various values for parameters s_1 and s_2, other superquadratic shapes can also be combined to create more complex structures. More details about them can be found in (Barr, 1981; Berger, 1986; Wang & Lane, 1997).

(48) and (49) describe a 3D robot in the standard orientation with $\theta_1=\theta_2=\theta_3=0$. It's necessary to give a general representation of its spatial orientation. The concepts of Euler angle and Euler angle conversion are introduced in the following.

The Euler angles comprise three arbitrary rotations in 3D space to describe the spatial orientation of an object. How the Euler angle is defined and how the rotation matrix R is obtained are briefly described, with the assumption that we start in frame S with Cartesian axes, x_{old}, y_{old}, and z_{old}. A positive (anti-clockwise) rotation of magnitude a about the z axis of S is first carried out and the resulting frame is called S'. Then it is followed by a positive rotation of magnitude about the y' axis of frame S' and the resulting frame is called S''. Finally, a positive rotation of magnitude about the z'' axis of S'' is made and the resulting frame is called S'''. Fig. 8 illustrates the combined effect of these steps.
The combined result of these three rotations is mathematically expressed by the following rotation matrix:

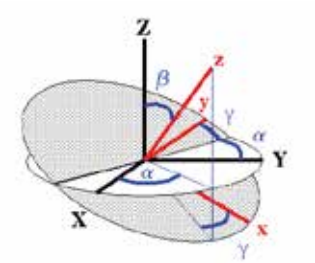


Fig. 8. Euler angle conversion.

$$\begin{pmatrix} x_{new} \\ y_{new} \\ z_{new} \end{pmatrix} = R \cdot \begin{pmatrix} x_{old} \\ y_{old} \\ z_{old} \end{pmatrix} \tag{50}$$

where R is the rotation matrix:

$$\begin{pmatrix} \cos\alpha\cos\beta\cos\gamma-\sin\alpha\sin\gamma & \sin\alpha\cos\beta\cos\gamma+\cos\alpha\sin\gamma & -\sin\beta\cos\gamma \\ -\cos\alpha\cos\beta\sin\gamma-\sin\alpha\cos\gamma & \sin\alpha\cos\beta\sin\gamma-\cos\alpha\cos\gamma & \sin\beta\sin\gamma \\ \cos\alpha\sin\beta & \sin\alpha\sin\beta & \cos\beta \end{pmatrix}$$

(50) is equivalent to (51) as follows:

$$\begin{cases} x_{new} = & x_{old}\cdot(\cos\alpha\cos\beta\cos\gamma-\sin\alpha\sin\gamma) \\ & +y_{old}\cdot(\sin\alpha\cos\beta\cos\gamma+\cos\alpha\sin\gamma) \\ & -z_{old}\cdot(\sin\beta\cos\gamma); \\ y_{new} = & x_{old}\cdot(-\cos\alpha\cos\beta\sin\gamma-\sin\alpha\cos\gamma) \\ & +y_{old}\cdot(\sin\alpha\cos\beta\sin\gamma-\cos\alpha\cos\gamma) \\ & +z_{old}\cdot(\sin\beta\sin\gamma); \\ z_{new} = & x_{old}\cdot(\cos\alpha\sin\beta)+y_{old}\cdot(\sin\alpha\sin\beta) \\ & +z_{old}\cdot\cos\beta; \end{cases} \tag{51}$$

where vector $(x_{old}, y_{old}, z_{old})$ represents the point in the first coordinate system and $(x_{new}, y_{new}, z_{new})$ represent the point in the new coordinate system. The ranges for a, β, γ are

$$0\le a\le 2\pi,\ 0\le\beta\le\pi,\ 0\le\gamma\le 2\pi \tag{52}$$

The combination of rotation transformation (52) with (49) results in the parametric form for a superellipsoid robot in a general position $\boldsymbol{O}(x_o, y_o, z_o)$ and orientation $\boldsymbol{\Theta}(\theta_1, \theta_2, \theta_3)$:

$$
\begin{aligned}
x&=(r_x \cos^{s_1}(t_1)\ \cos^{s_2}(t_2))l_1+(r_y \cos^{s_1}(t_1)\ \sin^{s_2}(t_2))l_2+(r_z \sin^{s_1}(t_1))l_3+ x_o;\\
y&=(r_x \cos^{s_1}(t_1)\ \cos^{s_2}(t_2))m_1+(r_y \cos^{s_1}(t_1)\ \sin^{s_2}(t_2))m_2+(r_z \sin^{s_1}(t_1))m_3+ y_o;\\
z&=(r_x \cos^{s_1}(t_1)\ \cos^{s_2}(t_2))n_1+(r_y \cos^{s_1}(t_1)\ \sin^{s_2}(t_2))n_2+(r_z \sin^{s_1}(t_1))n_3+z_o.
\end{aligned}
\tag{53}
$$

where

where the oriental angles θ_1, θ_2, θ_3 are specified by three Euler angles. For more details about Euler angle conversion, see (Rose, 1957).

4. Converting the Robot Path Planning Problem into the Semi-infinite Optimization Problem

$$
\begin{aligned}
l_1&=cos(\theta_1)cos(\theta_2)cos(\theta_3)\text{-}sin(\theta_1)sin(\theta_3)\\
m_1&=\text{-}sin(\theta_1)cos(\theta_3)\text{-}cos(\theta_1)cos(\theta_2)sin(\theta_3)\\
n_1&=cos(\theta_1)sin(\theta_2)\\
l_2&=\text{-}cos(\theta_1)sin(\theta_3)+cos(\theta_2)sin(\theta_2)sin(\theta_3)\\
m_2&=\text{-}cos(\theta_1)cos(\theta_3)+sin(\theta_1)cos(\theta_2)sin(\theta_3\\
n_2&=sin(\theta_2)sin(\theta_3)\\
_3&=cos(\theta_1)sin(\theta_2)\\
_3&=sin(\theta_1)sin(\theta_2)\\
3&=cos(\theta_2)\\
\pi/2&\le t_1 \le \pi/2\\
&\le t_2 \le 2\pi
\end{aligned}
\tag{54}
$$

4.1 Collision-free Condition for the Obstacle Avoidance

According to **Proposition 1** and the representations of the two classes of obstacles, we can get the condition for a point to be collision-free from an obstacle:

Proposition 2: *The necessary and sufficient condition for a point* $\boldsymbol{x}$ *to be collision-free from a first-group obstacle* $A=\{\boldsymbol{x} \mid h(\boldsymbol{x})<0\}$ *is that* $\boldsymbol{x}\in \overline{A}$, *that is,* $h(\boldsymbol{x})\ge 0$.

Proposition 3: *The necessary and sufficient condition for a point* $\boldsymbol{x}$ *to be collision-free from a second-group obstacle* $A=\{\boldsymbol{x} \mid h_1(\boldsymbol{x})<0 \wedge h_2(\boldsymbol{x})<0 \wedge \ldots \wedge h_m(\boldsymbol{x})<0\}$ *is that* $\boldsymbol{x}\in \overline{A}$, *that is,* $h_1(\boldsymbol{x})\ge 0 \vee h_2(\boldsymbol{x})\ge 0 \vee \ldots \vee h_m(\boldsymbol{x})\ge 0$.

Let $v_i(\boldsymbol{x})=(h_i^2(\boldsymbol{x})+t^2)^{1/2}+h_i(\boldsymbol{x})$, $(i=1, 2, \ldots, m)$, and Δv be a small positive value, then according to **Theorem 1** and **Proposition 3**, we can get a realistically necessary and sufficient condition for a point to be collision-free from a second-group obstacle: $\Delta v-\sum_{i=1}^{m} v_i(x)\le 0$. In fact, with respect to the realistic requirement of robot path planning, we can represent the free space of the second-group obstacle as:

$$
= \{\mathbf{x} \mid \Delta v-\sum_{i=1}^{m} v_i(x)\le 0\}
\tag{55}
$$

Now let's consider the collision-free condition in the presence of multiple obstacles. In the presence of multiple obstacles, the condition for a point to be collision-free from all obstacles

in the workspace is obviously that the point must fall into the intersection of free spaces of all the obstacles. We get:

Theorem 2: *In the presence of multiple obstacles A_i, i=1, 2,…, j; (j>1), the necessary and sufficient condition for a point* $\boldsymbol{x}$ *to be collision-free from all the obstacles is that* $\boldsymbol{x} \in \overline{A_1} \cap \overline{A_2} \cap ... \cap \overline{A_j}$.

Suppose in a workspace there are totally s first-group obstacles and m second-group obstacles, and they are respectively defined as:

$$_i = \{ \boldsymbol{x} \mid g_i(\boldsymbol{x}) < 0 \},\ i=1, 2,\ldots, s \tag{56}$$

$$_j = \{ \boldsymbol{x} \mid h_{j,1}(\boldsymbol{x}) < 0 \wedge h_{j,2}(\boldsymbol{x}) < 0 \wedge \ldots \wedge h_{j,k_j}(\boldsymbol{x}) < 0 \},\ j=1, 2, \ldots, m.$$

We note $G_i(\boldsymbol{x}) = -\, g_i(\boldsymbol{x})$, $i=1, 2,\ldots, s$, and consequently the representation of the free space of A_i changes into:

$$= \{ \boldsymbol{x} \mid G_i(\boldsymbol{x}) \le 0 \},\qquad i=1, 2,\ldots, s. \tag{57}$$

Let $v_{j,r}(\boldsymbol{x}) = (h_{j,r}^2(\boldsymbol{x}) + t^2)^{1/2} + h_{j,r}(\boldsymbol{x})$, ($j=1, 2, \ldots, m$; $r=1, 2,\ldots, k_j$), and Δv_j ($j=1, 2, \ldots, m$) be small positive values. And further let $H_j(\boldsymbol{x}) = \Delta v_j - \sum_{r=1}^{k_j} v_{j,r}(x)$, then free space of B_j can be consequently represented as:

$$\overline{B_j} = \{ \boldsymbol{x} \mid H_j(\boldsymbol{x}) \le 0 \},\qquad j=1, 2, \ldots, m. \tag{58}$$

Thus, the condition for a point $\boldsymbol{x}$ to be collision-free from all the obstacles is that:

$$\in S = \overline{A_1} \cap \overline{A_2} \cap ... \cap \overline{A_s} \cap \overline{B_1} \cap \overline{B_2} \cap ... \cap \overline{B_m}$$

$$\{ \boldsymbol{x} \mid G_1(\boldsymbol{x}) \le 0 \} \cap \{ \boldsymbol{x} \mid G_2(\boldsymbol{x}) \le 0 \} \cap \ldots \cap \{ \boldsymbol{x} \mid G_s(\boldsymbol{x}) \le 0 \}$$

$$\{ \boldsymbol{x} \mid H_1(\boldsymbol{x}) \le 0 \} \cap \{ \boldsymbol{x} \mid H_2(\boldsymbol{x}) \le 0 \} \cap \ldots \cap \{ \boldsymbol{x} \mid H_m(\boldsymbol{x}) \le 0 \}$$

$$\{ \boldsymbol{x} \mid G_1(\boldsymbol{x}) \le 0 \wedge G_2(\boldsymbol{x}) \le 0 \wedge \ldots \wedge G_s(\boldsymbol{x}) \le 0 \wedge H_1(\boldsymbol{x}) \le 0 \wedge H_2(\boldsymbol{x}) \le 0 \wedge \ldots \wedge H_m(\boldsymbol{x}) \le 0 \} \tag{59}$$

that is,

$$G_1(\boldsymbol{x}) \le 0 \wedge G_2(\boldsymbol{x}) \le 0 \wedge \ldots \wedge G_s(\boldsymbol{x}) \le 0 \wedge H_1(\boldsymbol{x}) \le 0 \wedge H_2(\boldsymbol{x}) \le 0 \wedge \ldots \wedge H_m(\boldsymbol{x}) \le 0 \tag{60}$$

4.2 Constraints of Path Planning Problem

An obvious ***necessary and sufficient*** condition for a robot to be collision-free from multiple obstacles is that: all the points inside or on the boundary of the robot fall into the intersection of free spaces of all the obstacles. A little weaker, but sufficient in almost all realistic cases, condition is that: *all the points on the boundary of the robot fall into intersection of all the free spaces.*

Suppose in the workspace there are totally s first-group obstacles and m second-group obstacles, just as defined in (56), and also a static superellipsoid-shaped robot with squareness parameters s_1, s_2, geometric extent r_x, r_y, r_z, center position $\boldsymbol{O}(x_o, y_o, z_o)$ and orientation $\boldsymbol{\Theta}(\theta_1, \theta_2, \theta_3)$. According to (53), the set of the robot boundary points is:

$$T = \{ (x, y, z) \mid$$

$$x = (r_x \cos^{s_1}(t_1) \cos^{s_2}(t_2)) l_1 + (r_y \cos^{s_1}(t_1) \sin^{s_2}(t_2)) l_2 + (r_z \sin^{s_1}(t_1)) l_3 + x_o$$

$$y = (r_x \cos^{s_1}(t_1) \cos^{s_2}(t_2)) m_1 + (r_y \cos^{s_1}(t_1) \sin^{s_2}(t_2)) m_2 + (r_z \sin^{s_1}(t_1)) m_3 + y_o$$

$$z = (r_x \cos^{s_1}(t_1) \cos^{s_2}(t_2)) n_1 + (r_y \cos^{s_1}(t_1) \sin^{s_2}(t_2)) n_2 + (r_z \sin^{s_1}(t_1)) n_3 + z_o \quad -\pi/2 \le t_1 \le \pi/2;\ 0 \le t_2 \le 2\pi. \tag{61}$$

l_i, m_i, n_i, (i=1,2,3) are defined same as in (54). Let (x, y, z) = ($\boldsymbol{X}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, t_1, t_2)$, $\boldsymbol{Y}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, t_1, t_2)$, $\boldsymbol{Z}(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3, t_1, t_2)$), and the space configuration vector $\boldsymbol{u}$= (x_o, y_o, z_o, θ_1, θ_2, θ_3), then the collision-free condition, according to (60), can be represented as:
For all $\boldsymbol{v}$ = (t_1, t_2) that $-\pi/2 \le t_1 \le \pi/2$, $0 \le t_2 \le 2\pi$, the following expression hold:

$$G_1(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge$$
$$G_2(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge \ldots \wedge$$
$$G_s(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge$$
$$H_1(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge$$
$$H_2(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \wedge \ldots \wedge$$
$$H_m(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})) \le 0 \tag{62}$$

Let P_i ($\boldsymbol{u}$, $\boldsymbol{v}$) = $G_i(\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v}))$ i=1,2,…,s; Q_j ($\boldsymbol{u}$, $\boldsymbol{v}$) =H_j ($\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Y}(\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{Z}(\boldsymbol{u}, \boldsymbol{v})$) j = 1,2,…, m, then (62) changes into the following:

$$P_1(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge P_2(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge \ldots \wedge P_s(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge$$
$$Q_1(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge Q_2(\boldsymbol{u}, \boldsymbol{v}) \le 0 \wedge \ldots \wedge Q_m(\boldsymbol{u}, \boldsymbol{v}) \le 0 \tag{63}$$

In path planning the configuration variables also have certain range limits. This requirement can be described as:

$$x_L \le x_o \le x_U, y_L \le y_o \le y_U, z_L \le z_o \le z_U, \alpha_1 \le \theta_1 \le \beta_1, \alpha_2 \le \theta_2 \le \beta_2, \alpha_3 \le \theta_3 \le \beta_3 \tag{64}$$

that is,

$$\boldsymbol{u}_L \le \boldsymbol{u} \le \boldsymbol{u}_U \tag{65}$$

where $\boldsymbol{u}_L$=(x_L, y_L, z_L, α_1, α_2, α_3), $\boldsymbol{u}_U$=(x_U, y_U, z_U, β_1, β_2, β_3).
(63) and (65) form the inequality constraints required in the formulation of the SICO problem for path planning.

4.3 Design of the Objective Function

There are many ways to design the objective function. In a nonlinear programming problem, this function must represent some meaning of the practical problem, for example, minimum time, minimum distance, minimum energy, or minimum cost. From a mathematical viewpoint, this function must have a minimum lower bound. For the path planning problem, the goal configuration must be designed as the unique global minimum of the configuration variables. We use a quadratic function of the form (66) as the objective function, with the goal configuration point (x_g, y_g, z_g) and goal configuration angles (φ_1, φ_2, φ_3) being its unique global minimum point and satisfying the condition that $min\ f(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3) = f(x_g, y_g, z_g, \varphi_1, \varphi_2, \varphi_3) = 0$.

$$f(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3) =$$
$$w((x_o-x_g)^2+(y_o-y_g)^2+(z_o-z_g)^2)+(1-w)((\theta_1-\varphi_1)^2+(\theta_2-\varphi_2)^2+(\theta_3-\varphi_3)^2) \tag{66}$$

where w is a non-negative weighted factor that satisfies: $0 \le w \le 1$.
In (66), w is used to adjust the relative effects of the spatial position $(x_o-x_g)^2+(y_o-y_g)^2+(z_o-z_g)^2$ and the spatial orientation $(\theta_1-\varphi_1)^2+(\theta_2-\varphi_2)^2+(\theta_3-\varphi_3)^2$. When w=1, only the effect of the spatial position factor is considered. w can be adaptively adjusted and be revised during the searching process. (66), (63), and (65) together form a semi-infinite constrained optimization

problem. If we use the initial configuration variables $(x_s, y_s, z_s, \delta_1, \delta_2, \delta_3)$ as the initial estimate of the optimization problem, the optimum search for $(x_o^*, y_o^*, z_o^*, \theta_1^*, \theta_2^*, \theta_3^*)$ is equivalent to searching the goal configuration variables. If the algorithm is convergent and the problem has a solution, then we will find that $x_o^* = x_g, y_o^* = y_g, z_o^* = z_g, \theta_1^* = \varphi_1, \theta_2^* = \varphi_2, \theta_3^* = \varphi_3$.
In summary, the fundamental idea for this approach is to represent the free space determined by the robot and obstacles as inequality constraints for a semi-infinite constrained optimization problem in 3D space. The goal configuration is designed as the unique global minimum point of the objective function. The initial configuration is treated as the first search point for the optimization problem. Then the numerical algorithm developed for solving the semi-infinite constrained optimization problem can be applied to solve the robot motion planning problem. Every point generated using the semi-infinite constrained optimization method is guaranteed to be in free space and therefore is collision free.

5. Implementation Considerations and Simulation Results

When implementation is carried out, we only consider the motion of the robot in 3D space. The vehicle is modeled as a superellipsoid with different shapes and the obstacles are modeled by circle, ellipsoid, cylinder, tetrahedron, cuboids, and various other shapes of superellipsoid, which belong to either the first group or the second group.

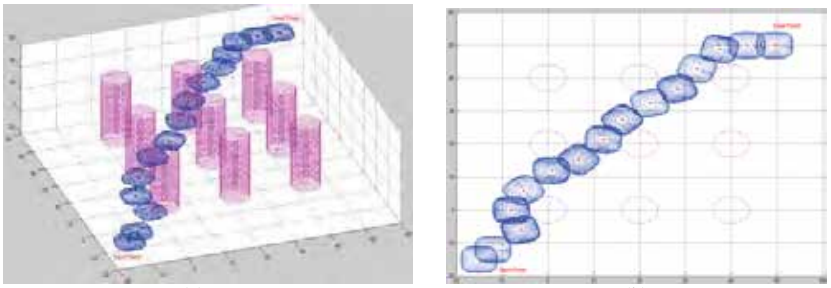
5.1 Algorithm Implementation Consideration

The implementation of the semi-infinite optimization is based on the constrained optimization toolbox (The Math Works Inc., 1993), but some modifications have been made. The first is the control of the output. In (The Math Works Inc., 1993), only the final result of the vector $\boldsymbol{x} = (x_o^*, y_o^*, z_o^*, \theta_1^*, \theta_2^*, \theta_3^*)$ is provided. However, the important thing for the robot path planning problem is to generate a smooth path. Therefore, a new function which outputs the current $(x_o, y_o, z_o, \theta_1, \theta_2, \theta_3)$ at every iteration has been added. The second is the control of the change of Euler Angles in the line search algorithm for every iteration. Recall that the principle of developing an algorithm in nonlinear programming is to minimize the number of function evaluations, which represents the most efficient way for finding the optimum $\boldsymbol{x}^*$. Thus, the changing step is automatically chosen as large as possible to minimize the objective function in the line search direction. The disadvantage of this strategy when applied to path planning is that it sometimes leads to a non-smooth path, so a modification has been made. For more details on the implementation of the semi-infinite optimization algorithms, see the reference (The Math Works Inc., 1993).
In the following the results obtained for 3D path planning will be given. In all the experiments the algorithm adopted is the Sequential Quadratic Programming (SQP) (The Math Works Inc., 1993). The limits of the workspace are: Ol=(-20, -20, -20) and Ou=(60, 60, 60), thus the workspace is surrounded by a cube with length 80(-20, 60), width 80(-20, 60), and height 80(-20,60). The inequality Ol≤O≤Ou will guarantee that the generated path must be in the workspace. Let Os=(xs, ys, zs) and Θs=(δ1, δ2, δ3) denote the initial configuration and Og=(xg, yg, zg), Θg=(φ1, φ2, φ3) denote the goal configuration. The robot's start and goal configurations are chosen as (xs, ys, zs, δ1, δ2, δ3)=(-20, -20, -20, 0, 0, 0) and (xg, yg , zg , φ1, φ2, φ3)=(50, 50, 50, 0, 0, 0) respectively . The objective function is defined as:

$$f = w((x_o-50)^2+(y_o-50)^2+(z_o-50)^2)+(1-w)(\theta_1^2+\theta_2^2+\theta_3^2) \quad (67)$$

where w is chosen as $0 \le w \le 1$.

5.2 Simulation Results with the First Kind of Objects



(a) (b)

Fig. 10. Workspace with nine cylinders and the generated path. (a) 3D view. (b) 2D view.

In order to demonstrate the convergence of the problem formulation to the goal configuration, the workspace considered in example 1 contains seven obstacles as shown in Figs. 9(a) and 9(b). The obstacles are represented as spheres with the following boundary expressions: $(x-20)^2 + (y-20)^2 + (z-20)^2 = 225$, $x^2 + (y-20)^2 + (z-20)^2 = 25$, $(x-40)^2 + (y-20)^2 + (z-20)^2 = 25$, $(x-20)^2 + y^2 + (z-20)^2 = 25$, $(x-20)^2 + (y-40)^2 + (z-20)^2 = 25$, $(x-20)^2 + (y-20)^2 + z^2 = 25$, $(x-20)^2 + (y-20)^2 + (z-40)^2 = 25$.The robot is represented by a superellipsoid with $r_x=5$, $r_y=4$, $r_z=3$, $s_1=s_2=1$, which is an ellipsoid. Figs. 9(a) and 9(b) show the same generated path, the same obstacles and the same robot, but from different view angles. It can be observed that the optimization algorithm does converge to the goal and a smooth path has been generated.

In order to illustrate the suitability of the approach for different obstacle shapes, we have shown another distributed-obstacle situation as shown in Figs. 10(a) and 10(b). The results simulate a real world path planning task encountered in offshore industry. The designed workspace contains a set of cylinders which could be easily recognized as pipelines or a base of an offshore structure. In this example, we have 9 cylinders to represent obstacles with the following boundary equations: $x^2 + y^2 = 16$, $x^2 + (y-20)^2 = 16$, $x^2 + (y-40)^2 = 16$, $(x-20)^2 + y^2 = 16$, $(x-20)^2 + (y-20)^2 = 16$, $(x-20)^2 + (y-40)^2 = 16$, $(x-40)^2 + y^2 = 16$, $(x-40)^2 + (y-20)^2 = 16$, $(x-40)^2 + (y-40)^2 = 16$ where $-20 \le z \le 60$. The robot is represented by a superellipsoid with $r_x=5$, $r_y=4$, $r_z=3$, $s_1=s_2=1.5$, which is a superellipsoid. Fig. 10(a) is a 3D view and Fig. 10(b) is a 2D view. In this example the 2D view clearly shows the collision avoidance of the robot from the obstacles. From Figs. 10(a) and 10(b), we can also observe that the generated path passes behind the cylinder without touching it, and the plotted path avoids all the obstacles and converges to the goal in a smooth way.

In addition to the simulation results shown in Figs. 9 and 10, we have carried out another test with mixed superellipsoids and cylinders as shown in Figs. 11(a) and 11(b). In this test, five obstacles are included and the robot is represented by a superellipsoid with $r_x=8$, $r_y=8$, $r_z=6$, $s_1=s_2=0.8$. The boundary expressions for the obstacles are: 2 cylinders : $(x-40)^2 + y^2 = 144$, $30 \le z \le 50$, $(x-30)^2 + (y-30)^2 = 100$, $-5 \le z \le 30$, 3 superellipoids: $((x-15)/12) + ((y+10)/8) + ((z-12)/12) = 1$, $((x-15)/20)^{2/3} + ((y-40)/18)^{2/3} + ((z+10)/12)^{2/3} = 1$, $(x-10)^2 + (y-10)^2 + (z-10)^2 = 100$. Fig. 11(b) is an enlarged view of Fig. 11(a). The experimental results show that the robot can adjust its orientation angles autonomously to reach the goal point.

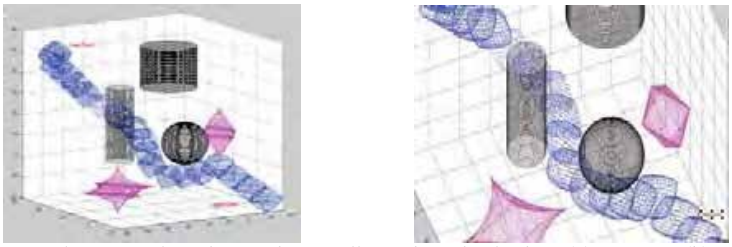
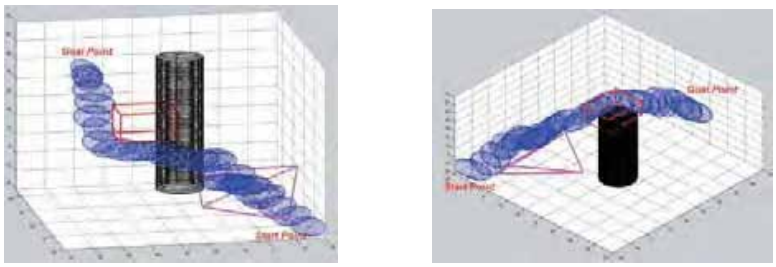


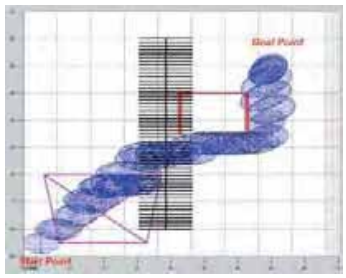
Fig. 11. Simulation result with mixed superellipsoids and cylinders. (a) 3D view. (b) another 3D view.

5.3 Simulation Results with Both the First and Second Kinds of Objects

Figs. 12(a), 12(b), and 12(c) show the experimental results with the environment including a triangular pyramid and a cube which belong to the second group and a cylinder-like obstacle belonging to the first group. The triangular pyramid's four vertexes are: V_0=(-5,-10,-15), V_1=(20,15,-15), V_2=(5,-15,10), V_3=(5,20,10) and its outside is represented by {($h_1 \geq -x-y-0.6z-14$) $\cup$ ($h_2 \geq -x+0.4z+1$) $\cup$ ($h_3 \geq x+0.6z-11$) $\cup$ ($h_4 \geq -x+y-0.8z-7$)}.The cube is represented by $10 \leq x \leq 25$, $25 \leq y \leq 45$, $25 \leq z \leq 40$, while the cylinder is descried as $(x-35)^2 + (y-20) \leq 64$, $-10 \leq z \leq 50$. The robot is represented by a superellipsoid with r_x=8, r_y=6, r_z=4, s_1=s_2=1.It is obvious that the path generated clearly avoids the obstacle and converges to the goal.



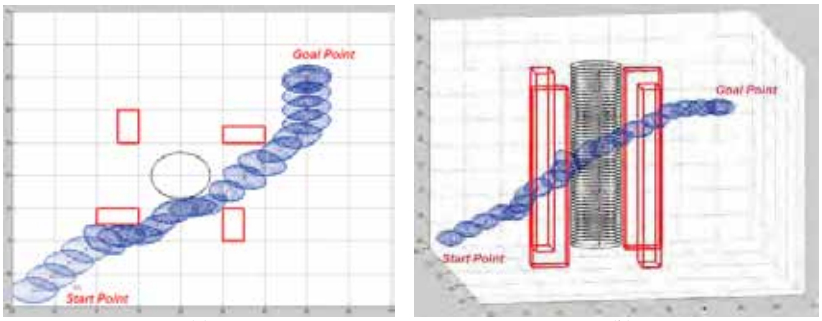
(a) (b)



(c)

Fig. 12. Simulation result with both the first and the second kinds of objects. (a) 3D view. (b) another 3D view. (c) 2D view.

Figs. 13(a) and 13(b) show the results simulating another typical real world path planning task encountered in offshore industry. The designed workspace contains a cylinder and several cubes which may be models for pipelines or a base of an offshore structure. The cylinder is described as follows: $(x-20)^2 + (y-20)^2 \leq 49, -10 \leq z \leq 50$. The four cubes are represented as: $\{0 \leq x \leq 10, 5 \leq y \leq 10, -15 \leq z \leq 55\}$, $\{30 \leq x \leq 40, 30 \leq y \leq 35, -15 \leq z \leq 55\}$,$\{5 \leq x \leq 10, 30 \leq y \leq 40, -15 \leq z \leq 55\}$,$\{30 \leq x \leq 35, 0 \leq y \leq 10, -15 \leq z \leq 55\}$. The robot is represented by a superellipsoid with $r_x=6$, $r_y=4$, $r_z=3$, $s_1=s_2=1.2$. Figs. 13(a) and 13(b) show the clear avoidance of the obstacles by the robot and the convergence to the goal.



(a) (b)

Fig. 13. Simulation result with four cubes and one cylinder. (a) 2D view. (b) 3D view.

5.4 Discussion about Efficiency of the Approach

A criterion for efficiency analysis used by many previous path planning algorithms is the worst case computational complexity when the robot path planning problem is formulated as a discrete mathematics problem (Kreyszig, 1993; Rosen, 1991), based on the assumption that objects are represented by polygons or polyhedrons and a discrete search algorithm is used. However in constrained optimization there is no such a criterion corresponding to that used in discrete mathematics for measuring the worst case computational complexity, because the time used depends on both the size of the problem (i.e. the number of the equalities) and the form of the objective function and the constraints.

As shown in our simulation experiments, the time for the robot to take a step varies, because at some places it must adjust its pose or step size, and at other places it doesn't have to. In the simulation experiment shown in Figs. 10 (a) and 10(b), the average time for one step is about 0.8s with Intel Pentium 4 CPU running the algorithm realized in Matlab. Besides improving the performance of the implementation and the algorithm itself, there is still big room for efficiency improvement. For example, in our experiments the value of the distance that robot is permitted to be close to obstacles is so small (close to 0) that sample points of the surface of the robot (represented as a superellipsoid) and the obstacles must be very dense, otherwise collisions are likely to occur. Density of sample points is one of the main sources of the computational complexity, but in real applications, the distance is usually bigger and the sample points may be sparse, and consequently the time needed for computation may be much less.

To precisely analyze the computational complexity of our method is an important, necessary, but difficult job. It must include determining proper complexity description model,

analyzing performance of the implementation of SCO algorithm which is applied in our method, etc. We hope to address it in further papers.

6. Conclusions

In this chapter, inequality transformation and semi-infinite constrained optimization techniques have been presented for the development of a realistic Robot Path Planning approach. We have shown the principle of converting the path planning problem into the standard Semi-infinite Constrained Optimization problem. This direct path planning approach considers the robot's 3D shape and is totally different from the traditional approaches in the way that the calculation of the Cspace obstacles is no longer needed. From the viewpoint of robot path planning, this paper presents a new way of using a classical engineering approach. The generality of representing the free space of all the objects using the inequalities $g_i(x) \leq 0$ makes this optimization-based approach suitable for different object shapes, and it has significantly simplified the construction of the objects by sensors and computer vision systems. The iterative nature of the search for the optimization point makes it particularly suitable for on-line sensor-based path planning. Once a new object is detected, a new inequality can be added before running the next iteration. When an old obstacle is passed, its corresponding inequality may also be deleted. Every time when an inequality is added or deleted, a new optimization problem is formed. The current variable is always treated as the initial point of the optimization problem and search starts again. This mechanism indicates this approach is efficient and particularly suitable for on-line path planning. Other advantages of the approaches include that mature techniques developed in nonlinear programming theory with guarantee of convergence, efficiency, and numerical robustness can be directly applied to the robot path planning problem. The semi-infinite constrained optimization approach with an adaptive objective function has the following advantages:

1) The robot does not need to be shrunk to a point, the obstacles do not need to be expended to Cspace, and the entire course of path planning can be carried out in real 3D space.
2) The goal point is guaranteed to be the only global minimum of the objective function.
3) The standard search techniques which have been developed for more than thirty years in the nonlinear programming field can be used.
4) The approach is suitable for on-line task planning.

The investigation carried out in this chapter has also indicated that robot path planning can be formulated in different ways. It's important to seek more efficient and realistic methods for problem formalization. Computational complexity analysis must be developed based on a proper problem formulation which considers enough constraints. Although the fundamentals for the nonlinear programming theory have existed for many years, they have not attracted enough attention for such applications. The context presented in this chapter covers a wide range of subjects such as robot kinematics, CAD, CAM, computer graphics and nonlinear programming theory, and a basic framework has been developed. Our treatment is consistent. The study presented in this chapter has shown its great potential as an on-line motion planner. The future work includes the extension of the principle developed here to the obstacle avoidance problem for manipulator without the calculation of Cspace obstacles, and the adoption of the interpolation techniques to deal with the local minima problem (Wang et al., 2000).

The constraints added by the kinematics and the shape of a manipulator are more complex than the subsea vehicle we have done in this chapter. In addition, for a car_like moving

robot, the nonholonomic constraints must be taken into account. Path planning without the Cspace computation and with the consideration of those practical issues is still the challenge we are facing.

7. Acknowledgement

The authors gratefully acknowledge the financial support from Chinese Academy of Sciences, P. R. China and Royal Society of United Kingdom for the joint research project under grant No. 20030389, 2003-2006, the National High-Tech Development 863 Program of China under Grant No. 2003AA1Z2220, the National Natural Science Fouondation of China under Grant No. 60373053, and the financial support from Chinese Academy of Sciences and Chinese State Development Planning Commission for Bai Ren Ji Hua (BRJH), 2002-2005.

8. References

Barr, A. (1981). Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1, 1, (January 1981), 11-23, ISSN: 0272-1716.

Berger, M. (1986). *Computer Graphics with Pascal*. Benjamin-Cummings Publishing Co., Inc., ISBN: 0805307915, California.

Blackmore, D.; Leu, M. & Wang, L. (1997). The sweep-envelope differential equation algorithm and its application to NC machining verification. *Computer Aided Design*, 29, 9, (September 1997), 629-637, ISSN: 0010-4485.

Blechschmidt, J. & Nagasuru, D. (1990). The use of algebraic functions as a solid modelling alternative: an investigation. *Proceedings of the 16th ASME Design Automation Conference*, pp. 33-41. Chicago, September 1990, ASME, New York.

Brooks, R. & Lozano-Perez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Trans. on Systems, Man and Cybernetics*, 1985, 15, 2, (March 1985), 224-233, ISSN: 1083-4427.

Chang, J. & Lu, H. (2001). Backing up a simulated truck via grey relational analysis. *Journal of the Chinese Institute of Engineers*, 24, 6, (November 2001), 745-752, ISSN: 0253-3839.

Chiyokura, H. (1988). *Solid Modelling with Designbase: Theory and Implementation*. Addison-Wesley Publishing Limited, ISBN: 0-201-19245-4, New York.

Comba, P. (1968). A procedure for detecting intersections of three-dimensional objects. *JACM*, 15, 3, (July 1968), 354-366, ISSN: 0004-5411.

Conn, R. & Kam, M. (1997). On the moving-obstacle path planning algorithm of Shih, Lee, and Gruver. *IEEE Trans. Systems, Man and Cybernetics, Part B: Cybernetics*, 27, 1, (February 1997), 136-138, ISSN: 1083-4419.

Connolly, I. (1997). Harmonic function and collision probabilities. *Int. J. Robotics Research*, 16, 4, (August 1997), 497-507, ISSN: 0278-3649.

Fletcher, R. (1987). *Practical Methods of Optimization*. John Wiley & Sons, Inc., ISBN: 0-471-91547-5, New York.

Franklin, W. & Barr, A. (1981). Faster calculation of superquadric shapes. *IEEE Computer Graphics & Application*, 1, 3, (July 1981), 41-47, ISSN: 0272-1716.

Gill, P.; Murray, W. & Wright, M. (1981). *Practical Optimization*. Academic Press, ISBN: 0-12-283952-8, London.

Hall, M. & Warren, J. (1990). Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics & Applications*, 10, 6, (November 1990): 33-42, ISSN: 0272-1716.

Haug, E.; Adkins, F. & Cororian, D. (1998). Domains of mobility for planar body moving among obstacles. *Trans. the ASME, Journal of Mechanical Design*, 120, 3, (September 1998), 462-467, ISSN: 1050-0472.

Hu, T.; Kahng, A. & Robins, G. (1993). Optimal robust path planning in general environment. *IEEE Trans. Robotics and Automation*, 9, 6, (December 1993), 755-774, ISSN: 1042-296X.

Huang, H. & Lee, P. (1992). A real-time algorithm for obstacle avoidance of autonomous mobile robots. *Robotica*, 10, 3, (May 1992), 217-227, ISSN: 0263-5747.

Hwang, Y. & Ahuja, N. (1992). A potential field approach to path planning. *IEEE Trans. on Robotics and Automation*, 8, 1, (February 1992), 23-32, ISSN: 1042-296X.

Khatib, O. (1986). Real-time obstacle avoidance for manipulator and mobile robots. *Int. J. Robotics Research*, 5, 1, (February 1986): 90-98, ISSN: 0278-3649.

Kreyszig, E. (1993). *Advanced Engineering Mathematics*. John Wiley & Sons, Inc., ISBN: 0471728977, New York.

Latombe, J. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, ISBN: 0-7923-9206-X, Boston.

Lozano-Perez, T. (1983). Spatial planning: A configuration space approach. *IEEE Trans. on Computers*, 32, 2, (February 1983), 108-120, ISSN: 0018-9340.

Lu, H. & Yeh, M. (2002). Robot path planning based on modified grey relational analysis. *Cybernetics and Systems*, 33, 2, (March 2002), 129-159, ISSN: 0196-9722.

Luenberger, D. (1984). *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, ISBN: 0-201-15794-2, London.

Lumelsky, V. (1991). A comparative study on path length performance of maze-searching and robot motion planning. *IEEE Trans. Robotics and Automation*, 7, 1, (February 1991), 57-66, ISSN: 1042-296X.

Oriolo, G.; Ulivi, G. & Vendittelli, M. (1998). Real-time map building and Navigation for autonomous robots in unknown environments. *IEEE Trans. Systems, Man and Cybernetics, PartB: Cybernetics*, 28, 3, (June 1998), 316-333, ISSN: 1083-4419.

Park, T.; Ahn, J. & Han, C. (2002). A path generation algorithm of an automatic guided vehicle using sensor scanning method. *KSME International Journal*, 16, 2, (February 2002), 137-146, ISSN:1226-4865.

Petillot, Y.; Ruiz, I. & Lane, D. (2001). Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar. *IEEE Journal of Oceanic Engineering*, 26, 2, (April 2001), 240-251, ISSN: 0364-9059.

Petillot, Y.; Ruiz, T.; Lane, D.; Wang, Y.; Trucco, E. & Pican, N. (1998). Underwater vehicle path planning using a multi-beam forward looking sonar. *IEEE Oceanic Engineering Society, OCEANS'98*, pp. 1194-1199, ISBN: 0-7803-5045-6, Nice, France, September 1998, IEEE Inc., Piscataway.

Polak, E. & Mayne, D. (1984). Control system design via semi-infinite optimization: a review. *Proceeding of the IEEE*, 72, 12, (December 1984): 1777-1793, ISSN: 0018-9219.

Rao, S. (1984). *Optimization, theory and applications*. John Wiley & Sons, Inc., ISBN: 0-470-27483-2, New York.

Ricci, A. (1973). A constructive geometry for computer graphics. *The Computer Journal*, 16, 2, (April 1973), 157-160, ISSN: 0010-4620.

Rose, E. (1957). *Elementary Theory of Angular Momentum*. John Wiley & Sons, Inc., ISBN: 0471735248, New York.

Rosen, K. (1991). *Discrete mathematics and Its Applications*. McGraw-Hill, Inc., ISBN: 0-07-053744-5, New York.

Ruiz, T.; Lane, D. & Chantler, M. (1999). A comparison of inter-frame feature measures for robust object classification in sector scan sonar image sequences. *IEEE Journal of Oceanic Engineering*, 24, 4, (October 1999), 458-469, ISSN: 0364-9059.

Shih, C. & Jeng, J. (1999). Stabilization of non-holonomic chained systems by gain scheduling. *International Journal of Systems Science*, 30, 4, (April 1999), 441-449, ISSN: 0020-7721.

Sundar, S. & Shiller, S. (1997). Optimal obstacle avoidance based on the Hamilton-Jacobi-Bellman equation. *IEEE Trans. Robotics and Automation*, 13, 2, (April 1997), 305 -310, ISSN: 1042-296X.

Tanak, Y.; Fukushima, M. & Ibaraki, T. (1988). A comparative study of several semi-infinite nonlinear programming algorithms. *European Journal of Operational Research*, 36, 1, (July 1988), 92-100, ISSN: 0377-2217.

The Math Works Inc. (1993). *MATLAB Optimization Toolbox User's Guide*.

Trucco, E.; Petillot, Y.; Ruiz, I.; Plakas, K. & Lane, D. (2000). Feature tracking in video and sonar subsea sequences with applications. *Computer Vision and Image Understanding*, 79, 1, (July 2000), 92-122, ISSN: 1077-3142.

Wang, W. & Wang, K. (1986). Geometric Modeling for swept volume of moving solids. *IEEE Computer Graphics & Applications*, 6, 12, (December 1986), 8-17, ISSN: 0272-1716.

Wang, Y. (1995). *Kinematics, motion analysis and path planning for four kinds of wheeled mobile robots* [Dissertation]. Dept. Mechanical Engineering, Edinburgh University.

Wang, Y. (1997). A note on solving the find-path problem by good representation of free space. *IEEE Trans. Systems, Man and Cybernetics, Part B: Cybernetics*, 27, 1, (February 1997), 723-724, ISSN: 1083-4419.

Wang, Y. & Cartmell, M. (1998a). Autonomous vehicle parallel parking design using function fitting approaches. *Robotica*, 16, 2, (March 1998), 159-170, ISSN: 0263-5747.

Wang, Y. & Cartmell, M. (1998b). Trajectory generation for four-wheel steering tractor-trailer system: a two step method. *Robotica*, 16, 4, (July 1998), 381-386, ISSN: 0263-5747.

Wang, Y. & Cartmell, M. (1998c). New Model for Passing Sight Distance on Two-Lane Highways. *ASCE J. of Transportation Engineering*, 124, 6, (November 1998), 536-545, ISSN: 0733-947X.

Wang, Y.; Cartmell, M.; Tao, Q. & Liu, H. (2005). A generalized real-time obstacle avoidance method without the Cspace calculation. *J. Computer Science & Technology*, 20, 6, (November 2005), 774-787, ISSN: 1000-9000.

Wang, Y. & Lane, D. (1997). Subsea vehicle path planning using nonlinear programming and constructive solid geometry. *IEE Proc., Part D, Control theory and applications*, 144, 2, (March 1997), 143-152, ISSN: 1350-2379.

Wang, Y. & Lane, D. (2000). Solving a generalized constrained optimization problem with both logic AND and OR relationships by a mathematical transformation and its application to robot path planning. *IEEE Trans. Systems, Man and Cybernetics, Part C: Application and Reviews*, 30, 4, (November 2000), 525-536, ISSN: 1094-6977.

Wang, Y.; Lane, D. & Falconer, G. (2000). Two novel approaches for unmanned underwater vehicle path planning: constrained optimization and semi-infinite constrained optimization. *Robotica*, 18, 2, (March 2000), 123-142, ISSN: 0263-5747.

Wang, Y. & Linnett, J. (1995). Vehicle kinematics and its application to highway design. *ASCE J. of Transportation Engineering*, 121, 1, (January 1995), 63-74, ISSN: 0733-947X.

Wang, Y.; Linnett, J. & Roberts, J. (1994a). Motion feasibility of a wheeled vehicle with a steering angle limit. *Robotica*, 12, 3, (May 1994), 217-226, ISSN: 0263-5747.

Wang, Y.; Linnett, J. & Roberts, J. (1994b). Kinematics, kinematic constraints and path planning for wheeled mobile robots. *Robotica*, 12, 5, (September 1994), 391-400, ISSN: 0263-5747.

Wang, Y.; Liu, H.; Li, M.; Wang, Q.; Zhou, J. & Cartmell, M. (2004). A real-time path planning approach without the computation of Cspace obstacles. *Robotica*, 22, 2 (March 2004), 173-187, ISSN: 0263-5747.

Xu, W. & Ma, B. (1999). Polynomial motion of non-holonomic mechanical systems of chained form. *Mathematical Methods in the Applied Sciences*, 22, 13, (September 1999), 1153-1173, ISSN: 0170-4214.

Zhang, Y. & Valavanis, K. (1997). A 3-D potential panel method for robot motion planning. *Robotica*, 15, 4, (July 1997), 421-434, ISSN: 0263-5747.

21

A Pursuit-Rendezvous Approach for Robotic Tracking

Fethi Belkhouche
School of arts and sciences
Texas A & M Int. University
Laredo, Texas
USA

Boumediene Belkhouche
EECS Department
New Orleans, LA 70118
USA

1. Introduction

Robot navigation in dynamic environments and tracking of moving objects are among the most important topics addressed by the robotics community. These problems are more difficult than classical navigation problems, where the robot navigates to reach a stationary object. More interesting and complex applications involve moving targets. For example, applications such as dynamic surveillance can benefit from the tracking and navigation towards a moving goal. In surveillance problems the aim is for a robot to keep an evader in the field of view of the robot's sensory system (which consists of vision sensors in most cases). While adequate solutions to the problem of navigation towards a stationary goal have been elaborated, the problem of navigation and tracking of moving objects is still an open problem. This problem is fairly new and much more difficult. Algorithms developed in the motion planning community are highly effective at computing open loop controls, but cannot provide closed loop systems. This makes these algorithms less appropriate for tracking and navigation towards a moving object. This problem is a real-time problem that requires a closed loop strategy. The problem of navigation towards a moving goal in the presence of obstacles is a more difficult problem. The problem combines both local and global aspects. The local navigation aspect deals with the navigation on a small scale, where the primary problem is obstacle avoidance. The global navigation aspect deals with a larger scale, where the problem resides in reaching the goal.

Various methods based on different strategies such as computer vision, fuzzy logic, and Lyapunov theory have been suggested to solve this problem. Methods used for tracking moving targets can be classified into two different families: model-based methods and feature-based methods. Model-based methods aim to build a model of the tracking problem. Feature-based methods track the features of the object. Vision-based methods

are among the most important feature-based methods. These methods are widely used for tracking and reaching moving objects (Tsai et al., 2003; Oh & Allen, 2001). Other authors consider the problem of tracking humans using a wheeled mobile robot. The suggested algorithms can be used in different surveillance applications. Feyrer and Zell (Feyrer & Zell, 2001) suggested an algorithm that allows the detection, tracking and pursuing of humans in real time. The navigation is based on a potential field method and the detection process is based on an approach that combines colour, motion, and contour information. An algorithm for tracking humans from a moving platform is suggested in (Davis et al., 2000). Visual servoing methods are also used to keep the target in the field of vision of the robot (Thuilot et al., 2002). This problem is related to the problem of positioning and localization of the robot with respect to the moving object (Kim et al., 2001; Chaumette et al, 1999).

Even though vision-based methods are widely used, they may suffer from the following drawbacks:

- Most vision-based methods use complex algorithms that are computationally expensive, especially to track fast moving objects.
- It is necessary to keep the moving object in the field of view of the camera. This requires camera calibration. However, this task is difficult for manoeuvring and fast moving targets.

Several solutions have been suggested to solve these problems. Data reduction and the use of fast algorithms are among the most used solutions.

The problem of cooperative hunting behaviour by mobile robots troops is considered in (Yamaguchi, 2003). Clearly, this problem evolves navigation towards the prey. This task is accomplished using a model-based method.

Even though vision sensors are the most used, other sensors such as LADAR sensors, and acoustic sensors (Parker et al., 2003) are also used for target tracking and interception. Sensor planning and control for optimally tracking of targets is discussed in (Spletzer & Taylor, 2003). The existence of a strategy to maintain a moving target within the sensing range of an observer is discussed in (Murrieta et al., 2003).

Methods from control theory are also used for target-tracking. In (Lee et al., 2000), Lyapunov theory is used to derive an asymptotically stable solution to the tracking problem. A combination between control theory and artificial potential field methods is discussed in (Adams, 1999), where the asymptotic stability is discussed in details.

Model-based methods can be divided into three main families:

- Methods based on artificial intelligence.
- Methods based on optimal control and differential games theory.
- Methods based on geometric and kinematics equations.

In many situations, the robot tracks an intelligent evader. In this case more elaborate control strategies are needed. Methods based on artificial intelligence are used mainly by researchers in the robotics community to pursue and keep a moving target in the field of view of the robot. Optimal control methods are widely used in the aerospace community. These methods require an estimation of the time-to-go, which is a difficult task in practice. Kinematics-based methods are based on the derivation of a kinematics model for the motion. Kinematics-based methods can be used in various applications in surveillance and domain coverage. One important advantage of model-based methods is that it is possible to implement these methods using different types of sensors.

Here, we suggest a novel approach for tracking of and navigation towards a moving object. Our approach is based on a combination of the kinematics equations with geometric rules. The goal is to derive a closed loop control law for the robot's orientation angle. Since the target's motion is not a-priori known to the robot, it is necessary to design an online control law. Our approach combines the pursuit law with a rendezvous strategy and consists of a closed loop strategy. Thus, the control law for the robot's orientation angle has two terms. The first term corresponds to the pursuit, and the second term corresponds to the rendezvous. In the pursuit behavior the robot tracks or follows the path of the target. In the rendezvous behavior the robot does not follow or move towards the goal, but it moves towards a point that both the robot and the goal will reach simultaneously.

Our navigation law depends on a real variable, which we call the navigation parameter. This parameter allows controlling the navigation law. Thus, we can obtain a pure pursuit behavior or a pure rendezvous behavior, or a combination between the pursuit and the rendezvous. The navigation parameter can be time-varying too. In the presence of obstacles, the navigation law is combined with an obstacle avoidance algorithm; therefore, the robot moves in two modes: the tracking mode and the obstacle avoidance mode. We also suggest studying the control law in terms of the optimality of the path traveled by the robot. The method presents various advantages over other classical methods such as:

- Robustness: Kinematics-based methods are well-known by their robustness.
- Model-based: Our method is model-based, which means that the method can be implemented using different types of sensors.
- Proof of correctness: Our method allows us to rigorously prove that the robot navigating under our control law will reach the target successfully.

The algorithm discussed here relies on a localization technique to determine the visibility angle. However, only the control loop is discussed. The localization problem and the influence of the sensory system are beyond the scope of this study.

2. Preliminaries: Geometry and kinematics

In this section, we introduce several important concepts and definitions. The workspace consists of a subset χ of IR^2. The robot and the goal are shown in figure 1. The reference point of the robot is denoted by R. The goal point is defined to be $(x_G, y_G) \subset \chi$. It is denoted by G. Let point O be the origin of an inertial reference frame of coordinates. With reference to figure 1, we define the following quantities:

1. The visibility line robot-goal: This is the imaginary straight line that starts at the robot's reference point and is directed towards the goal. This line is defined even in the presence of obscurities.
2. Based on the visibility line, we define the visibility angle denoted by η as shown in figure 1. η is a function of the robot's and the goal's coordinates.
3. The visibility angles for the robot and the goal are given by η_R and η_G, respectively.
4. r_R and r_G denote the Euclidian distances from the origin to the reference points of the robot and the goal, respectively. The relative distance robot-goal is denoted by r.
5. The robot's coordinates in the Cartesian frame are given by (x_R, y_R), and the goal's coordinates are given by (x_G, y_G). The position error vector is given by $[x_e, y_e]^T$, with $x_e = x_G - x_R$, $y_e = y_G - y_R$.

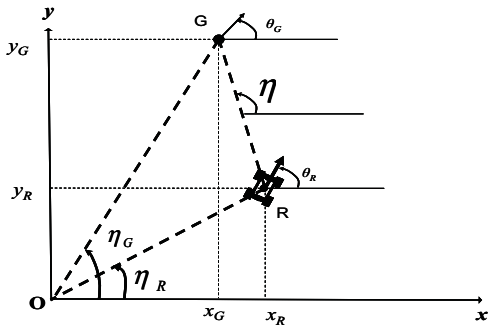


Fig. 1. A representation of the geometry.

The movement of the robot is controlled by its parameterized linear velocity $v_R(t)$ and angular velocity $\omega_R(t)$. The kinematics of the robot are described by the following equation:

$$\begin{aligned} \dot{x}_R &= v_R \cos(\theta_R) \\ \dot{y}_R &= v_R \sin(\theta_R) \\ x_R(t_0) &= x_{R0}, y_R(t_0) = y_{R0} \end{aligned} \tag{1}$$

with $\dot{\theta}_R = \omega_R$, where θ_R is the robot's orientation angle with respect to the positive x-axis. The kinematics equations describe the relationship between the control functions and the resulting trajectories. The goal is a moving point that moves according to the following kinematics equations:

$$\begin{aligned} \dot{x}_G &= v_G \cos(\theta_G) \\ \dot{y}_G &= v_G \sin(\theta_G) \\ x_G(t_0) &= x_{G0}, y_G(t_0) = y_{G0} \end{aligned} \tag{2}$$

where v_G is the goal's linear velocity and θ_G is the goal's orientation angle. v_G and θ_G are not a-priori known to the robot. However, it is assumed that the robot has a sensory system that allows to obtain these quantities in addition to the goal's position in real time. The kinematics model given by system (1) is in fact the kinematics model of a wheeled mobile robot of the unicycle type. The position of the robot in the reference frame of coordinates is given by the vector:

$$\begin{aligned} \vec{r}_R &= \overrightarrow{OR} = x_R \vec{u}_x + y_R \vec{u}_y \\ \vec{r}_R &= (r_R \cos \eta_R)\vec{u}_x + (r_R \sin \eta_R)\vec{u}_y \end{aligned} \tag{3}$$

In a similar way, the position of the goal is given by the vector:

$$\begin{aligned} \vec{r}_G &= \overrightarrow{OG} = x_G \vec{u}_x + y_G \vec{u}_y \\ \vec{r}_G &= (r_G \cos \eta_G)\vec{u}_x + (r_G \sin \eta_G)\vec{u}_y \end{aligned} \tag{4}$$

where $\vec{u}_x, \vec{u}_y$ are the unit vectors along the x- and y-axes, respectively. The time derivative of $\vec{r}_R$ and $\vec{r}_G$ gives the velocity vectors as follows:

$$\begin{aligned} \dot{\vec{r}}_R &= \dot{x}_R \vec{u}_x + \dot{y}_R \vec{u}_y \\ \dot{\vec{r}}_G &= \dot{x}_G \vec{u}_x + \dot{y}_G \vec{u}_y \end{aligned} \tag{5}$$

By taking the time derivative of x_R, y_R, x_G, and y_G, we obtain:

$$\begin{aligned}
\dot{x}_R &= \dot{r}_R \cos\eta_R - r_R\dot{\eta}_R \sin\eta_R \\
\dot{y}_R &= \dot{r}_R \sin\eta_R + r_R\dot{\eta}_R \cos\eta_R \\
\dot{x}_G &= \dot{r}_G \cos\eta_G - r_G\dot{\eta}_G \sin\eta_G \\
\dot{y}_G &= \dot{r}_G \sin\eta_G + r_G\dot{\eta}_G \cos\eta_G
\end{aligned} \tag{6}$$

By replacing $\dot{x}_R, \dot{y}_R$ by their values in the kinematics model of the robot, we obtain:

$$\begin{aligned}
\dot{r}_R &= v_R \cos(\theta_R - \eta_R) \\
\dot{\eta}_R &= v_R \sin(\theta_R - \eta_R) r_R^{-1}
\end{aligned} \tag{7}$$

By replacing $\dot{x}_G, \dot{y}_G$ by their values in the kinematics model of the goal, we obtain:

$$\begin{aligned}
\dot{r}_G &= v_G \cos(\theta_G - \eta_G) \\
\dot{\eta}_G &= v_G \sin(\theta_G - \eta_G) r_G^{-1}
\end{aligned} \tag{8}$$

Now we consider the relative range between the robot and the goal which is given by

$$\vec{r} = \vec{r}_G - \vec{r}_R \tag{9}$$

Its time derivative gives the relative velocity

$$\dot{\vec{r}} = \dot{\vec{r}}_G - \dot{\vec{r}}_R \tag{10}$$

The relative velocity can be decomposed into two components along and across the visibility line robot-goal. This allows us to obtain:

$$\begin{aligned}
\dot{r} &= v_G \cos(\theta_G - \eta) - v_R \cos(\theta_R - \eta) \\
\dot{\eta} &= (v_G \sin(\theta_G - \eta) - v_R \sin(\theta_R - \eta)) r^{-1}
\end{aligned} \tag{11}$$

The relative kinematics model given by system (11) is very important in the formulation of our navigation-tracking problem. This model gives a good description of the motion of the goal as seen by the robot. The first equation gives the relative distance robot-goal. A decreasing range corresponds to $\dot{r} < 0$. The second equation gives the rate of turn of the goal with respect to the robot. The sign of $\theta_G - \eta$ indicates whether the goal is approaching or moving away from the robot. For $\theta_G - \eta \in]-\pi/2, \pi/2[$ the goal is moving away from the robot. For $\theta_G - \eta \in]\pi/2, 3\pi/2[$ the goal is approaching from the robot. System (11) is highly nonlinear. Its solution gives the robot's path in the plane (r, η). However, the analytical solution is difficult except in a few particular cases.

3. Problem statement

The workspace is cluttered with N stationary obstacles denoted by B_i, $(i=1,..,N)$. The robot moves in the workspace according to the kinematics equations given by (1). The path of the robot is given by $P_R(t)$. The path of the moving goal is given by $P_G(t)$. This path is not a-priori known to the robot. Our goal is to design a closed loop control law for the robot in order to reach the goal and avoid possible collisions with obstacles. This can be stated as follows:

$$\begin{aligned} &\left\| P_G\left(t_f\right) - P_R\left(t_f\right) \right\| < \varepsilon \\ &P_R\left(t\right) \cap B_i = \Phi \end{aligned} \tag{12}$$

where ε is a small real number and t_f is the interception time. We will also design a control law for the robot linear velocity to keep the goal within a given coverage range from the robot.

It is assumed that

1. The control input for the robot is $[v_R, \theta_R]$ instead of $[v_R, \omega_R]$.
2. The robot is faster than the goal. This means that $v_R > v_G$.
3. The robot does not have a pre-decided knowledge of the environment. However, it has a sensory system that allows detecting obstacles and obtaining the necessary information on the goal. As we mentioned previously, the influence of the sensory system is beyond the scope of this study.

This problem is difficult, because it combines two different aspects, navigation towards the goal and obstacle avoidance. Navigation towards the goal has a global aspect while obstacle avoidance has a local aspect. The goal can perform two different types of motion, namely accelerating motion, and non-accelerating motion. In the case of an accelerating motion, either the linear velocity or the orientation angle of the goal is time- varying. In the case of a non-accelerating motion, both θ_G and v_G are constant. It is more difficult for the robot to reach an accelerating goal.

4. The control law

As we mentioned previously there exist two approaches for navigating a robot towards a moving object. The first approach is the pursuit (Belkhouche & Belkhouche, 2005), the second approach is the rendezvous. In the pursuit, the robot follows the moving object directly. That is the robot is always heading towards the goal at any time. This is the most obvious way to reach a moving goal. Various sensor-based algorithms use the pursuit even though they do not model the problem using the pursuit equations. The rendezvous approach uses a completely different principle which is the opposite extreme of the pursuit. In the rendezvous, the robot does not follow the path of the goal, but it moves towards a point where the robot and the goal will arrive at the same time. To accomplish this task the robot moves in lines that are parallel to the initial line of sight. The strategy discussed in this chapter is a new strategy that combines the pursuit and the rendezvous. In this approach, the robot orientation angle is given by the following equation:

$$\theta_R = \eta + c \sin^{-1}\left(K_v \sin\left(\theta_G - \eta\right)\right) \tag{13}$$

where c is a real number that satisfies $0 \le c \le 1$. c is the control variable of the robot's orientation angle. $K_v = v_G / v_R$ is the speed ratio. Equation (13) is a closed loop control law, where the control input depends on the state of the system, mainly the state of the goal. For $c = 0$, the control law acts like the pure pursuit. For $c = 1$, the control law acts like the pure rendezvous. For $0 < c < 1$, the control law has a behaviour that combines the rendezvous and the pursuit. For example, $c=0.9$ corresponds to 90% rendezvous and 10% pursuit. The control law given by (13) allows to reach the goal from any arbitrary initial position when the assumptions stated previously are satisfied. The main result concerning the navigation using the control law given by (13) is stated as follows:

Proposition:
Under the control law given by (13), the robot reaches the goal from any initial state when $v_R > v_G$.

Proof
The proof is based on the differential equation of the relative range rate; recall that $\dot{r}_R < 0$ corresponds to a decreasing range. We put

$$\lambda = \sin^{-1}(K_v \sin(\theta_G - \eta)) \tag{14}$$

Under the control law the relative range varies as follows:

$$\dot{r} = v_G \cos(\theta_G - \eta) - v_R \cos(c\lambda) \tag{15}$$

Recall that under the assumption that the robot is faster than the goal we have $K_v < 1$. The inverse of the sine function maps the domain [-1, 1] to [-π/2, π /2], and since $K_v < 1$, we have

$$\lambda = \sin^{-1}(K_v \sin(\theta_G - \eta)) \in]-\pi/2, \pi/2[\tag{16}$$

The cosine function of λ is strictly positive, i,e,

$$\cos(\lambda) > 0 \tag{17}$$

Since the cosine function of λ is strictly positive and $0 \le c \le 1$, it turns out that

$$\cos(\lambda) \le \cos(c\lambda) \tag{18}$$

Since $K_v < 1$, we have

$$\cos(\theta_G - \eta) < \cos(c\lambda) \tag{19}$$

Therefore

$$\begin{aligned} &\dot{r} = v_G \cos(\theta_G - \eta) - v_R \cos(c\lambda) < 0 \\ &\forall \theta_G \end{aligned} \tag{20}$$

Thus, since $\dot{r} < 0$ under the control law, the relative distance between the robot and the goal is a decreasing function of time.

■

There exist major differences in the behaviour of the control law for different values of *c*. For $c = 0$, the path of the robot is more curved near the interception. Thus more corrections are required near the interception. The opposite is true for $c = 1$, where the path of the robot is more curved at the beginning. This requires more corrections at the beginning of the navigation process. These aspects are discussed in the simulation. The control law becomes a simple pure pursuit when the goal is stationary.

4.1 Heading regulation

The initial state of the robot's orientation angle is given by $\theta_{R0} = \theta_R(t_0)$. In most cases, the value of θ_R given by the control law is different from θ_{R0}. Our goal is to design a smooth feedback plan that solves the planar navigation problem. For this reason a heading regulation is necessary. The heading regulation is accomplished by using the following formula:

$$\begin{aligned} \dot{\theta}_R &= -K_\theta\left(\theta_R - \theta_R^{des}\right) \\ \theta_R(t_0) &= \theta_{R0} \end{aligned} \tag{21}$$

where θ_R^{des} is given by the control law in equation (13). K_θ is a real positive number. The heading regulation is a transition phase that allows putting the robot in a configuration where the application of the control law is possible. The heading regulation phase is used whenever modes are switched during the collision avoidance process. Heading regulation is illustrated in figure 2 for both the pure pursuit (PP) and pure rendezvous (PR). The robot initial orientation angle is given by 90^0.

4.2 The pure rendezvous

As we mentioned previously, the pure rendezvous corresponds to $c = 1$. By replacing θ_R by its value in the equation of the visibility angle rate, we obtain

$$\dot{\eta} = 0 \tag{22}$$

which implies that the visibility angle is constant, i.e., η=*constant*. This is the most important characterization of the pure rendezvous law. As a result, the motion of the goal as seen by the robot is linear, meaning that the robot moves in a straight line if the goal is moving in a straight line. The visibility angle is given by

$$\tan\eta = \frac{y_e}{x_e} \tag{23}$$

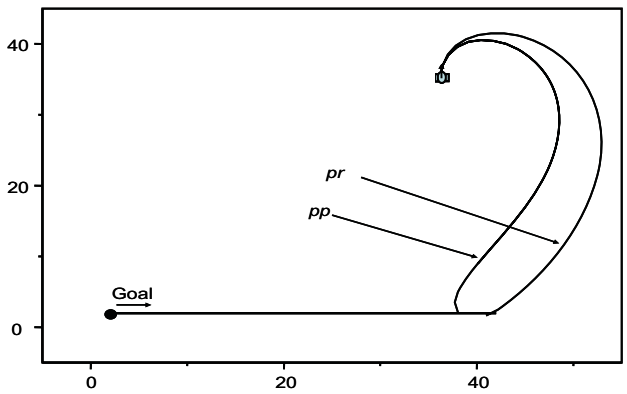


Fig. 2. An illustration of the heading regulation for the PP and PR. The robot's initial orientation angle is 90^0.

which is constant under the pure rendezvous. By taking the time derivative we obtain

$$\tan\eta = \frac{\dot{y}_e}{\dot{x}_e} \tag{24}$$

This allows us to write

$$\frac{y_e}{x_e} = \frac{\dot{y}_e}{\dot{x}_e} \tag{25}$$

This equation is another important equation that characterizes the pure rendezvous. The orientation angle under the pure rendezvous is constant when the goal is not accelerating, and the robot is moving with a constant linear velocity. This is stated in the following result.

Proposition:
Under the pure rendezvous with v_R=*constant*, the robot's orientation angle is constant for non-manoeuvring goals.

Proof:
The first step resides in proving that the visibility angle is constant under the pure rendezvous. By replacing θ_R by its value, we obtain

$$\dot{\eta} = \left(v_G \sin\left(\theta_G - \eta\right) - v_R \sin\left(c \sin^{-1}\left(K_v \sin\left(\theta_G - \eta\right)\right)\right)\right) r^{-1} \tag{26}$$

It turns out that under the pure rendezvous (*c=1*), we have $\dot{\eta} = 0$, therefore $\eta = const$. As a result, for a non-manoeuvring target and a robot moving with constant speed we have θ_R=constant. Thus the robot moves in a straight line.

■

4.3 The pure pursuit

The pure pursuit is another important particular case. It corresponds to *c = 0*, and thus the robot's orientation angle is equal to the visibility angle. The kinematics equations under the pure pursuit are given by

$$\begin{aligned} \dot{r} &= v_G \cos\left(\theta_G - \eta\right) - v_R \\ \dot{\eta} &= \left(v_G \sin\left(\theta_G - \eta\right)\right) r^{-1} \end{aligned} \tag{27}$$

It is clear from the first equation in the system that the range rate is negative when $v_G < v_R$. Unlike the pure rendezvous, the visibility angle is not constant in the case of the pure pursuit. In fact, in the pure pursuit the visibility angle tracks the goal's orientation angle with time.
The pure pursuit and the pure rendezvous are illustrated in figures 3 and 4. The difference in the path is obvious. Note that in the case of the pure rendezvous, the visibility line angles are parallel to each other. A more detailed comparison is shown in our simulation.

4.4 Navigation with time varying navigation parameter

We have seen that the navigation parameter enables us to control the navigation law between two extreme strategies. Previously we have considered only constant values of c. However the navigation parameter can be time varying too. This property is used to combine the advantages of the pure pursuit with those of the rendezvous in one navigation law. It is possible to use different formulae for *c (t)*. Two possibilities are the following:

$$c(t) = 1 - e^{-bt} \tag{28}$$

and

$$c(t) = e^{-bt} \tag{29}$$

where b is a real positive number. By using equation (28), the navigation law acts like the pure pursuit near the initial state and like the pure rendezvous near the interception. The opposite is true with equation (29). It is also important to note that equations (28) and (29) can be used for transition between the pure pursuit and the pure rendezvous. Smaller values of b are required in this case.

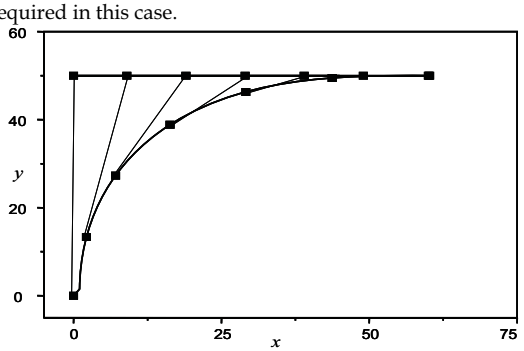


Fig.3. An illustration of the pure pursuit.

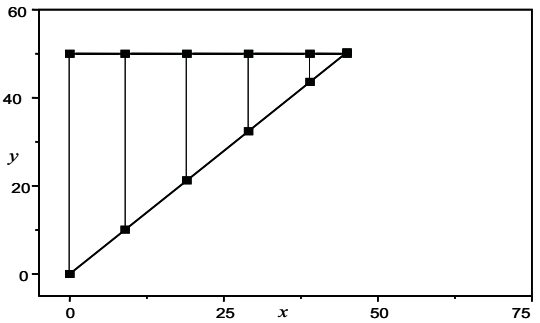


Fig. 4. An illustration of the pure rendezvous.

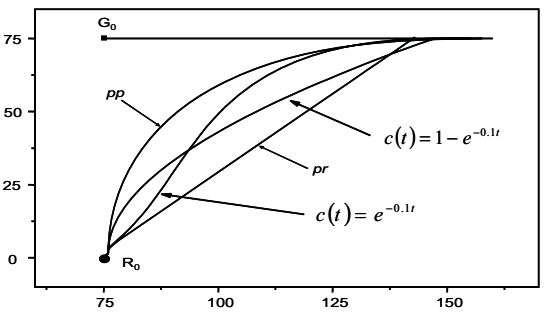


Fig. 5. An illustration of the of c(t) to switch between the pure pursuit (PP) and the pure rendezvous (PR).

4.5 The time-to-go

The time-to-go is the time it takes the robot to reach the moving goal. The time-to-go is very important for any comparison between control strategies. The time-to-go can be estimated by the following equation:

$$t_{to-go} = -\frac{r}{\dot{r}} \tag{30}$$

In general, it is difficult to estimate the time-to-go since it depends on many factors that are time-varying. The most important factors are the velocity ratio, and the target manoeuvers. The time-to-go may be used to determine the appropriate value of b to adjust $c(t)$. The only case where it is possible to find the time-to-go analytically is when the goal moves in a straight line, ($\theta_G = const$), v_R and v_G are constant, and the robot is applying a pure rendezvous approach. In this case, the time-to-go is given by

$$t_{to-go} = -\frac{r_0}{v_G \cos(\theta_G - \eta) - v_R \cos(c\lambda)} \tag{31}$$

It is obvious that the time-to-go is proportional to the initial range.

5. In the presence of obstacles

It is clear that the problems of navigation and reaching a moving object in the presence of obstacles are among the most difficult problems in robotic navigation. They combine local path planning for collision avoidance with global path planning for reaching the goal. In our formulation, the robot moves in two modes, the navigation mode and the obstacle avoidance mode. Clearly, the obstacle avoidance mode has the priority. The collision avoidance is accomplished by building a polar histogram of the environment. The polar histogram is based on the angular information obtained from the sensory system. Only obstacles that appear within a given region called the active region are considered. The polar histogram allows determining free directions and directions corresponding to the obstacles. A snapshot of the local environment from a given position of the robot is a characterization of the visible obstacles and the angles they make with the robot.

The first stage in the polar histogram is to represent the robot's surrounding environment using angular information provided by the robot's onboard sensors. The angles λ_{i1} and λ_{i2} are the limit angles characterizing obstacle B_i as shown in figure 7. The polar diagram denoted by D is obtained as follows:

$$D = \sum_{i=1}^{k} d_i \tag{32}$$

where k denotes the number of obstacles in the active region, and d_i is given as follows:

$$d_i = 1 \text{ if } \theta_R - \eta \in [\lambda_{i1} - \eta, \lambda_{i2} - \eta] \tag{33}$$

$d_i = 0$ otherwise

Note that the polar histogram is constructed based on the angle $\beta = \theta_R - \eta$, therefore the pure pursuit corresponds to $\beta = 0$, and the pure rendezvous corresponds to $\beta = \lambda$. The obstacle avoidance mode is activated when at least one obstacle appears in the active region, and the robot navigates by using the polar histogram. It is also easy to represent the goal's

orientation angle in the polar histogram. The robot deviates from its nominal path only if an obstacle appears in its path. The algorithm for collision avoidance is the following:

Procedure Deviation

1. Choose an intermediary point M such that $\eta_M - \eta$ has the same sign as $\theta_G - \eta$. η_M is the visibility angle between the robot and point M.
2. Navigate towards this point using the pure pursuit. A heading regulation procedure is used to keep the smoothness of the path. The equation for the heading regulation is similar to (21).

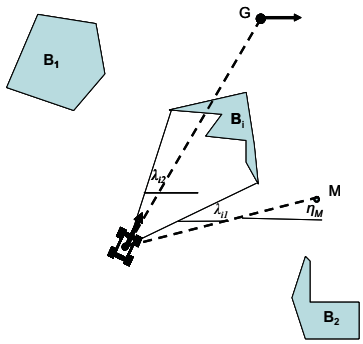


Fig. 6. Collision avoidance.

Collision avoidance algorithm:

1. If obstacle detected within the active region, then the collision avoidance mode is activated.
2. If the robot is in a collision course with obstacle B_i, then call procedure deviation
3. After obstacle passed go back to the pursuit-rendezvous mode. Since $\eta_M - \eta$ and $\theta_G - \eta$ have the same sign, the robot orientation angle and the goal orientation angle are on the same side of the visibility line.

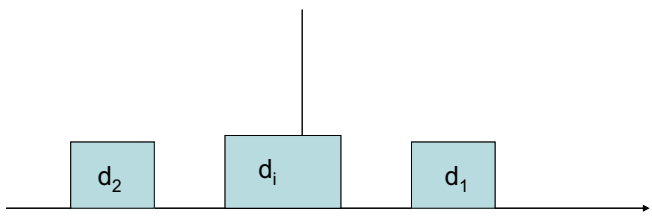


Fig. 7. Polar histogram for the environment in figure 6.

6. Pursuit-rendezvous for target dynamic coverage

Dynamic target coverage by a wheeled mobile robot or a group of mobile robots has been considered in the literature recently. This problem is important in various applications, such

as cleaning, security and patrolling, and sensor network deployment. Dynamic target coverage aims to generate a trajectory and the corresponding linear velocities. In the previous section, we designed a control law that allows the robot to reach the moving goal from an arbitrary initial state. In this section our goal is to design a second control law to keep the moving object within a given distance from the robot so that the goal stays in the robot's field of view. That is,

$$r_1^{des} \le r(t) \le r_2^{des} \tag{34}$$

with $r_1^{des} \le r^{des} \le r_2^{des}$. r^{des} is the desired value of the coverage range, r_1^{des} and r_2^{des} are the range limits for r^{des}. The coverage range is represented by a circle as shown in figure 8. Dynamic coverage is necessary in various surveillance and tracking applications. For example, in many situations it is important to keep the goal in the field of view of the robot's sensory system. To accomplish this task, it is necessary to design a control law for the robot's linear velocity. Note that a constant range between the robot and the moving object corresponds to $\dot{r} = 0$; that is,

$$v_G \cos(\theta_G - \eta) = v_R \cos(c\lambda) \tag{35}$$

In order to combine the navigation mode with the tracking at a constant distance mode, we use the method which is known as feedback linearization (Drakunov et. 1991) in combination with backstepping or block control (Drakunov et. 1991) which gives

$$\dot{r} = -K_r \left(r - r^{des}\right) \tag{36}$$

where K_r is a real positive number. Equation (36) allows to drive the relative range smoothly to its desired coverage range. By replacing $\dot{r}$ by its value, we obtain

$$v_G \cos(\theta_G - \eta) - v_R \cos(c\lambda) = -K_r \left(r - r^{des}\right) \tag{37}$$

From which the relative velocity of the robot can be obtained as follows:

$$v_R = \frac{K_r \left(r - r^{des}\right) + v_G \cos(\theta_G - \eta)}{\cos(c\lambda)} \tag{38}$$

The system converges to a steady state that satisfies equation (35). We have the following remarks concerning equation (38):

1. The term $K_r (r - r^{des})$ goes to zero with time.
2. If the goal applies a pure escape strategy, then $\theta_R = \eta$ and $v_R = v_G$. This is true for both the pure pursuit and the pure rendezvous.
3. In general, the required value of v_R is smaller in the case of the pure pursuit.

In the case of the pure pursuit, the dynamic coverage of a target is characterized by an important property, which can be states as follows:

Proposition

Under the pure pursuit, the dynamic coverage is characterized by $\theta_R \to \eta$ and $v_R \to v_G$. This means that the robot's orientation angle will track the target's orientation angle, and the robot's linear velocity will track the target's linear velocity.

Proof
The kinematics model under the pure pursuit is written as

$$\begin{aligned} \dot{r} &= v_G \cos(\theta_G - \eta) - v_R \\ \dot{\eta} &= (v_G \sin(\theta_G - \eta))r^{-1} \end{aligned} \tag{39}$$

The equilibrium position for the second equation is given by $\eta^* = \theta_T$. By using the classical linearization, it turns out that this equilibrium position is asymptotically stable. Therefore, $\eta \to \theta_G$, since $\eta = \theta_R$ under the pure pursuit, which gives $\theta_R \to \theta_G$. From equation (38) under the pure pursuit ($c = 0$), we have $v_R \to v_G$ as $\theta_R \to \theta_G$.

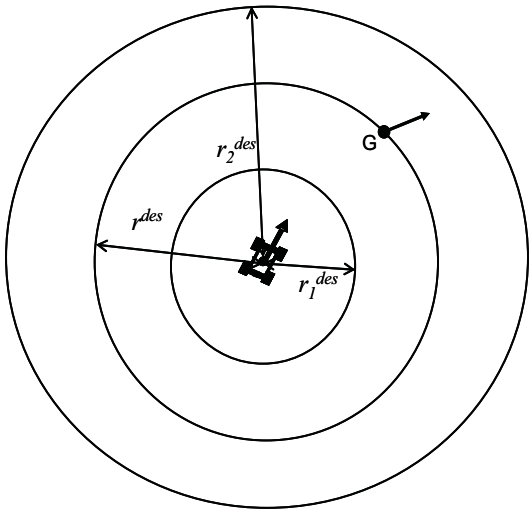


Fig. 8. An illustration of the dynamic coverage ranges.

7. Simulation

Here we consider several simulation examples to illustrate the suggested approach.

Example 1: A comparison between the pure pursuit (PP) and the pure rendezvous (PR)
Three scnearios are shown here. The first scneario shown in figure 9 corrresponds to a goal moving in a straight line. The second scenario shown in figure 12 corresponds to a goal moving in a circle. The third scenario is shown in figure 13, the goal moves in a sinusoidal motion, which is among the most difficult paths to reach. Note that the path of the goal is not a-priori known to the robot. For the scenario of figure 9, the visibility angle is shown in figure 10, and the robot orientation angle in figure 11. From figure 10, the visibility angle is constant under the PR. From figure 11, it is clear that more corrections and manoeuvers are required under the PP. Figure 14 shows the robot path for different values of c. In all cases the robot reaches the goal successfully.

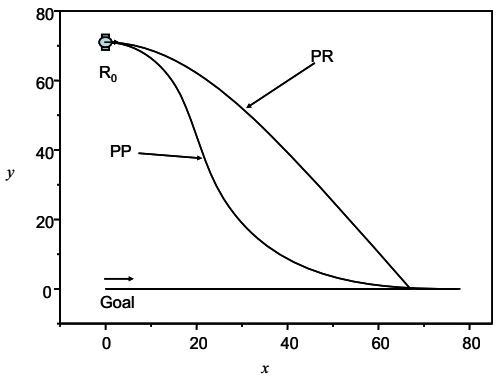


Fig. 9. Reaching a goal moving in a line.

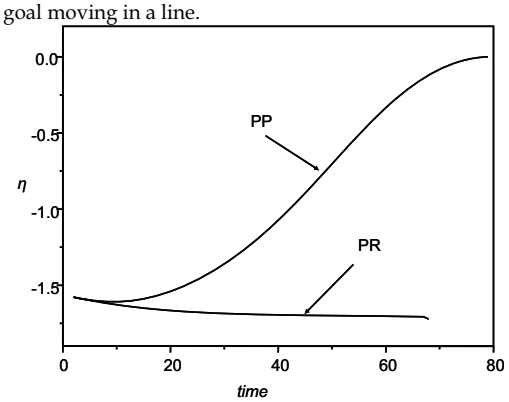


Fig. 10. Evolution of the visibility angle for the scenario of figure 9.

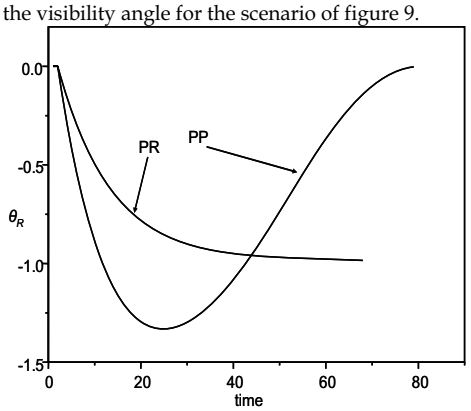


Fig. 11. Evolution of the robot's orientation angle for the scenario of figure 9.

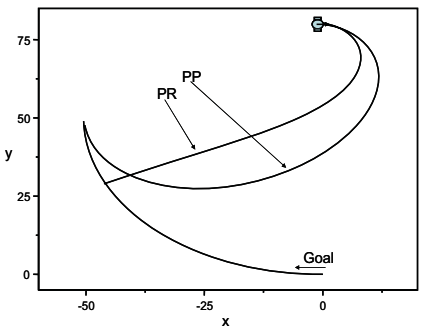


Fig. 12. Reaching a goal moving in circle.

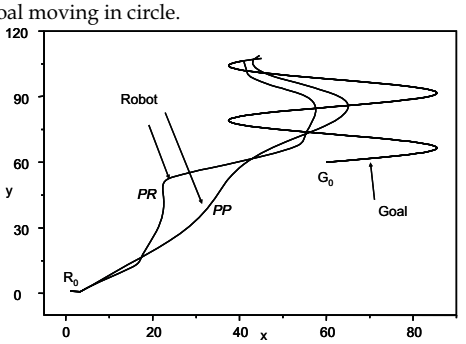


Fig. 13. Reaching a goal moving in a sinusoidal motion.

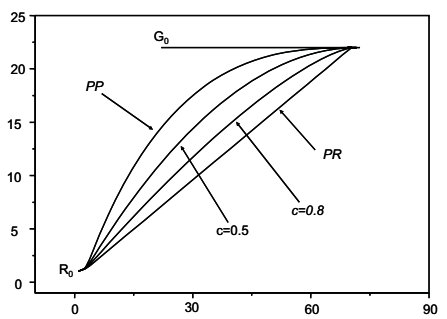


Fig. 14. Robot' path for different Values of c.

Example 2: in the presence of obstacles:

Two scenarios are shown in figures 14 and 15 to illustrate the navigation towards a moving goal in the presence of obstacles. The paths of the robot under the PP and the PR are different as shown in the figures. The robot accomplish the navigation and obstacle avoidance tasks successfully.

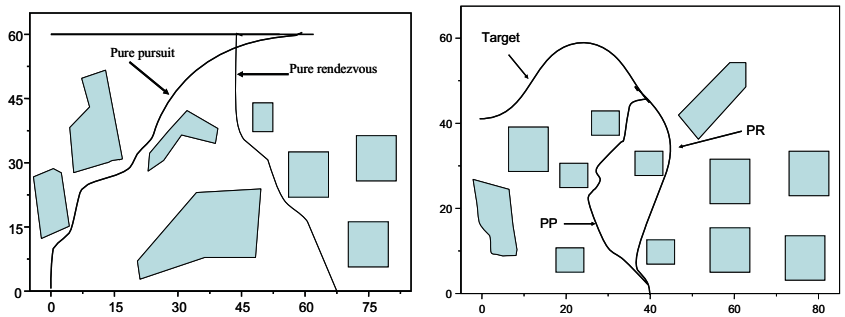


Fig. 15. Tracking and navigation in the presence of obstcales, goal moving in a line presence of obstacles.

8. Conclusion

We presented a method for robotic navigation and tracking of an unpredictably moving object. Our method is kinematics-based, and combines the pursuit law with the rendezvous law. First a kinematics model is derived. This kinematics model gives the motion of the goal with respect to the robot. The first equation gives the range rate between the robot and its goal. The second equation gives the turning rate of the goal with respect to the robot. The control law is then derived based on this kinematics model. This law is controlled by a real variable, which may be constant or time-varying. The most important properties of the control law are discussed. The dynamic coverage of the target is also discussed, where a second law for the robot's linear velocities is derived.

9. References

Adams, M.D. (1999) High speed target pursuit and asymptotic stability in mobile robotics. *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 2, pp. 230-236.

Chaumette, F.; Rives P. and Espiau, B. (1991) Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2248-2253 California.

Davis, L; Philomin, V. and Duraiswami, R. (2000). Tracking humans from a moving platform, *Proceedings of IEEE International Conference on Pattern recognition*, pp. 171-178.

Drakunov, S.; Izosimov, D.; Lukjanov, A.; Utkin, V.I. and Utkin, V. Block control principle,. *Automation and Remote Control*, Vol. 51, No. 6, pp. 737.746, 1991.

Drakunov, S.; Izosimov, D.; Lukjanov, A.; Utkin, V.I. and Utkin, V. Block control principle,. *Automation and Remote Control*, Vol. 51, No. 5, pp. 601.608, 1991.

Feyrer, S. and Zell, A. (1999). Detection, tracking, and pursuit of humans with an autonomous mobile robot. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 864-869.

Feyrer, S. and Zell, (1999). Tracking and pursuing persons with a mobile robot P*roceedings of the International Workshop on Recognition, Analysis and Tracking Faces and Gestures in Real time Systems*, pp. 83-88, California.

Kim, B.H.; Roh, D.K; Lee, J.M; Lee, M.H; Son, K; Lee, M.C; Choi, J.W; and Han, S.H. (2001) Localization of a mobile robot using images of a moving target, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 253-258 Seoul.

Lee, S.O.; Cho, Y.J.; Hwang-Bo, M.; You, B.J. and Oh, S.R.(2000) A stable target-trackingcontrol for unicycle mobile robots, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1822-1827.

Murrieta, R.; Sarmiento, A. and Hutchinson, S. (2003) On the existence of a strategy to maintain a moving target within the sensing range of an observer reacting with delay, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1184-1191.

Oh, P.Y. and Allen, P.K. (2001). Visual servoing by partitioning degrees of freedom. *IEEE Transactions on Robotics and Automation*, Vol. 17; No. 1, 1-17.

Thuilot, B; Martinet, P; Cordesses, L. and Gallice, J. (2002) Position based visual servoing: Keeping the object in the field of vision, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp 1624-1629 Washington DC.

Tsai, M ; Chen, K; Cheng, M; and Lin, K. (2003). Implementation of a real-time moving object tracking system using visual servoing. *Robotica*, Vol. 21, No. 6, 615-625.

Parker, L.; Birch, B.; Reardon, C. (2003) Indoor target intercept using an acoustic sensor network and dual wavefront path planning, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 278-283.

Spletzer, J.and Taylor,C. (2003) Dynamic Sensor Planning and Control for Optimally Tracking Targets, *The International Journal of Robotics Research*, Vol. 22, No. 1, 7-20.

Yamaguchi, H. (1999) A cooperative hunting behavior by mobile-robot troops, The International Journal of Robotics Research, Vol. 18, No. 9, 931-940.

Belkhouche F. and Belkhouche B. (2005) A method for robot navigation toward a moving goal with unknown maneuvers, Robotica, Vol. 23, No 6, 709-720.

22

Sensor-based Global Planning for Mobile Manipulators Navigation using Voronoi Diagram and Fast Marching

S. Garrido, D. Blanco, M.L. Munoz *, L. Moreno and M. Abderrahim
University Carlos III of Madrid, Spain

** Polytechnic University of Madrid, Boadilla del Monte, Spain.*

1. Introduction

Mobile Manipulators are special cases of mobile robots that need a careful consideration when planning their motion. Due to their higher and changing centre of gravity caused by the position of the manipulator and its configuration, abrupt motion or change of direction can have dangerous consequences and the toppling of the whole mobile manipulator. Therefore, the motion planning process has to produce smooth and safe trajectories. Since the 1980's mobile robot motion planning problems have been an important research topic that has attracted the attention of many researchers who worked extensively to obtain efficient methods to solve these problems. Such problems have been approached in two general ways: one approach has concentrated on solving motion planning problems by considering a previously known global environment or obstacles information and robot characteristics; the second approach has focused on planning motions by using local sensor information and robot characteristics. The first approach has been used extensively at a global planning level in robotics to plan trajectories from one environment location to another. This plan functions under the assumption of a perfectly controlled and modelled environment that in most situations does not exist. Unexpected obstacles, persons or moving elements make this approach difficult to utilize except in robustly controlled environments, such as in industrial manipulation environments (manufacturing). The second approach became evident in mobile robotics from the very beginning. A mobile robot's environment presents unexpected objects which makes it impossible to use the first approach exclusively. The second approach produces local plans by using local sensor information to avoid obstacles. Obviously a local plan is a solution for a local scale problem and needs to be integrated with a global planner or with global information to guarantee the existence of a solution to the global problem. This method received different names such as collision avoidance methods, local planners, and navigation methods.

To navigate in complex environments, an autonomous mobile robot needs to reach a compromise between the need for reacting to unexpected events and the need for efficient and optimized trajectories. Usually, path planning methods work in two steps: in the first

step, a global path over an existing map is calculated, and in the second step, the robot reads the sensor data and modifies the local trajectory in a reactive way. This provides the path adaptation mechanism to cope with unexpected obstacles or dynamic environments. These methods calculate the global trajectory off-line in a priori known map, and while the robot is moving local modifications are made continuously based on the sensor data.

The reason for using a two level planning approach is due to high computational cost that is required in most motion planning techniques to achieve an updated environment model (and to plan a smooth and dynamically adapted trajectory).

The use of a two level planner strategy decreases computational cost by activating the global planner occasionally (it can be done off line) while the local planner which is much faster runs on line. This two level approach affects the control architecture of the mobile robot. The first mobile robots were based on a sense-model-plan scheme referred to in the literature as planned architectures (Meystel, 1986). These architectures present some difficulties providing fast responses to environmental changes.

Posterior reactive architectures (Brooks, 1986) have the advantage of using fast response methods based on a sensor-decision-action scheme to react to environmental changes, but also show the difficulty of extending the reactivity to upper levels. Finally, hybrid deliberative/reactive architectures (Arkin, 1990), (Arkin, 1998), (Alami et al., 1998), (Chatila, 1995), (Bonasso et al., 1996) and (Low et al., 2002) have emerged as the result of recognizing the advantages provided with planning at high control levels and reactive architectures at lower control levels.

The Voronoi Fast Marching method is based on a sensor-based global path planning paradigm. This is a planning approach based on a fast sense-model-plan scheme able to integrate sensor information in a simple grid based environment model and to calculate a globally consistent, smooth and safe trajectory, fast enough to be used as a reactive navigation method. This approach presents some advantages. One is the ability of global planning methodologies to guarantee a path between a given point and the goal point, if one exists. And the other is the smoothness and safety of the solution obtained. This solution eliminates the local minima trap problem and the oscillations in narrow places present in other methods, and also indirectly eliminates the use of a supervision system able to detect local minima failures (obstructed paths), in order to initiate the search for a new and feasible global path from the current position to the goal point.

To calculate the trajectory, the proposed method combines the Voronoi Distance transform and the Fast Marching Method. The Voronoi approach to path planning has the advantage of providing the safest trajectories in terms of distance to obstacles, but because its nature is purely geometric it does not achieve enough smoothness. On the other hand, the Fast Marching Method has been applied to path planning (Sethian, 1996a), and their trajectories are of minimal distance, but they are not very safe because the path is too close to obstacles, and more importantly the path is not smooth enough. In order to improve the safety of the trajectories calculated by the Fast Marching Method, two solutions are possible. The first possibility, in order to avoid unrealistic trajectories, produced when the areas are narrower than the robot, the segments with distances to the obstacles and walls less than half the size of the robot need to be removed from the Voronoi Diagram before the Distance Transform. The second possibility, used in this paper, is to dilate the objects and the walls in a safely distance to ensure that the robot does not collide nor accepts passages narrower than the robot size. The last step is to calculate the trajectory in the image generated by the Voronoi Diagram using the Fast Marching Method, the path obtained verifies the smoothness and

safety required for mobile manipulator's path planning. The advantages of this method include the ease of implementation, the speed of the method, and the quality of the trajectories produced. The method works in 2D and 3D, and it can be used on a global or local scale, in this case operating with sensor information instead of using a priori map (sensor based planning).
The chapter is organized as follow, after discussing related work in Section 2, Section 3 explains the requirements and basic ideas of the method and in Sections 4, 5 and 6 the theoretical background of various ideas used in the method are discussed. Section 7 discusses some results of the method "Voroni Diagram and Extended Fast Marching" for an holonomic mobile robot. Sections 8 discuss the contributions of the method with respect to other methods and Section 9 concludes the chapter.

2. Related work

The objective of our work is to calculate collision-free trajectories for a mobile robot and manipulators operating in environments with unknown obstacles (dynamic or not). The technique proposed here avoids the classical partition in global based on a priori information (motion planning or global planning), and local based on sensory information (reactive navigation, collision avoidance or sensor based local planning).
From a theoretical point of view, the motion planning problem is well understood and formulated, and there is a set of classical solutions able to compute a geometrical trajectory avoiding all known obstacles. Mobile robot path planning approaches can be divided into five classes according to Latombe (Latombe, 1991). Roadmap methods extract a network representation of the environment and then apply graph search algorithms to find a path. Exact cell decomposition methods construct non-overlapping regions that cover free space and encode cell connectivity in a graph. Approximate cell decomposition is similar, but cells are of pre-defined shape (e.g. rectangles) and not exactly cover free space. Potential field methods differ from the other four in such a way that they consider the robot as a point evolving under the influence of forces that attract it to the goal while pushing it away from obstacles. Navigation functions are commonly considered as special case of potential fields. Most of these general methods are not applicable if the environment is dynamic or there are no modelled obstacles. Hence, to avoid the problem of executing an unrealistic geometrical trajectory which can collide with obstacles, obstacle avoidance algorithms have been developed to provide a robust way of coping with this problem.
These methods are based on a perception-action process that is repeated periodically at a high rate. The process has two steps: first, the sensory information is collected and then a motion command is calculated to avoid collisions while moving the robot toward a given goal location (that is while maintaining the global plan previously determined)
Due to fast operation rate of the collision avoidance algorithms they can deal robustly with unknown obstacles and dynamic scenarios. However, it is difficult to obtain optimal solutions and to avoid trap situations since they use only local sensory information. This classical approach to motion planning is demonstrated schematically in figure 1. Next, we describe some of these related methods.
One of the classical methods for dynamically coping with the collision avoidance problem is the potential field approach developed by Khatib (Khatib, 1986 and Khatib & Chatila, 1995). This approach is based on the creation of an artificial potential field in which the target is an attractive pole and the obstacles are repulsive surfaces. The robot follows the gradient of

this potential toward its minimum. The derived force induces a collinear and proportional acceleration enabling easy dynamics and kinematic control actions. This technique can be used at a global or local level depending on the information used to generate the potentials. The major advantage of this method is its simplicity, and the capability of being used dynamically due to easy treatment of fixed and mobile obstacles. Its major disadvantage is the possible existence of local minima and the oscillations for certain configurations of obstacles. Despite of such problem this technique has been used extensively in reactive architectures because of its ability to encapsulate specific robot behaviours.

A variation of this idea is the vector field histogram (VFH) method (Borenstein & Koren, 1991). VHF uses a method consisting of a local heuristic choice based on a bi-dimensional grid modelling of the environment. A polar histogram is constructed around the robot representing the polar densities of the obstacles. The sector associated with the least density closest to the goal direction is chosen. This technique is simple, but may lead to oscillations and can also be trapped in certain configurations. Posteriorly, several improvements have been proposed in the VFH+ to improve the security distance (object enlarging), to reduce oscillations between valleys (hysteresis), and to obtain path smoothness (Ulrich & Borenstein, 1998). A third version, VHF* has been proposed that includes a "look-ahead" verification of the robots motions in order to avoid local traps (Ulrich & Borenstein, 2000). A projected position tree is built by predicting each possible movement of the robot and searches using A* classic method to choose a motor command.

Between the obstacle avoidance (sensor-based local planners) we can find the Curvature-Velocity method (Simmons, 1996), which treats obstacle avoidance as a constrained optimization in velocity space. Vehicle dynamics and obstacle information are converted into constrains and used in the optimization process.

The Dynamic Window approach (Fox et al., 1997) uses a similar approach to the Curvature-Velocity method. It also uses a constrained search in the velocity space to determine convenient speed commands. The use of a grid based representation on one hand is simple but on the other hand increases the memory requirements and the computational cost. The Nearness Diagram approach (Minguez & Montano, 2004) is a situated-activity reasoning system that based on the sensor information identifies the situation and determines the corresponding action. It uses a reactive navigation method designed by a symbolic level decision tree based on a set of complete and exclusive definitions of the possible situations.

The Elastic Bands proposed in (Quinlan & Khatib, 1993) represent connected bubbles of the free space subject to repulsive forces from obstacles and attractive forces from neighbouring bubbles. The elastic band iteratively smoothes the plan and adapts it to moving or unexpected obstacles.

The majority of the above methods have commutation difficulties, due to the fact that local avoidance algorithms provide modifications of global trajectories, but do not offer solutions when the global trajectory is blocked or trapped in local minima. The classical solution to this problem is to include a supervisor to analyze the global path execution. In the case where a local trap or a blocked trajectory is detected, it begins a search for a new global path from the current location to the goal point.

The objective of the present work is to unify the global motion planner and the obstacle avoidance planner in a single planner. This provides a smooth and reliable global motion path that avoids local obstacles from the current position to the goal destination. The motion planning structure with the proposed planner is shown in figure. 2.

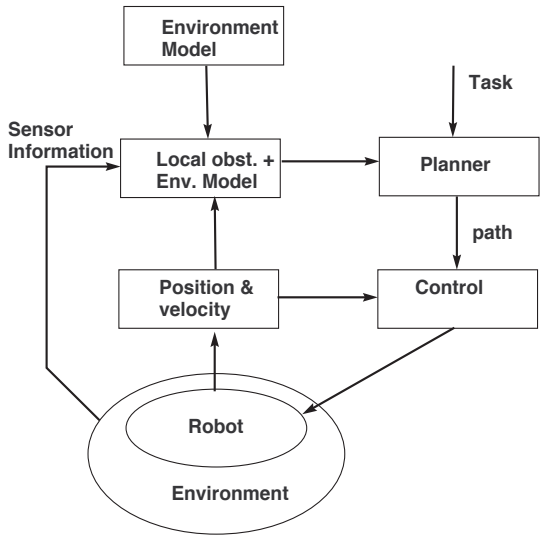


Fig. 2. Sensor-based global planning approach.

3. The sensor-based global planner algorithm

The sensor-based global planner algorithm proposed in this study is based on three key points:

- *Fast response*. The planner needs to be fast enough to be used reactively in case unexpected obstacles make it necessary to plan a new calculation. To obtain such a fast response, a fast planning algorithm and a fast-simple treatment or modelling of the sensor information is required. This requires a low complexity order algorithm to obtain a real-time response to unexpected situations.
- *Smooth trajectory*. The planner must be able to provide a smooth motion plan which can ~~to~~ be executed by the robot motion controller. This avoids the need for a local refinement of the trajectory.
- *Reliable trajectory*. The planner will provide a safe (reasonably far from a priori and detected obstacles) and reliable trajectory (free from local traps). This avoids the coordination problem between the local avoidance planners and the global planners when local traps or blocked trajectories exist in the environment.

To satisfy all these requirements the sensor based global planner approach follows these steps:

1. The first step of the algorithm integrates sensor measurements into a grid based map by updating the corresponding occupied cells. The sensory information is included in the environment map avoiding complex modelling or information fusion.
2. In the second step the objects are enlarged in the radius of the mobile robot, which will eliminate unfeasible paths~~,~~ and avoid additional path verifications.

3. The updated environment model is then converted to an environment safety image by using the Euclidean distance transform in the environment image. In the new image, each position contains the distance to the closest object that produces a distance map based on the Euclidean distance transform where maxima are obtained in the Voronoi diagram of the environment model (skeleton). This reveals the collision risk of a given trajectory in terms of the distance to obstacles at each point of the trajectory. The step starts with the calculation of the Voronoi Diagram of a priori map of the environment (which are the cells located equidistant to the obstacles). This process is done by means of morphological operations on the image of the environment map. To be more precise, a skeletonisation is done by using image techniques in order to obtain the Voronoi Diagram. Simultaneously, the method calculates the Extended Voronoi Transform of the map to obtain the Euclidean distances of each cell to the closest obstacle in the environment. This part of the method is also very because it is done by efficient image processing techniques.
4. Based on this distance map to obstacles, the Level Set Method (Fast Marching), is used to calculate the shortest trajectories in the potential surface defined by the Extended Voronoi Transform of the map. The calculated trajectory is geodesic in the potential surface, i.e. with a viscous distance. This viscosity is indicated by the grey level. If the Level Set Method were used directly on the environment map, we would obtain the shortest geometrical trajectory, but then the trajectory would be neither safe nor smooth.

The main reason to separate the trajectory calculation into two parts (in classical approaches) is because the global path calculation is very slow and it is impossible to re-calculate it continuously as the robot is moving. The method proposed here is extremely fast, where the whole process (sensing-model-plan) takes only 0.15 seconds (for a medium length trajectory), letting the algorithm to re-calculate a path from the current robots position to the goal location reactively.

4. Voronoi Diagram and Skeleton determination

It has been observed that the skeleton is embedded in the Voronoi diagram of a polygonal shape (Lee, 1982). Similarly, the skeleton of a shape described by a discrete sequence of boundary points can be approximated from the Voronoi diagram of the points. Both approaches yield a connected, Euclidean skeleton, but the latter is perhaps more appropriate for images since point sequences are more easily obtained than polygons. Although it is not true in general, if one restricts the shapes to those which are morphologically open and closed with respect to a finite-sized disk, the resulting skeleton approximated from the Voronoi diagram of a finite sampling of the boundary is close to the actual skeleton. In this case, the approximation error can be quantified, and made arbitrarily closer to zero.

Consider the set F, closed in R^2. A Voronoi region is associated with each point in F.

$$V_F(p) = \{q : d(q,p) \leq d(q, F/\{p\})\} \tag{1}$$

The Voronoi diagram of F is the union of the boundaries of all the Voronoi regions.

$$VD(F) = \bigcup_{p \in F} \partial V_F(p). \tag{2}$$

A maximal disk in G is one which is contained in G and not contained in any other disk in G. Assume that all maximal disks in G are bounded. The skeleton $\sigma(G)$ is the set of centres of maximal disks in G. One desires the skeleton to be "graph-like" retraction of the original set. In general, this cannot be assured due to the presence of infinitesimal detail. However, it is possible to eliminate these fine structures by assuming a reasonable subclass, the regular sets. A compact set, K is said to be r – *regular* if it is morphologically open and closed with respect to a disk of radius $r > 0$ (Serra, 1982). It is possible to show that ∂K is a disjoint union of closed simple C^2 curves with curvature magnitude no greater than $1/r$. The skeleton of the interior of K is well-behaved and graph-like.

4.1. Skeleton-based generalization algorithm

One issue that needs improvement is the existence of spurious "hairs" on the generated skeletons. This is a well-known artefact of skeleton generation, where any irregularities in the boundary generates unwanted skeleton branches. Ogniewicz (Ogniewicz & Kubler, 1995) attempted to reduce skeletons formed from raster boundary points to a simple form by pruning the leaf nodes of the skeleton until a specified minimum circum-circle was achieved. However, with the development of the one-step crust and skeleton algorithm this process may be greatly simplified. Blum (Blum, 1967), as well as Alt (Alt & Schwarzkopf, 1995) showed that leaf nodes of a skeleton correspond to locations of minimum curvature on the boundary. For a sampled boundary curve this means that three adjacent sample points are co-circular, with their centre at the skeleton leaf. If we wish to simplify the skeleton we should retract leaf nodes to their parent node location, creating four co-circular points instead of three.
The retraction is performed by taking the central point of the three, defining the leaf node, and moving it towards the parent node of the skeleton until it meets the parent node circum-circle, which smoothes outward-pointing salient in the boundary of the object. The same process is repeated from the other side of the boundary, which may displace some of the points involved in the first smoothing step, but as the process is convergent a small number of iterations suffice to produce a smoothed curve having the same number of points as the original, but with a simplified skeleton. Figure 3 shows the Voronoi diagram obtained by skeletonization of a room.

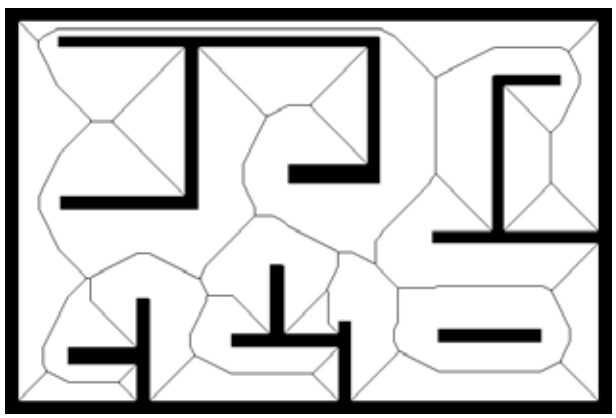


Fig. 3. Map of the room with the Voronoi made by skeletonization.

5. The Extended Voronoi Transform

A k-dimensional binary image is a function I from the elements of an $n_1 \times ... \times n_k$ array to [0,1]. The elements are called pixels when $n = 2$ and voxels when $n \geq 3$. Voxels of value are called background and foreground or feature voxels, respectively. For a given distance metric, the Extended Voronoi Transform (also called Image Distance Transform) of an image I is an assignment to each voxel x of the distance between x and the closest feature voxel in I, i.e. it consists of associate grey levels for each cell. As a result of this process, a kind of potential proportional to the distance to the nearest obstacles for each cell is obtained. Zero potential indicates that a given cell is part of an obstacle and maxima potential cells correspond to cells located in the Voronoi diagrams (which are the cells located equidistant to the obstacles). This function introduces a potential similar to a repulsive electric potential.

At each dimension level, the transform is computed by constructing the intersection of the Voronoi diagram whose sites are the feature voxels with each row of the image. This construct is performed efficiently by using the transform in the next lower dimension.

The algorithm calculates the Voronoi and the Extended Voronoi Transform simultaneously, therefore, saving time. The algorithm used (Maurer *et al.*, 2003) for a k-dimensional binary image is linear in the total number of voxels N, i.e. it has a $O(N)$ computational complexity. A parallel version of the algorithm runs in $O(N/p)$ time with p processors. Figure 4, shows the potential image generated by the Extended Voronoi Transform for the previous room example.

Fig. 4. Potential of the Extended Voronoi Transform.

6. Level Set Method and the Fast Marching Planning Method

The level set method was devised by Osher and Sethian (Osher & Sethian, 1988) as a simple and versatile method for computing and analyzing the motion of the interface in two or three dimensions. The goal is to compute and analyze subsequent motion of the interface under a velocity field. This velocity can depend on position, time, and the geometry of the interface and on external physics. The interface is captured for later time, as the zero level set of a smooth (at least Lipschitz continuous) function. Topological merging and breaking are well defined and easily performed.

The original level set idea of Osher and Sethian involves tracking the evolution of an initial front γ_0 as it propagates in a direction normal to itself with a given speed, function V. The main idea is to match the one-parameter family of fronts $\{\gamma_t\}_{t\geq 0}$, where γ_t is the position of the front at time t, with a one-parameter family of moving surfaces in such a way that the zero level set of the surface always yields the moving front. To determine the front propagation, a partial differential equation must be solved for the motion of the evolving surface. To be more precise, let γ_0 be an initial front in R^d, d $\geq$ 2 and assume that the so-called *level set function* $\phi : \mathbb{R}^d \times \mathbb{R}_+ \to \mathbb{R}$ is such that at time t $\geq$ 0 the zero level set of ϕ is the front γ_0. We further assume that $\phi(x;0) = \pm d(x)$; where $d(x)$ is the distance from x to the curve γ_0. We use a plus sign if x is inside γ_0 and minus if it is outside. Let each level set of ϕ along its gradient field with speed V. This speed function should match the desired speed function for the zero level set of ϕ. Now consider the motion of, e.g., the level set

$$\left\{x \in \mathbb{R}^d : \phi(x;t) = 0\right\}. \tag{3}$$

Let $x(t)$ be trajectory of a particle located at this level set so that

$$\phi(x(t);t) = 0. \tag{4}$$

The particle speed $dxdt$ in the direction n normal to the level set is given by the speed function V, and hence

$$\frac{dx}{dt} \cdot n = V. \tag{5}$$

Where the normal vector n is given by

$$n = -\frac{\nabla\phi}{|\nabla\phi|}. \tag{6}$$

This is a vector pointing outwards, giving our initialization of u. By the chain rule

$$\frac{\partial\phi}{\partial t} + \frac{dx}{dt} \cdot \nabla\phi = 0. \tag{7}$$

Therefore $\phi(x;t)$ satisfies the partial differential equation (the level set equation)

$$\frac{\partial\phi}{\partial t} - V|\nabla\phi| = 0, \tag{8}$$

and the initial condition

$$\phi(x;t=0) = \pm d(x). \tag{9}$$

This is called an Eulerian formulation of the front propagation problem because it is written in terms of a fixed coordinate system in the physical domain.

If the speed function V is either always positive or always negative, we can introduce a new variable (the arrival time function) $T(x)$ defined by

$$\phi(x, T(x)) = 0. \tag{10}$$

In other words, $T(x)$ is the time when $\phi(x;t) = 0$. If $\frac{d\phi}{dt} \neq 0$, then T will satisfy the stationary Eikonal equation

$$V|\nabla T| = 1, \tag{11}$$

coupled with the boundary condition

$$T_{\partial(\Omega)=0} = 0. \tag{12}$$

The advantage of formula (11) is that we can solve it numerically by the fast marching method (Sethian, 1996c), which is exactly what will be done in this study.

The direct use of the Fast Marching method does not guarantee a smooth and safe trajectory. Due to the way that the front wave is propagated and the shortest path is determined, the trajectory is not safe because it selects the shortest path, resulting in un-safe trajectory that touches the corners and walls, as is shown in figure 5. This problem can easily be reduced by enlarging the obstacles, but even in this case the trajectory tends to go close to the walls.

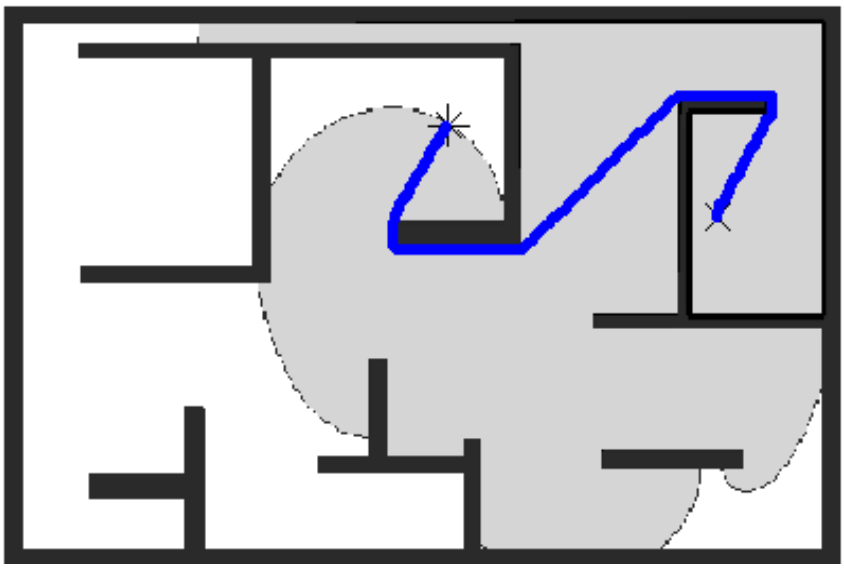


Fig. 5. Trajectory calculated by Fast Marching Method directly, without the use of the Extended Voronoi Transform.

The use of the Extended Voronoi transform together with the Fast Marching method improves the quality of the calculated trajectory considerably. On the one hand, the trajectories tend to go close to the Voronoi skeleton because of the optimal conditions of this area for robot movement; on the other hand the trajectories are also considerably

smoothened. This can be observed in figure 6, where safe and smooth quality of the trajectory can be noted.

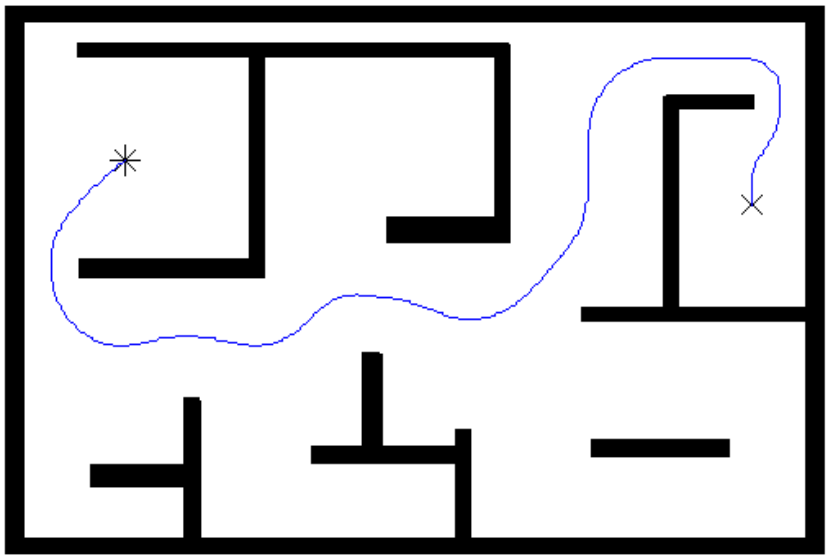


Fig. 6. Trajectory calculated using Fast Marching with the Extended Voronoi Transform.

Summing up, the central mathematical idea is to view the moving front γ_t as the zero level set of the higher-dimensional level set function $\phi(x;t)$. Depending on the form of the speed function V, the propagation of the level set function $\phi(x;t)$ is described by the initial problem for a nonlinear Hamilton-Jacobi type partial differential equation (7) of first or second order.

If $V>0$ or $V<0$, it is also possible to formulate the problem in terms of a time function $T(x)$ which solves a boundary value problem for a stationary Eikonal equation (11).

Fast Marching Methods are designed for problems in which the speed function never changes sign, so that the front is always moving forward or backward. This allows us to convert the problem to a stationary formulation, because the front crosses each grid point only once. This conversion to a stationary formulation, plus a whole set of numerical tricks, gives it tremendous speed.

Because of the nonlinear nature of the governing partial differential equation (7) or (11), solutions are not smooth enough to satisfy this equation in the classical sense (the level set function and the time function are typically only Lipschitz). Furthermore, generalized solutions, i.e., Lipschitz continuous functions satisfying the equations almost everywhere, are not uniquely determined by their data and additional selection criteria (entropy conditions) are required to select the (physically) correct generalized

solutions. The correct mathematical framework to treat Hamilton-Jacobi type equations is provided by the notion of viscosity solutions (Crandall & Lions, 1983; Crandall et al., 1992).

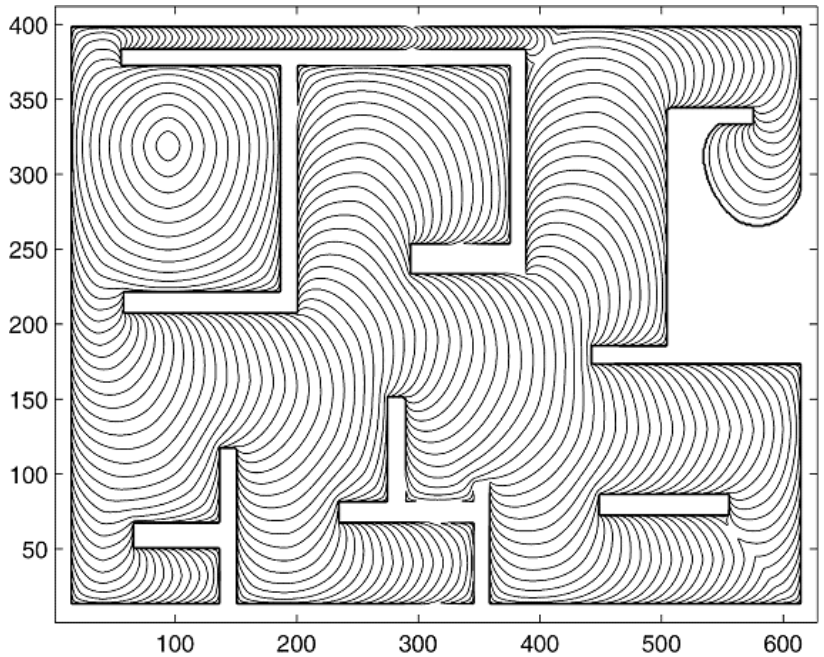


Fig. 7. Front propagation of the Fast Marching Method with the Extended Voronoi Transform.

After its introduction, the level set approach has been successfully applied to a wide array of problems that arise in geometry, mechanics, computer vision, and manufacturing processes, see (Sethian, 1996b). Numerous advances have been made in the original technique, including the adaptive narrow band methodology (Adalsteinsson & Seth, 1995) and the fast marching method for solving the static Eikonal equation ((Sethian, 1996a), (Sethian, 1996b)). For further details and summaries of level set and fast marching techniques for numerical purposes, see (Sethian, 1996b). The Fast Marching Method is an $O(n\log(n))$ algorithm.

7. Results

To illustrate the capability of the proposed method three tests are shown. In the first test, the method proposed is applied directly to the data obtained from a laser scan around the robot. Note how the method obtains a good trade off between trajectory distance, distances to obstacles and smooth changes in the trajectory (figures 8 and 9).

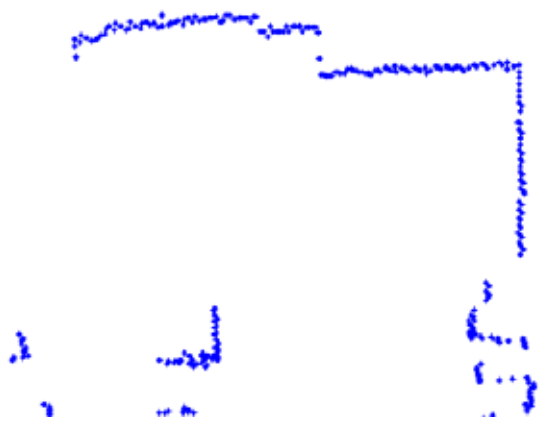


Fig. 8. Laser scan data.

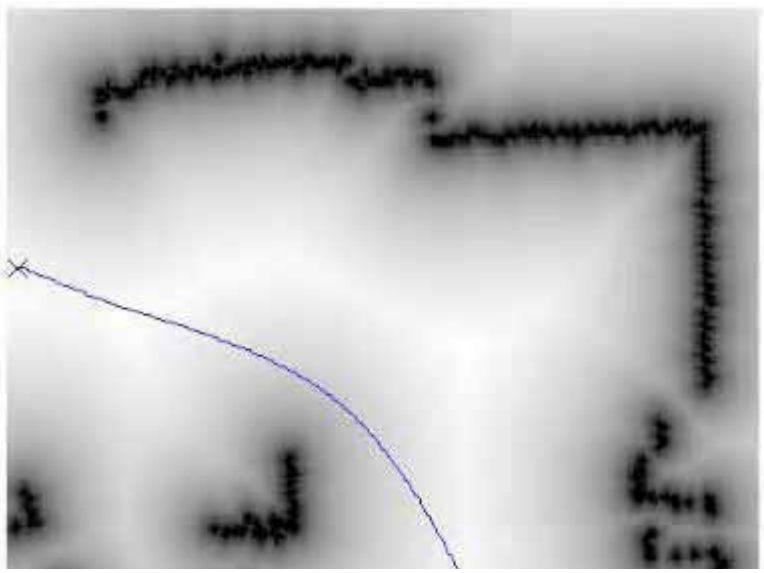


Fig. 9. Extended Voronoi transform of the scanned data and trajectory obtained with the Fast Marching Method

In the second test, in order to show global plan capabilities, the method is applied to the whole plant of the building where our laboratory is located. The laboratory plant is around 2000 square meters (medium size). The results are shown in figures 10 and 11.

Fig. 10. Extended Voronoi transform of the UC3M Robotics Lab plan

Fig. 11. Motion trajectory obtained with the Extended Fast Marching Method

The last test shows the combination of the global and local properties of this method. In this case a simple trajectory motion is determined from an initial position to the goal position. During the robot motion the robot observes the environment with its laser scan, introduces it in the map, and plans a new trajectory. Local observations (obstacles located in the middle of the corridor) that originate may slightly modify the trajectories to avoid detected obstacles (figure 12). In the last image of figure 12 the obstacles detected blocked the corridor and the sensor based global planner finds a completely different trajectory.
This technique shows the advantage by not only being local, but also global, and combines sensor based local planning capabilities with global planning capabilities to react to the obstacles rapidly, while maintaining reliability in the planned trajectory. The method always finds a solution, if it exists.

8. Discussion

We present here a discussion regarding alternative planning approaches, the limitations of the proposed method, and the potential effect on mobile robot's control architecture.

8.1. Comparison with existing methods

The common limitation of all the reactive navigation methods analyzed in this section is that they cannot guarantee global convergence to the goal location, because they use only a fraction of the information available (the local sensory information). Some researchers have worked on introducing global information into the reactive collision avoidance methods to avoid local traps situations. This approach has been adopted in (Ulrich & Borenstein, 2000) which utilizes a look-ahead verification in order to analyze the consequences of a given motion a few steps ahead avoiding trap situations. Other authors exploit the information about global environment connectivity to avoid trap situations (Minguez & Montano, 2001). Such solutions still maintain the classical two-level approach, and require additional complexity at the obstacle avoidance level to improve the reliability.

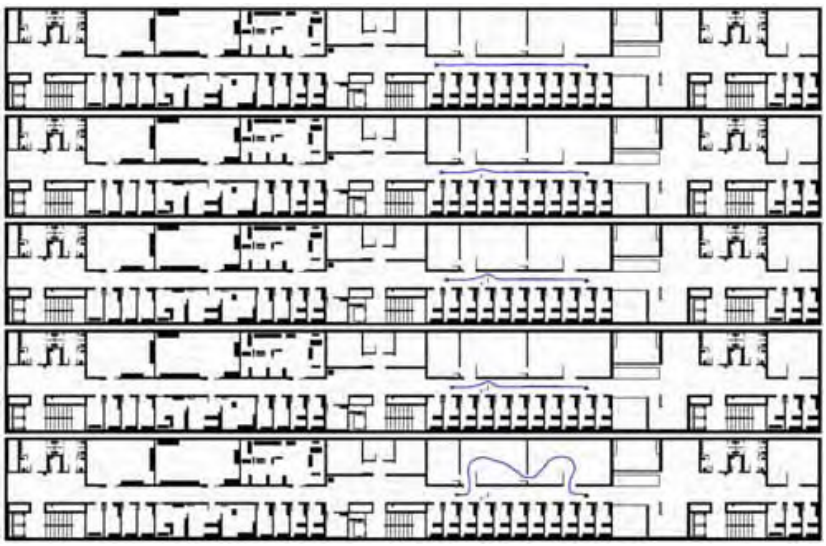


Fig. 12. Dynamical evolution of the path when the robot reads information about the new obstacles that are not in the previous map and the robot cannot pass through the corridor.

The Voronoi Fast Marching method is consistent at local and global scale, because it guarantees a motion path (if it exists), and does not require a global re-planning supervision to re-start the planning when a local trap is detected or a path is blocked. Additionally, the path calculated has good safety and smoothness characteristics.

8.2. Method limitations

The main limitation of the Voronoi Fast Marching method is its computational complexity and the computational differences between the initial and the final stages of task implementation. Because the fast marching method is a wave propagation technique, the area explored by the algorithm can be quite different. This causes the algorithm to be faster at the end of the task. Despite of its computational efficiency for medium size environments, the computational cost increases with the environment size. In our test site, the laboratory plant is about 2000 square meters (medium size) and the slowest computation time detected was 150 milliseconds (2.2 Ghz Pentium 4 processor). This time is acceptable for most applications, but because the algorithm has to operate in a reactive way in case of bigger environments, the computational cost has to be controlled. One possible solution is to increase the cell size, another is to use a multi-resolution map system which uses a big cell size to calculate the motion path in areas outside of a window located around the robot.

The method can be extended to non-holonomic robot structures by modifying the algorithm to operate in the robot's configuration space instead of the environment map. However, the computational cost increases and its capability to operate in real time decreases.

8.3. Architectural effect

Due to the fact that the VFM method integrates global motion planning and obstacle avoidance capabilities in an algorithm, which lets us simplify the mobile robot control architecture. This technique has been originally designed for mobile manipulators, which requires a smoother and safer motion plan compared to mobile robots. From an architectural point of view, this uncovers an old discussion in the mobile robot research community. Can a planned architecture be as efficient and as responsive as a reactive architecture? This difficult question we think may be answered by utilizing sensor-based global planning method that can lead us to develop efficient and responsive sensor-based planned architectures, and to also simplify the current hybrid architectures which operate deliberatively at upper architectural levels and reactively at lower levels.

9. Conclusions

We have addressed here a sensor-based global planner. We have presented a method able to deal simultaneously with global and local planning requirements using a combination of the extended Voronoi transform and the Fast Marching method. The advantage is that our design is able to provide a smooth and safe motion trajectory, and at the same time guarantee that the trajectory obtained from the current location to the goal location is free from local traps (with the information available at that moment).

To illustrate its possibilities, the method has been used for planning a trajectory in different situations: In the first case operating only with sensor information (in this case only as a local planner); in the second case working as a global planner by using a priori map only; in the third situation operating as a sensor-based global planner using a priori information and sensor information, to re-plan dynamically the trajectory from the current position to the goal position as soon as the sensor originates information changes around the robot. The dimensions of the laboratory environment are 116x14 meters (the cell resolution used in the a priori information map are 12 cm). For this environment the first step (the Extended Voronoi Transform) takes 0.05 seconds in a Pentium 4 at 2.2 Ghz, and the second step (Fast Marching) takes between 0.05 and 0.15 seconds, depending on the length of the trajectory. For a medium distance trajectory on the map, the value is around 0.10 seconds.

The proposed method is highly efficient from a computational point of view because it operates directly over a 2D image map (without extracting adjacency maps), and due to the fact that Fast Marching complexity is $O(n\log(n))$ and the Extended Voronoi Transform complexity is $O((n))$, where n is the number of cells in the environment map. The main contribution of the method is that it robustly achieves smooth and safe motion plans during dynamic scenarios in real time that can be used at low control levels without any additional smoothing interpolations. This permits the method to fuse collision avoidance and global planning in only one module, which can simplify the control architecture of the mobile robots and manipulators.

10. Acknowledgment

The authors would like to thank the Spanish Government for the funds provided by the MICYT project DPI2004-00594.

11. References

Adalsteinsson, D. &. Sethian, J.A. (1995). A fast level set method for propagating interfaces, *J. Computational Physics*, Vol. 11, Nº 2, pp. 269–277, 1995. ISSN: 0021-9991

Alami, R. ; Chatila, R.; Fleury, S. ; Ghallab M. & Ingrand, F. (1998). An Architecture for Autonomy, *Int. Journal of Robotics Research,* Vol. 17(4), pp. 315–337, 1998, pp. 315–337, ISSN: 0278-3649

Alt, H. & Schwarzkopf, O. (1995). The Voronoi diagram of curved objects, *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pp. 89–97, 1995

Arkin, R. C. (1998). *Behaviour-Based Robotics*, MIT Press, Cambridge, MA

Arkin, R. C. (1990). Integrating Behavioral, Perceptual and World Knowledge in Reactive Navigation, *Robotics and Autonomous Systems*, vol. 6, Nº12, June 1990, pp. 105–122, ISSN: 0921-8890.

Blum, H. (1967). A transformation for extracting new descriptors of shape, *Models for Perception of Speech and Visual Form*, MIT Press, pp. 153–171, 1967.

Borenstein, J. & Koren, Y. (1991). The vector field histogram - fast obstacle avoidance for mobile robots, *IEEE Trans. on Robotics and Automation*, Vol. 3, Nº 7, June 1991, pp. 78–288, ISSN: 1042-296X

Bonasso, R.P.; Kortenkamp, D. ; Miller, D.P. & Slack, M. (1996). Experiences with Architecture for Intelligent, Reactive Agents, *Proc. IJCAI'95 Workshop (ATAL), Lecture Notes in Computer Science 1037*, pp. 187–202, ISBN: 3 540 60805 2

Brooks, R. (1986). A Robust Layered control System for a Mobile Robot, *IEEE Trans. on Robotics and Automation*, Vol. 2(1), pp. 14–23, ISSN: 1042-296X

Chatila, R. (1995). Deliberation and reactivity in autonomous mobile robot, *Journal of Robotics and Autonomous Systems*, Vol. 16, Nº2-4, Dec. 1995, pp. 197–211, ISSN: 0921-8890

Crandall, M.G. & Lions, P.L. (1983). Viscosity solutions of Hamilton Jacobi equations, *Trans. Amer. Math. Soc.*, Vol. 277, pp. 1–42, 1983

Crandall, M.G.; Ishii, H. & Lions, P.L. (1992). User's guide to viscosity solutions of second order partial differential equations, *Bull. Amer. Math. Soc.*, Vol. 27, Nº. 1, pp. 1–67, 1992

Fox, D.; Burgard, W. & Thrun, S. (1997). The dynamic window approach to collision avoidance, *IEEE Robotics & Automation Magazine*, Vol. 4, Mar. 1997, pp. 23–33, ISSN: 1070-9932

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Robot. Res.*, vol. 5, spring 1986, pp. 90–98, ISSN: 0278-3649

Khatib, M & Chatila, R. (1995). An extended potential field approach for mobile robot sensor-based motions, *Proceedings of the Int. Conf. on Intelligent Autonomous Systems.* pp. 490-496, Karlsruhe Germany, 27-30 March 1995, IOS Press, Amsterdam

Latombe, J.C. (1991). Robot motion planning, *Kluwer Academic Publishers*, 1991

Lee, D.T. (1982). Medial axis transformation of a planar shape, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 4, Nº 4, Jul 1982, pp. 363–369, ISSN: 0162-8828

Low, K. H. ; K. Leow, W. & Ang, M. H. (2002). A Hybrid Mobile Robot Architecture with Integrated Planning and Control, *Proceedings of the 1st International Joint Conference on: Autonomous Agents and Multiagent Systems*, Nº 2, pp 219-226, Bologna-Italy, July 2002, ACM press NY

Maurer, C. R., Qi, R. & Raghavan, V. (2003). A Linear Time Algorithm for Computing Exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25 , Nº 2, Feb. 2003, pp. 265 – 270, ISSN:0162-8828

Meystel, A. (1986). Planning in a Hierarchical Nested Autonomous Control System, *Mobile Robots, SPIE Proc.*, Vol. 727, pp. 42–76

Minguez, J. & Montano, L. (2001). Global Nearness Diagram navigation, *Proc. IEEE Int. Conf. on Robotics and Automation,* Vol. 1, pp.33–39, ISSN: 1050-4729, Seoul- Korea, May 2001, IEEE NJ

Minguez, J. & Montano, L. (2004). Nearness Diagram navigation: collision avoidance in troublesome scenarios, *IEEE Trans. Robot. and Automat.*, Vol. 20, Feb. 2004, pp. 45–59, ISSN: 1042-296X

Ogniewicz, R.L. & Kubler, O. (1995). Hierarchic Voronoi skeletons, *Pattern Recognition*, Vol. 28, Nº 3, Mar 1995, p 343-359, ISSN: 0031-3203

Osher, S. & Sethian, J.A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, *Journal of Computational Physics*, Vol. 79, pp. 12–49, 1988.

Quinlan, S. & Khatib, O. (1993). Elastic bands: connecting path planning and control, *Proc. IEEE Int. Conf. on Robotics and Automation,* Vol. 2, pp. 802-807, ISBN: 0-8186-3452-9, Atlanta-USA, May 1993, IEEE NJ

Serra, J. (1982). Image Analysis and Mathematical Morphology, *Academic Press*, 1982

Sethian, J.A. (1996a). Level set methods, *Cambridge University Press*, 1996.

Sethian, J.A. (1996b). Theory, algorithms, and applications of level set methods for propagating interfaces, *Acta numerica*, Cambridge Univ. Press, pp. 309–395, 1996.

Sethian, J.A.(1996c). A fast marching level set method for monotonically advancing fronts, *Proc. Nat. Acad Sci. U.S.A.*, Vol. 93, Nº 4, pp. 1591–1595, 1996.

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance, *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 4 pp. 3375–3382, Minneapolis, USA,. Apr. 1996

Ulrich, I. & Borenstein, J. (1998). VFH+: reliable obstacle avoidance for fast mobile robots, *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 2, pp. 1572–1577, Leuven, Belgium, May 1998.

Ulrich, I. & Borenstein J. (2000). VFH*: local obstacle avoidance with look-ahead verification," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2000. Vol. 3, pp 2505-2511, San Francisco, USA, Apr. 2000.

23

Effective Method for Autonomous Simultaneous Localization and Map Building in Unknown Indoor Environments

Y.L. Ip, A.B. Rad, and Y.K. Wong
Department of Electrical Engineering
The Hong Kong Polytechnic University
Hung Hom, Kowloon,
Hong Kong

1. Introduction

Many published papers [17, 44] on the map building of Autonomous Mobile Robots (AMRs) do not consider the question of autonomous exploration at all. This is, of course, often just a choice of research focus; effort is expended on the mechanics of map construction from sensor data without worrying about how the sensing positions were selected. Or the map is provided by the operator [7, 11] for any other applications. In our view, the autonomous exploration skill is an extremely important capability for a truly AMR. For example: as it is desired to build a map of unknown environments without human intervention, AMRs should be equipped with a skill of autonomous exploration which includes the competence of path finding, obstacle avoidance and monitor progress towards reaching a goal location or target.
Several possible strategies for exploration of unknown environment are described in the robotics literature. The following categorization is taken from Lee [24]:

1. Human Control – mobile motion is controlled by human operator.
2. Reactive Control – the mobile robot movement is relied on the perception system.
3. Approaching the unknown – the mobile robot move into the region that it knows least in the environment.
4. Optimal search strategies – the approach is focused on to search the shortest path for seeking the goal.

In the first category, the robot is guided around the environment by a human operator. This requires human intervention in the map building process. Therefore, it is not suitable for an autonomous exploration mobile robot.
For reactive exploration approach (2nd category), the sensory data (perception space) is used to calculate or determine the control actions (action space). The sensory data may be the distance information from infrared, sonar or laser range finder type sensors, visual information or processed information obtained after appropriate fusion of multiple sensor outputs. The control actions are usually a change in steering angle and setting a translation velocity of the robot that will avoid collisions with the obstacles on its way and reach the desired target. Pre-designed or adaptive systems based on fuzzy logic [15, 26, 28, 31, 40-41, 45-46, 48-49], neural-networks [9, 33-35, 38, 51] or combination of them [27] are designed by this reactive navigation

approach. Since a totally reactive behavior uses only locally available environment information, without any memory of the previously encountered situations, the autonomous robots are found to suffer from local minima situations. For that reason, a reactive approach cannot be applied to autonomous exploration robot independently.

In the strategy of approaching the unknown (3rd category), a mobile robot tries to move towards the regions of its environment about which it knows least. A mobile robot uses the perception sensor information to search the new territory and move towards that area. This process is repeated until the whole environment has been covered. Global grid model is used in some of map building system to represent the environment. Thrun [44] developed a system trained by an artificial neural network to translate neighboring groups of sonar readings onto occupancy values in the grid and then control the mobile robot to explore directed towards to an areas of high uncertainty in global grid map. However, this system required offline training by the robot simulator though neural network and the algorithm depended on the assumption that environment was rectilinear. In terms of topological maps approach for exploration, Edlinger and Weiss [12] developed a robot equipped with laser range finder to detect obstacle-free segments from the scans and it created topological relations between those scans. A similar approach proposed by Yamauchi [50] required an accurate laser range finder sensing to detect the open space. Recently, Duckett [10] proposed an exploration system to build a topological map which is augmented with metric information concerning the distance and angles between connected places. A trained neural network was used to detect an open space in the environment via sonar sensors and infrared sensor. The open space areas were added to a stack of unexplored locations which were visited in turn until the whole environment had been covered by the robot. This system was tested successfully in a middle-scale indoor environment with Nomad 200 mobile robot [36]. However, this approach relies on a set of sonar sensors and infrared sensors mounted on the rotating robot's turret (which can be rotated independently relative to the base of the robot). It needs to stop the robot on every 1m place in environment to scan and search the possible areas of uncharted territory in all directions. This means that the system would work with Nomad 200 mobile robot only and not suitable for the mobile robot which without rotating turret.

For the fourth category (Optimal search strategies), many researchers have provided mathematical analyses of strategies which are minimized the length of the path traveled by the robot during exploration. It is similar to the well-known traveling salesman problem [14].

In the last two decades, many researchers proposed robust and successful reactive navigation controller, such as behavior-based method [32, 40, 45, 49] and model-based method [2, 21-22, 30]. However, while reactive navigation approaches are often very robust, they cannot be guarantee to navigate all areas in an unknown environment. Therefore, the approaches based on reactive control (2nd category) and approaching the unknown (3rd category) would seem the most promising for autonomous exploration via mobile robot. Therefore, a novel mixing approach combined the reactive approach with approaching the unknown strategy will be presented in this paper.

Our approach is closest in spirit to that of Edlinger and Weiss [12] and Duckett [10], though it does not require an accurate laser range finder for perception system, a rotating robot's turret and a set of training for setting the open space detection neural network system. A simple and real-time system is designed for detecting an open free space via a Bayesian update theory [1] instead of pre-trained system or accurate sensing system. Reactive navigation scheme is applied to start the exploration by using a predefined Hierarchical Fused Fuzzy System (HFFS) [4, 20, 23]. Those proposed algorithm would generate a metric topological map model after the exploration.

In studying the problem of feature based SLAM, a number of specific problems should be considered. These include feature extraction, data association, map management and computation complexity.

In recent times, a number of research groups have attempted to implement real-time SLAM approach successfully with SICK laser scanner [3, 8, 16, 29] in indoor environments. The main advantage of the SICK laser scanner is that the sensor measurements from one robot position can be directly correlated to measurements taken from a nearby position. In contrast, the sonar sensor measurements are usually too noisy, ambiguous and spurious and hence it is difficult to apply the above technique to work properly. Also, it is a common belief that mobile robot navigation and mapping in indoor environments is far more difficult with sonar than with laser measurements. An alternative methodology [5, 6] to overcome limitations of the sonar sensor is to develop advanced custom-made sonar arrays that allow extracting and initializing geometric features from a single robot position. Most recently, some researchers [25, 43] have addressed this issue with a ring of sonar sensors successfully. Tardos et al. (2002) [43] proposed a new technique for perceptual feature extraction technique by using Hough Transform and map joining technique with two statistically independent and uncorrelated maps. On the other hand, Leonard et al. (2002) [25] proposed a feature initialization technique from multiple uncertain vantage points and focused on its state estimation aspects. Both these two algorithms were successfully applied to solve the SLAM problem with 24 sonar sensors in a circular array. The SLAM problem for non-circular or restricted sonar sensor array has not been addressed. This particular problem is addressed by the authors' pervious work [19].

The rest of this chapter is organized as follows: Section 2 states an overall framework of the proposed SLAM algorithm is described. Section 3 presents the complete implementation of the proposed autonomous exploration and mapping system. The performance of the proposed algorithm will be tested via robot simulator and physical mobile robot Pioneer 2DX in Section 4. Finally, the chapter will be concluded in Section 5.

2. Overall framework of the proposed SLAM with feature tracking algorithm

The authors' pervious proposed SLAM algorithm [19] which incorporate with conventional SLAM (which is introduced in appendix), enhanced adaptive fuzzy clustering EAFC feature extraction scheme [17], overlapping comparison technique and feature tracking scheme. The overall framework of this novel SLAM algorithm with feature tracking scheme is shown in the Figure 1. A step-by-step procedure for updating the estimated robot pose and map feature are presented as follows:

1. Obtain the control action and sensor measurements from the odometric sensor and sonar sensor respectively.
2. Store the sensor measurements in Overlapping sliding window sonar buffer.
3. Use the control action (odometric measurement) and present estimated robot pose to perform a SLAM prediction process (it is calculated by equation (A.3) to (A.5)).
4. Check the buffer is full (go to step 5). If the buffer is not empty (go to step 7).
5. To extract the two overlapping segments in the different group of buffer and calculate the comparison values.
6. To perform a "Feature Initialization" process (the map state matrix $X_m(k|k)$ and covariance matrix $P(k|k)$ are updated by equation (A.11) and (A.12), respectively) if the related overlapping segments comparison values are satisfied.

7. To perform a "Data Association" process to calculate sensor measurement prediction value. If the sensors can match with the existing feature in the map, go to step 10. Otherwise, go to next step.
8. Feature Tracking Scheme: Extend the current tracking feature in the map and repeat the "Data Association" again if the related comparison values are satisfied.
9. If the related sensor is matched, store the extended feature in the map. If not match, just forgo it.
10. Use the prediction value obtained in step 3 to perform the SLAM updating process (it is calculated by equation (A.6) to (A.10)).
11. The estimated robot pose and map are updated and their covariance as well. Go back to step 1.

3. Autonomous exploration strategy

In the proposed exploration system, two core components are required: a reactive navigation scheme by using hierarchical fused fuzzy system (section 3.1) and a navigation point generation system (section 3.2). When the mobile robot starts to navigate the environment, a point-mark will be placed on the acquired map sequentially. The map can be learned by either one of standard segment-based map building technique [13, 17, 43].

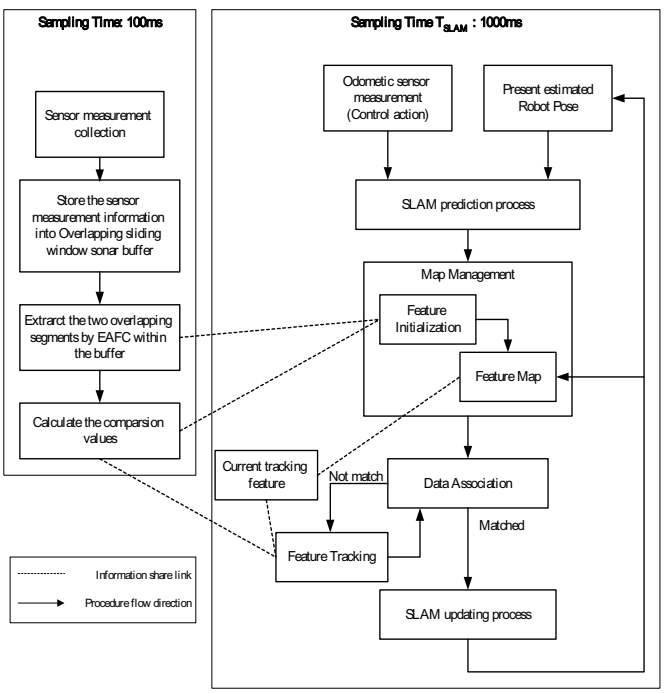


Fig. 1. The overall framework of the proposed SLAM.

The moving actions are relied on the reactive navigation system to control the mobile robot to navigate in the environment by wall following and collision avoiding technique until the mobile robot travels back to the traveled point-mark. On every point-mark, the open space evaluation system examines the eight given directions (45° apart each direction) for detecting possible areas of uncharted. An open space probability value is assigned and stored in each given direction at each traveled point-mark. In addition, the acquired map will be used to update if the open space is free or not. If the open space in a given direction at a traveled point-mark is occupied by a segment (map model), then this direction will be stated as not open space, otherwise it is registered as open space (the corresponding variables are described in section 2.2). When the mobile robot travels back to the traveled point-mark, subsequent movements by the mobile robot is required to validate whether the open space in a given direction at each point-mark actually free or not. Therefore, "A*" heuristic search algorithm [36] is used for planning routines in all traveled point-marks. This process guarantees finding the shortest path to the target location (which is the traveled point-mark still contain an open space in those given direction) from all the other traveled point-marks. Note, when the shortest path is generated by a path planning algorithm, the robot's heading is steered by the current robot direction to the next node on the path to the target location with constant velocity at a fixed sampling time. If the given direction at the traveled point-marked is stated as open space, a reactive navigation scheme will be activated again to explore in this given direction. The whole process is repeated until all the open space in a given direction at each traveled point-mark in the map are traveled or cleared by the robot or map feature, respectively.

3.1 Reactive navigation via HFFS

HFFS [4, 20, 23] uses smaller number of rules to represent the same amount knowledge, have higher mutually related interactions due to their cross-coupling between each element and level. Due to these properties, this structure is applied to design a reactive navigation controller to control the mobile robot to achieve some task, such as: keep off obstacles and wall or corridor following.

The schematic diagram of the HFFS is shown in Figure 2. In this system, six 2-input / 1-output fuzzy system are used. This HFFS consist 7 inputs and 2 outputs and the total number of rules are 60 only.

The strategy of creating the fuzzy rules is that decision should try to move forward, navigate along a corridor and at corner, keep off and parallel to wall and avoid obstacle. The antecedent variables to the HFFS are 5 sides' sonar sensors reading (i.e. left and right side, left and right front corner and front side) and 2 changes of side sensor readings (left and right).

The input sensor readings are fuzzified using the fuzzy set definitions as shown in Figure 3a. The variable is partitioned in three fuzzy sets namely, VN (very near), NR (near) and FR (far). The sensor readings are normalized between 0 and 1. The input membership function for the two changes of side sensor readings is shown in Figure 3b. Furthermore, the variable is partitioned in three fuzzy sets namely, N (negative), Z (zero) and P (positive). The change of side sensor reading is normalized between –1 and 1. The input L_M and R_M inside the HFFS are fuzzified using the fuzzy membership functions as shown in Figure 3c and they are normalized between –1 and 1. The labels of each subset are N (negative), Z (zero), VS (very slow), M (medium) and F (fast). Since normalized input variables are used. Four input scaling factors should be used, such as L_S' and R_S' sensor reading scaling factor σ_S, F_S'

sensor reading scaling factor σ_F, change of side sensor reading (ΔL_S and ΔR_S) scaling factor $\sigma_{\Delta S}$ and L_M and R_M velocity scaling factor σ_V.

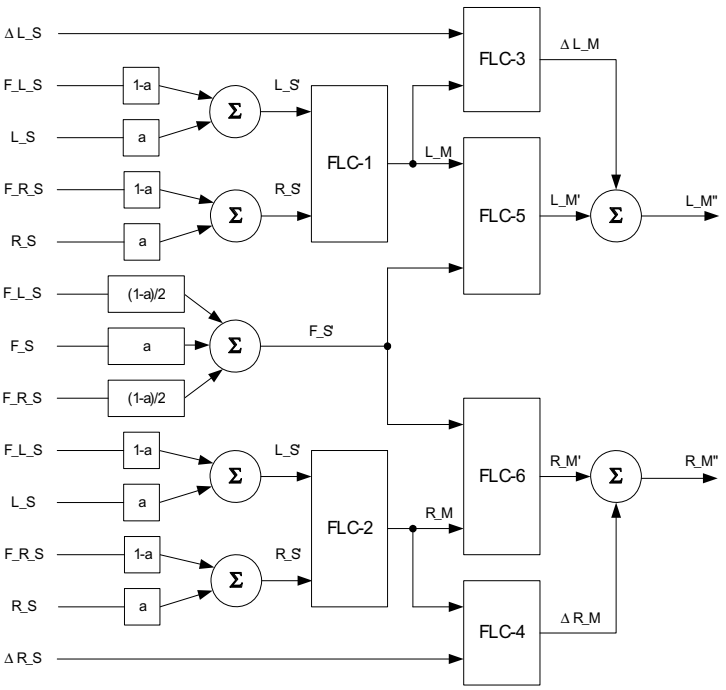
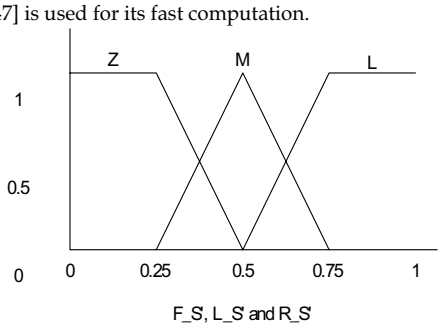


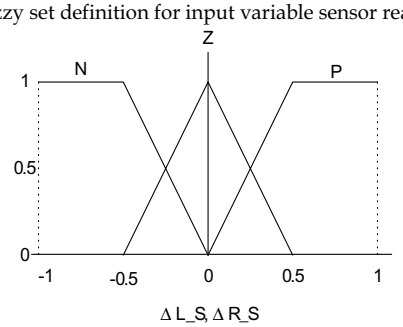
Fig. 2. The HFFS reactive navigation structure.

The singleton fuzzy set is used for the all output variables (L_M, L_M′, R_M, R_M″, ΔL_M and ΔR_M) in HFFS and it is shown in Figure 4. The fuzzy partition names for L_M, L_M′, R_M and R_M″ are same as that used in input L_M or R_M. The normalized ΔL_M and

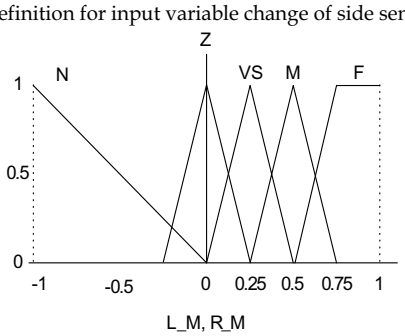
ΔR_M are partitioned into five fuzzy sets, namely NL (negative large), NS (negative small), Z (zero), PS (positive small) and PL (positive large). All the output variables are normalized between –1 and 1. The output scaling factor for the motor velocity L_M′ & R_M′ and the change of motor velocity ΔL_M & ΔR_M are stated as σ_V (same as the input scaling factor used in L_M and R_M) and $\sigma_{\Delta V}$, respectively. In the present implementation the center average de-fuzzifier [47] is used for its fast computation.



a) Fuzzy set definition for input variable sensor reading.

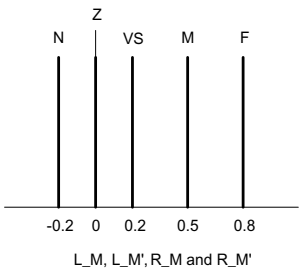


b) Fuzzy set definition for input variable change of side sensor readings.

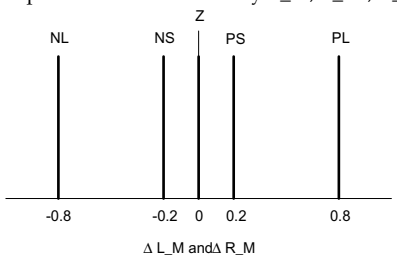


c) Fuzzy set definition for input variable motor velocity.

Figure 3 Input membership functions corresponding to HFFS.



a) Output consequences for motor velocity L_M, L_M′, R_M and R_M′.



b) Output consequences for the change of motor velocity ΔL_M and ΔR_M .

Fig. 4. Output consequences corresponding to the HFFS.

The corresponding fuzzy rule tables used in HFFS are shown in Table 1. From the rule table (Table 1c), we defined that the robot will turn right when both sides′ sensor readings are equal or a nearer obstacle occupied in left side.

O/P: L_M / R_M		R_S'		
		VN	NR	FR
L_S'	VN	Z / Z	VS / Z	M / N
	NR	Z / VS	M / M	F / M
	FR	N / M	M / F	F / F

a) Fuzzy rule base for FLC-1 and FLC-2 in HFFS.

O/P: ΔL_M or ΔR_M		L_M or R_M				
		N	Z	VS	M	F
ΔL_S	N	X	X	X	PS	PL
or	Z	X	X	X	Z	Z
ΔR_S	P	X	X	X	NS	NL

b) Fuzzy rule base for FLC-3 and FLC_4 in HFFS.

O/P: L_M' / R_M'		L_M / R_M				
		N	Z	VS	M	F
F_S'	VN	N	N	Z	A_1 / B_1	A_1 / B_1
	NR	N	VS	Z	A_2 / B_2	A_2 / B_2
	FR	N	VS	VS	M	F

c) Fuzzy rule base for FLC-5 and FLC_6 in HFFS.

Table 1. Fuzzy rule tables for the proposed HFFS reactive navigation controller.

3.2 Navigation point generation system

The objective of this system is to place and store a point-mark on the traveled robot trajectory and hence to search all possible open space on the mapping environment. Before we describe the system, few types of points and terms are introduced in Table 2. The main feature of this navigation point generation system is to arrange the navigation point in unevenly distribution. Different to the navigation system proposed by Duckett [10]. Duckett [10] suggests adding a "place" (equivalent to navigation point used in here) at every certain distance (1 m is used in their experiment). Therefore, the resulting topological path was distributed evenly and some of redundant "place" or navigation point was occurred. In contrast, we suggest adding a confirmed navigation point in a required region. For example: if the mobile robot navigates in a long corridor, few navigation points are required to represent the free path. Therefore, 8 possible free space directions and its related free space probability are assigned at each confirmed navigation point. We can compare their state (8 free space probability) of navigation point to past navigation point to verify their similarity when the mobile robot navigated in an unknown environment. The calculation of free space probability in a given direction at each navigation point is discussed in section 3.2.3

Symbol	Definition
T_NPn	n^{th} Test Navigation Point
P_NP	Potential Navigation Point
Cd_NP	Confirmed Navigation Point
L_Cd_NP	Last Cd_NP
N_Cd_NP	The Nearest Cd_NP to the current robot location
T_Cd_NP[i] where i = 0, 1, 2, 3	ith recent Traveled Cd_NP (The most recent Cd_NP is represented by i = 0)
E_Cd_NP	The Extra Past Cd_NP, i.e. At time k E_Cd_NP ≠T_Cd_NP[i] for i = 0, 1, 2, 3.
FreeSpaceCoverRadius	The maximum free space coverage radius
MinNavTravelDis	The minimum specified traveled displacement

Table 2. The Various symbols in Navigation point generation system.

In addition, three terms will be attached in each given direction at each Cd_NP, such as "FreeSpaceProbability (= 0 ~ 1)", "IsOpenSpace (= True or False)" and "IsExplored (= True or False)".

Figure 5 is a flow chart of the entire navigation point generation system. It is designed to allow direct translation into an implementation. In Figure 5, two types of line are used, i.e. dashed-line and solid-line. The dashed-line stated that the process should go to next step within same sampling interval. In contrast, the solid-line stated that the next step would be executed at next sampling interval. The flow of the algorithm is regulated by the state variables described in Table 2. The significance of some parts of the flow chart necessitates discussion. These regions have been labeled in Figure 4 and are described as following:

- **A** The state of the last confirmed navigation point (L_Cd_NP) is updated by 10 sampling robot steps before it go to next step. The state of Cd_NP updating process will be discussed in section 3.2.1.
- **B** When Condition 1 (which stated in Figure 4) is satisfied, a new confirmed navigation point (Cd_NP) is registered and reset the variable *n* (which states the total number of test navigation point) to 1. And then calculate and update the state of this new registered Cd_NP in next sampling interval.

C When the current test navigation point (T_NPn) is similar to the extra past confirmed navigation point (E_Cd_NP), then reset the current test navigation point and set n to 1. After that, register a new test navigation point in next sampling interval.

D If the current test navigation point is not similar to the current potential navigation point, then search the nearest confirmed navigation point (Cd_NP) in the list relative to the current potential navigation point. If this resulting point is similar to the current potential point and their distance is smaller than "MinNavTravelDis", then reset the current test navigation point and set n to 1.

3.2.1 Process for updating the state of confirmed navigation point (Cd_NP)

As mentioned that as before, three variables are attached in each given direction of each confirmed navigation point (Cd_NP). One floating point variable, i.e. "FreeSpaceProbability". The updating process of this variable will be introduced in section 3.2.3. Two Boolean variables, i.e. "IsOpenSpace" and "IsExplored", and the updating condition are discussed as follows:

"IsOpenSpace" This Boolean variable can be stated that the given direction of confirmed navigation point is open or not. And it can be determined by the "FreeSpaceProbability". If the free space probability is more than or equal to 0.5, then this given direction is open (IsOpenSpace = True). Otherwise, "IsOpenSpace" is equal to "False". On the other hand, this Boolean variable can be also updated by the acquired map model. Since a segment-based map is extracted by either one of segment-based map building technique simultaneously. (The corresponding map building will be discussed in section 4.) A simple logical updating algorithm is applied. If the given direction of confirmed navigation point is occupied by a segment, then the Boolean variable can be stated as "False". Otherwise, it is registered as "True".

"IsExplored" This Boolean variable states that the given direction of confirmed navigation point is explored or not. Three conditions are used to update this variable and are shown as following:

1. If the distance between two confirmed navigation points is smaller than FreeSpaceCoverRadius, then these Boolean variables (in the corresponding given direction at the two confirmed navigation points are registered as "True" (i.e. IsExpored = True).
2. If the distance between the current robot position and the corresponding confirmed navigation point is smaller than FreeSpaceCoverRadius, then it is stated as "True".
3. If the term "IsOpenSpace" is stated as "False" in a given direction at the corresponding confirmed navigation point, then the term "IsExplored" is equal to True in the same direction at the corresponding confirmed navigation point.

3.2.2 Redundant confirmed navigation point removing process

As we want to reduce the redundant confirmed navigation point in the map, a redundant point removing process is needed. In this process, we compare the T_Cd_NP[0] and T_Cd_NP[1]. If it is similar and the distance between T_Cd_NP[0] and T_Cd_NP[2] is smaller than FreeSpaceCoverRadius , then remove the T_Cd_NP[1]. The visualization of the proposed exploration algorithm is shown in Figure 6.

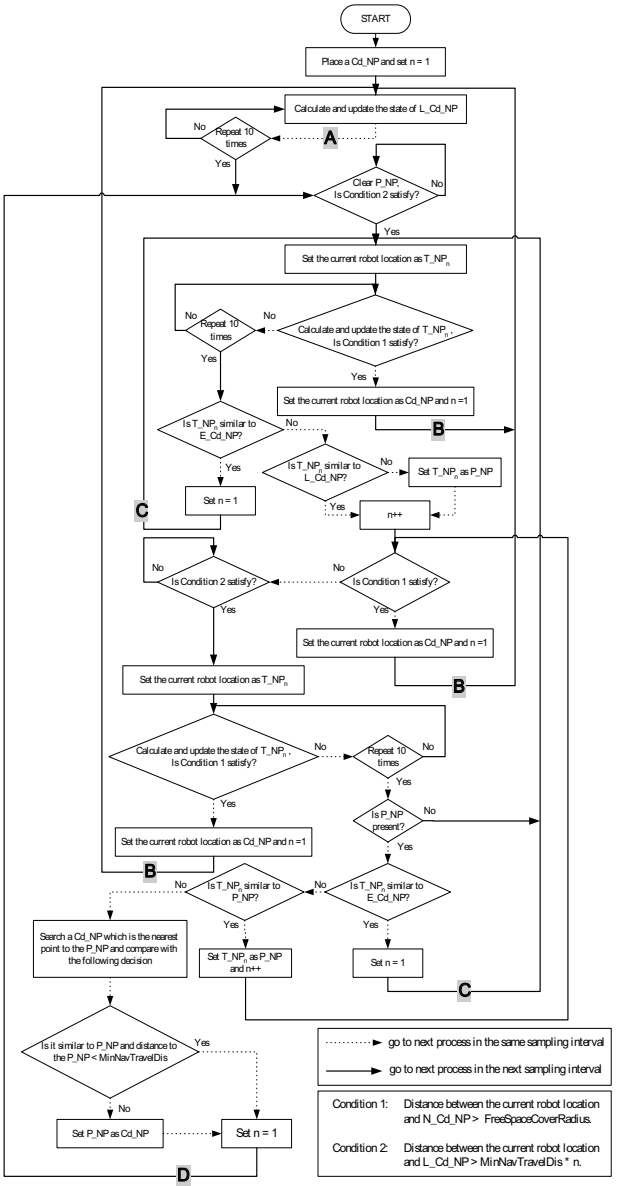


Fig. 5. The flow chart of the proposed navigation point generation system.

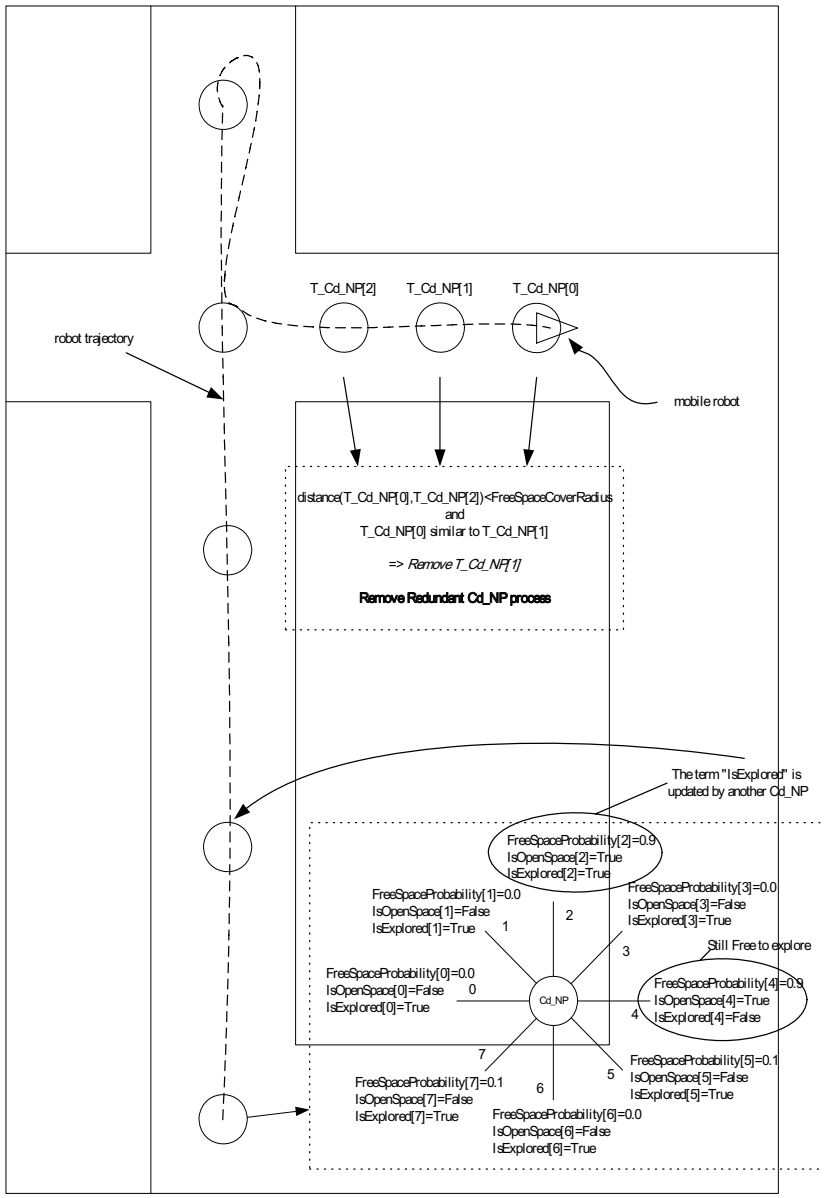


Fig. 6. The visualization of exploration algorithm.

3.2.3 Open space evaluation system

To evaluate areas of free space, we have developed a simple and efficient mechanism which uses a Bayesian update rule and sonar sensor model *pdf* to calculate the value of "FreeSpaceProbability" in a given direction at each point-mark (T_NP or Cd_NP). From equation (A-2), the sonar probability model $p(r \mid z, \theta)$ can be calculated. r is the sensor range reading. z represents the true parameter space range value (here we use the value of "FreeSpaceCoverRadius"). θ represents the azimuth angle measured with respect to the beam central axis (here we calculated as sonar beam angle θ_S subtract the given direction at a point-mark, all angles are referred to the global reference frame). In this application, the value k_ε and k_o become a constant in equation (A-2). i.e. k_ε = zero or very small real number and k_ε = 0.6. The variance of radial σ_r and angular σ_θ is equal to 250mm and 12.5°, respectively. After a sampling for all sonar sensors (16 sonar sensors used here), the probability is fused together by Bayes's theorem [1]. The value of "FreeSpaceProbability" in given direction is calculated as below:

$$P[NavDir_j = FREE \mid r] = \frac{p[r \mid NavDir_j = FREE]P[NavDir_j = FREE]}{\sum_{NavDir_j} p[r \mid NavDir_j]P[NavDir_j]} \tag{1}$$

for $j = 0, 1, 2, \ldots, 7$.

where $p[r \mid NavDir_j = FREE]$ corresponds to sensor *pdf* (for calculate the probability of free space, it is calculated as $(1 - p(r \mid z, \theta))$),

$P[NavDir_j = FREE]$ is prior presented free space probability of a given direction at a point-mark $NavDir_j$,

$P[NavDir_j = FREE \mid r]$ is new presented free space probability of $NavDir_j$ based on r observations.

The whole process is repeated until the robot has traveled far away from the point-mark with certain distance (300mm in these experiment). After the open space evaluation process is complete, 8 probability values (see bottom right in Figure 6) are attached in the point-mark (T_NPn or Cd_NP) for representing the free space probability values in given direction. Furthermore, another function for this open space evaluation system is aimed to a mobile robot escaping from a "U-trap" (see the upper part in Figure 6). We can detect a "U-trap" situation from the following 2 conditions:

1. L_S'<600mm and R_S'<600mm
2. An "IsOpenSpace[i]" in point T_Cd_NP[0] is equal FALSE, where i is equal to the direction (0 ~ 7) which is pointed to the current position of mobile robot.

If a "U-trap" is detected, the reactive navigation system will be disabled until the mobile robot is rotated for certain degree (120° in these experiments).

4. Experimental Results

In order to evaluate the performance of the proposed autonomous exploration strategy, several experiments were conducted. In the first two experiments, we applied the proposed reactive navigation system only to control a Pioneer 2DX mobile robot to navigate along a corridor and at corners. For third experiment, the complete autonomous exploration

strategy was conducted in a localization error free (no slippage or wheel drift error) mobile robot simulator platform (Pioneer simulator). For fourth experiment, the autonomous SLAM experiment was conducted in a well constructed unknown indoor environment with Pioneer 2DX mobile robot.

All of experiments were implemented in robot software "Saphira" [42] on a Pentium 4 1.6 GHz with 256RAM PC computer and it is communicated to the Pioneer 2DX mobile robot via wireless modem. Also, all sonar sensors measurements were limited up to 3 meters to reduce the uncertainty for map building process. In our proposed exploration scheme, two sides sensors and front sonar sensors array are used only. They are arranged in 5 input groups as shown in Figure 7 (i.e. L_S, L_F_S, F_S, R_F_S and R_S). The 10 sensors were categorized into 5 inputs for HFFS reactive navigation system and it is allocated as follows:

Left side: L_S = min(S_{16},S_1)

Left front corner: L_F_S = min(S_2,S_3)

Front side: F_S = min(S_4,S_5)

Right front corner: R_F_S = min(S_6,S_7)

Right side: R_S = min(S_8,S_9)

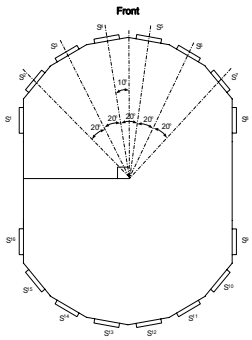


Fig. 7. Sensors arrangement for HFFS reactive navigation system.

The values (range) from direction left to right (i.e. L_S, L_F_S, F_S, R_F_S and R_S) were used as the input of the HFFS reactive navigation system and the outputs of the system were the linear velocity of the left (L_M") and right (R_M") driving motor of the mobile robot. For scaling factor σ_S, a simple adoption scheme was used and it is formulated as follows:

$$\sigma_S(t) = \begin{cases} \sigma_S(t-1) + \lambda(\sigma_{MAX} - \sigma_S(t-1)) & if\ (\text{R\_S} >= \sigma_{MAX})\ \&\ (\text{L\_S} >= \sigma_{MAX}) \\ \sigma_S(t-1) + \lambda(\text{L\_S} - \sigma_S(t-1)) & if\ (\text{R\_S} >= \sigma_{MAX})\ \&\ (\text{L\_S} < \sigma_{MAX}) \\ \sigma_S(t-1) + \lambda(\text{R\_S} - \sigma_S(t-1)) & if\ (\text{L\_S} >= \sigma_{MAX})\ \&\ (\text{R\_S} < \sigma_{MAX}) \\ \sigma_S(t-1) + \\ \lambda((\text{R\_S} + \text{L\_S}) - \sigma_S(t-1)) & if\ (\text{L\_S} < \sigma_{MAX})\ \&\ (\text{R\_S} < \sigma_{MAX}) \end{cases} \quad (2)$$

where σ_{MAX} is the maximum defined range value and is equal to FreeSpaceCoverRadius and

λ is a forgetting factor (0.7 in these experiments).

The initial value $\sigma_S(0)$ is defined as 1500mm. The setting of reminding input and output scaling factors are stated as follows:

σ_F is defined as FreeSpaceCoverRadius.

$\sigma_{\Delta S}$ is defined as 100mm (designed by human experience).

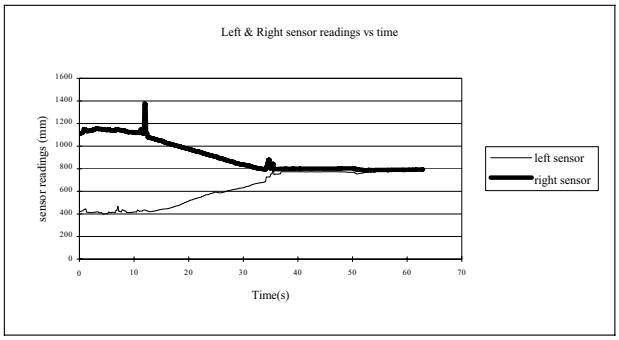
σ_V is defined as 200mm/s (predefined maximum translation velocity).

$\sigma_{\Delta V}$ is defined as 20mm/s (designed by human experience).

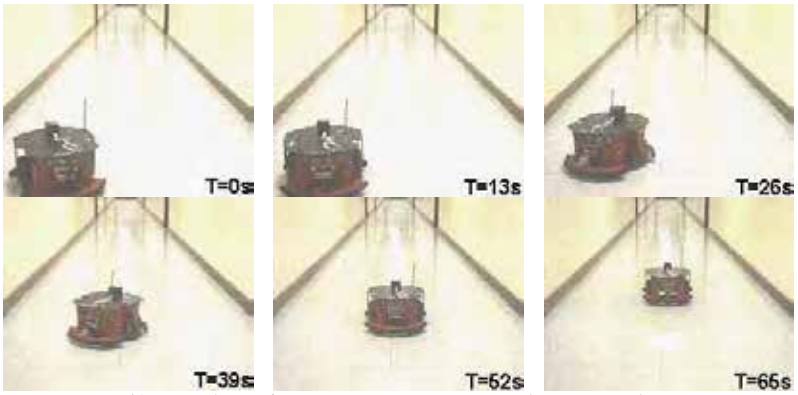
The sampling time for the proposed autonomous exploration strategy is 1s in all the following experiments.

4.1 Experiments with a real robot

In the first experiment, the system was tested in a long corridor with 1.5m widths. The objective of this experiment was to verify the performance when a mobile robot navigated along a corridor. Therefore, the minimum range value of the left and right side group sensors are plotted against time and it is shown in Figure 8a. In Figure 8a) and b), shows that the Pioneer 2DX mobile robot navigated along towards the middle of corridor with a smooth trajectory.



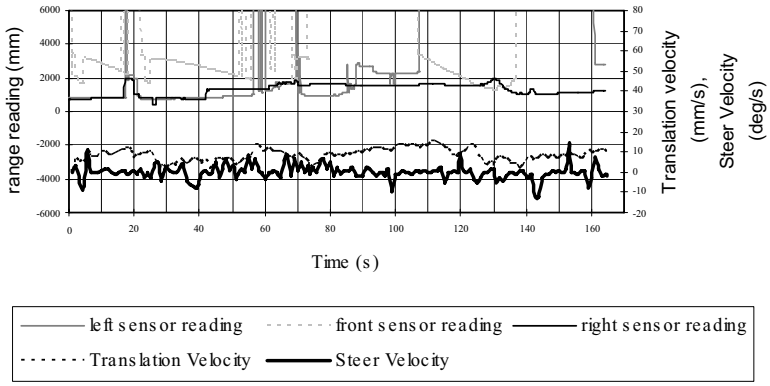
a) Left and Right wall distance measured by the left and right sonar.



b) Snap shots of a Pioneer 2DX navigating along a corridor.

Fig. 8. The performance of the proposed HFFS reactive navigation system while navigates along a corridor.

In the second experiment, the proposed reactive navigation system was used to control a Pioneer 2DX navigating in a more complex area where is located at the outside of our research laboratory in the university. Figure 9 shows the robot's information and the robot trajectory during navigation. At starting of the navigation (low bottom left in Figure 9b), the mobile robot traveled along a corridor. Then the mobile robot turned to right side when the robot's front sensor detected an obstacle (at time 70s, see Figure 9a). Then the mobile robot started to follow a set of lockers (by wall following behavior) until it's front sensor detect an obstacle again. Finally, it started to follow right hand side object at time 140s.



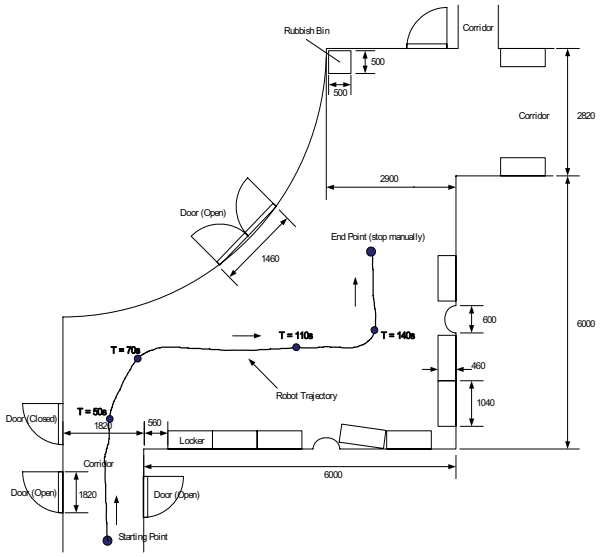


Fig. 9. The robot's information and robot trajectory while a Pioneer 2DX navigated at corner.

From the above two experiments, it can be demonstrated that the proposed HFFS reactive navigation system can achieve the goal of multi-behavior (such as: navigate along a corridor and at corner, keep off and parallel to wall and avoid obstacle) mobile robot controller. In the next experiment, the complete autonomous exploration strategy is applied to control a mobile robot for navigating in an unknown environment via robot simulator.

4.2 Experiment with a robot simulator

In this experiment, the EAFC segment-based map building algorithm [15] was adopted to extract the map information from raw sonar data. This map building algorithm is the authors' pervious work [17]. Other than that algorithm, we can also apply fuzzy sonar maps [13] (which was proposed by Gasos and Martin 1996) or Hough transform with sonar arc (which was proposed by Tardos *et. al.* 2002) for extracting a segment-based map. For the parameters setting in autonomous exploration strategy, it was selected as follow: "FreeSpaceCoverRadius" = 2500mm and "MinNavTravelDis" = 800mm.

The advantage for using a robot simulator to verify our proposed autonomous exploration strategy is that the localization error can be disabled or neglected. Since the localization problem will arise an error or affect the accuracy in the planning process. The Pioneer Simulator [42] can simulate several different types of typical noise that occur during robot navigation and sensor perception. To achieve the goal of this experiment, the percentage of encoder jitter, angle jitter and angle drift in robot simulator is reduced to zero. Nevertheless, the sonar sensor uncertainty is still occurring in the system. Figure 10 shows the navigation point-marks and the unexplored direction at each Cd_NP superposed on the actual map when the Pioneer 2DX navigates in the simulation world. We can see that the mobile robot can navigate in all regions in the unknown environment. Also, the navigation point-marks are distributed unevenly in the navigation environment. The raw sonar data and extracted map by EAFC during the autonomous navigation are shown in Figure 11 a) and b), respectively.

4.3 Autonomous SLAM experiment

In this experiment, the autonomous exploration strategy was combined with the SLAM algorithm [19] to form an effective SLAM algorithm. Basically, this effective SLAM algorithm is similar to the algorithm that was tested in section 4.2 except the map information (for aiming the navigation point generation system) is replaced by the SLAM map. An overview of the system architecture is shown in Figure 12. Since this was a real-time experiment, it was difficult to obtain a ground truth robot trajectory. Therefore, we used the authors' previous proposed fuzzy tuned extended Kalman filter FT-EKF model-based localization algorithm [18] to measure the robot trajectory during the autonomous SLAM process for comparison. The system was tested in our research office (8×8 m) and the floor plan.. The total trajectory of the mobile robot was around 30m, lasting around 20 minutes

The sampling rate of SLAM process and autonomous exploration strategy was 1000ms. The parameters settings for the autonomous exploration strategy were selected as: "FreeSpaceCoverRadius" = 2000mm and "MinNavTravelDis" = 700mm.

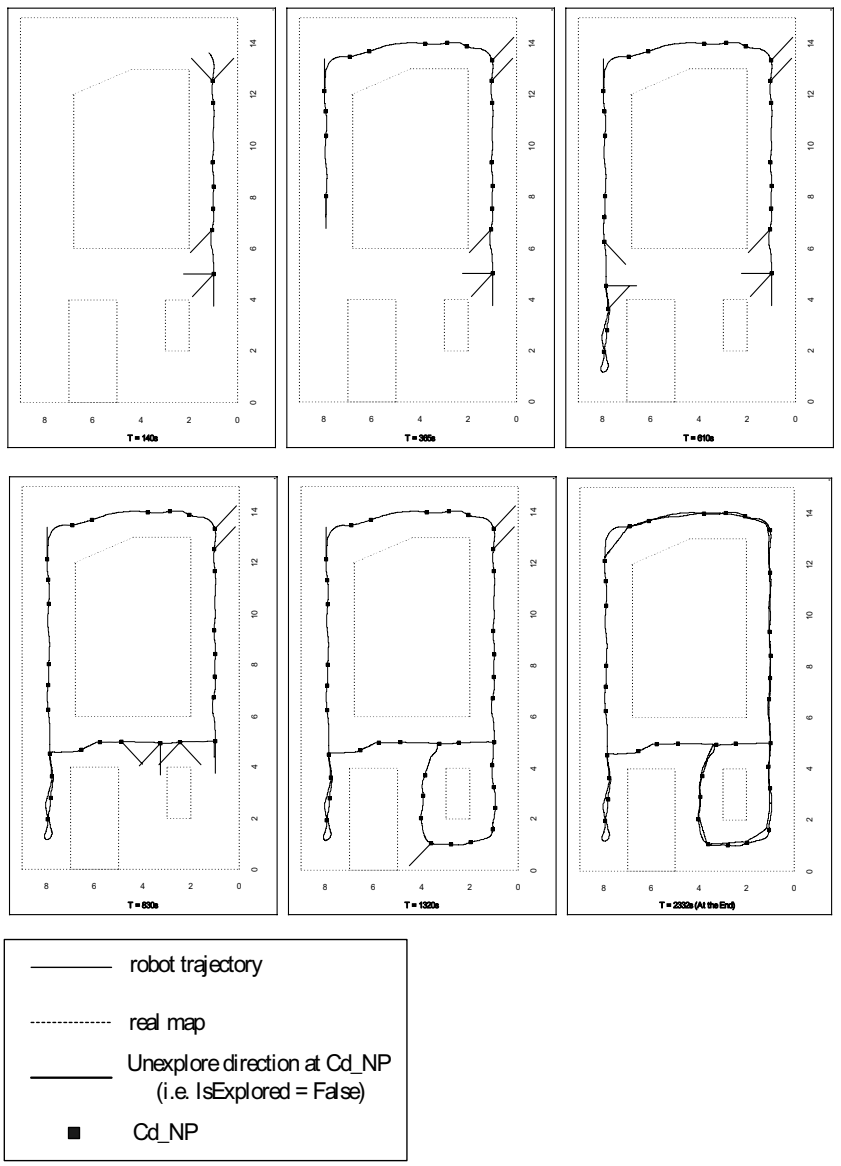
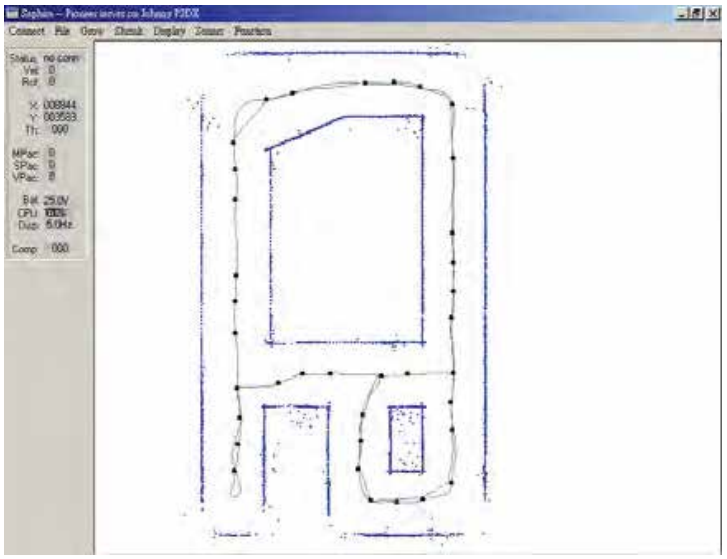
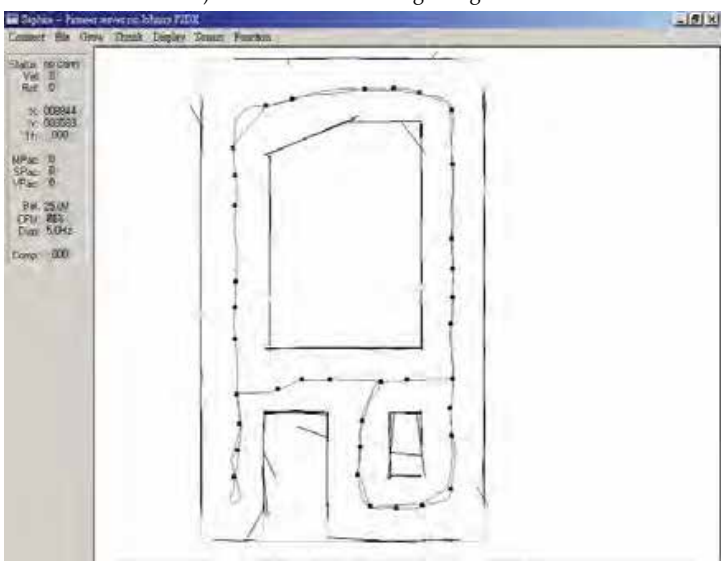


Fig. 10. Snap Shots for the Pioneer 2DX mobile robot navigating in the simulation world.



a) Raw sonar data during navigation.



b) (black line) Extracted line segments superposed on (gray line) real map.

Fig. 11. Robot trajectory, navigation point-marks, extracted map, raw data and real map captured from the robot software Saphira.

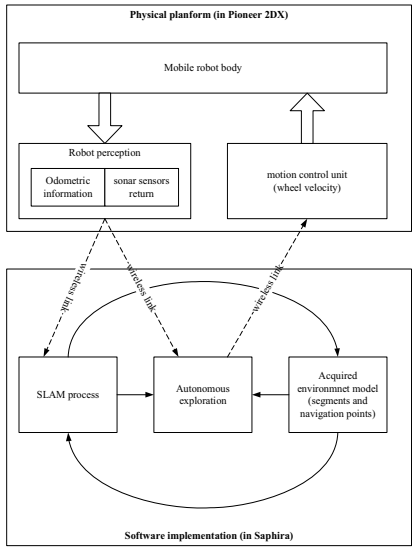
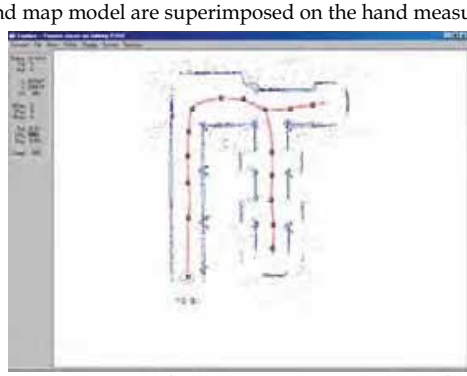
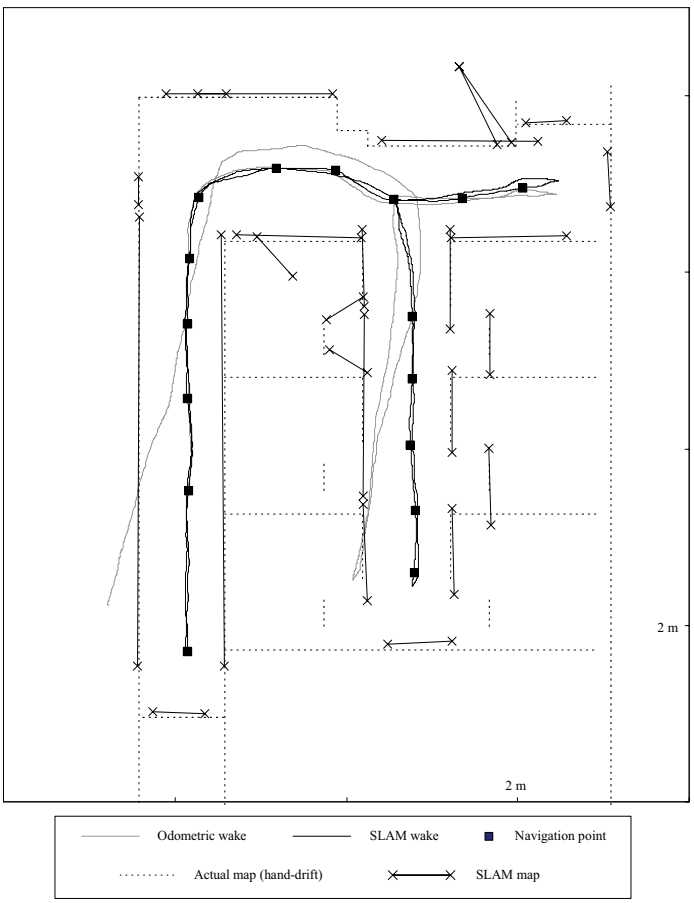


Fig. 12. Overall architecture of the proposed autonomous SLAM mechanism.

At the start of the experiment, the Pioneer 2DX was placed at end of the corridor (shown in lower left corner in Figure 13a). After all the given directions at each navigation point were navigated, the mobile robot traveled back to the starting position. The final global map acquired at end of the experiment is shown in Figure 13b. In addition, 25 line features and 16 navigation points were extracted in the final map and the final absolute position error in X and Y is 50mm and 64mm (measured by hand and relative to actual position), respectively. For comparison purposes, the odometric wake, the SLAM wake, extracted navigation points and map model are superimposed on the hand measured map model.



a) Sonar returns, navigation points and autonomous SLAM estimated wake obtained during the experiment. (Captured from the robot software "Saphira".) The range threshold of all sonar sensors is 1500mm. Therefore, a lot of ambiguous and noise measurements were filtered.



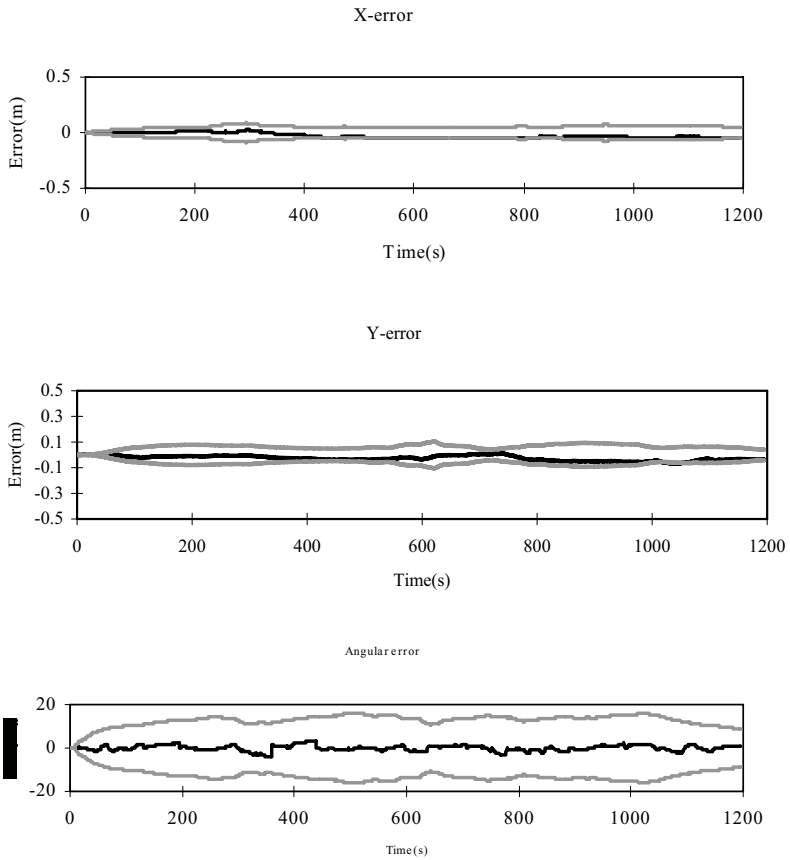
b) Extracted map model and navigation points superposed on the real map.

Fig. 13. Robot trajectory, navigation point-marks, extracted map, raw data and real map during the autonomous SLAM experiment.

To further analyze the consistency of our integrated approach, Figure 14 shows a comparison between the error in the autonomous SLAM pose versus model-based FT-EKF robot pose along with the 2-sigma (2σ) uncertainty bounds logged from the SLAM process. It is clearly demonstrated that those errors remain inside their 2σ uncertainly bounds at the most of time. From this on-line integrated experiment, we conclude that this approach can fulfill the three essential missions of mobile robot and those are operated in real time and simultaneously. Figure 15 shows snap shots captured from the robot software "Saphira", during the experiment.

5. Conclusions

In this chapter, a new autonomous exploration strategy for mobile robot was presented and extensively tested via simulation and experimental trials. The essential mechanisms used included a HFFS reactive navigation scheme, EAFC map extraction algorithm, SLAM process, an open space evaluation system cooperating with probability theory and Bayesian update rule and a novel navigation point generation system. The proposed autonomous exploration algorithm is a version of combination of a robust reactive navigation scheme and approaching the unknown strategy which ensure that the mobile robot to explore the entire region in an unknown environment automatically.



Fi

Fig. 14. Estimated errors in robot location during the autonomous SLAM process with sonar. (Gray lines represent two-sigma (2σ) uncertainly bounds.)

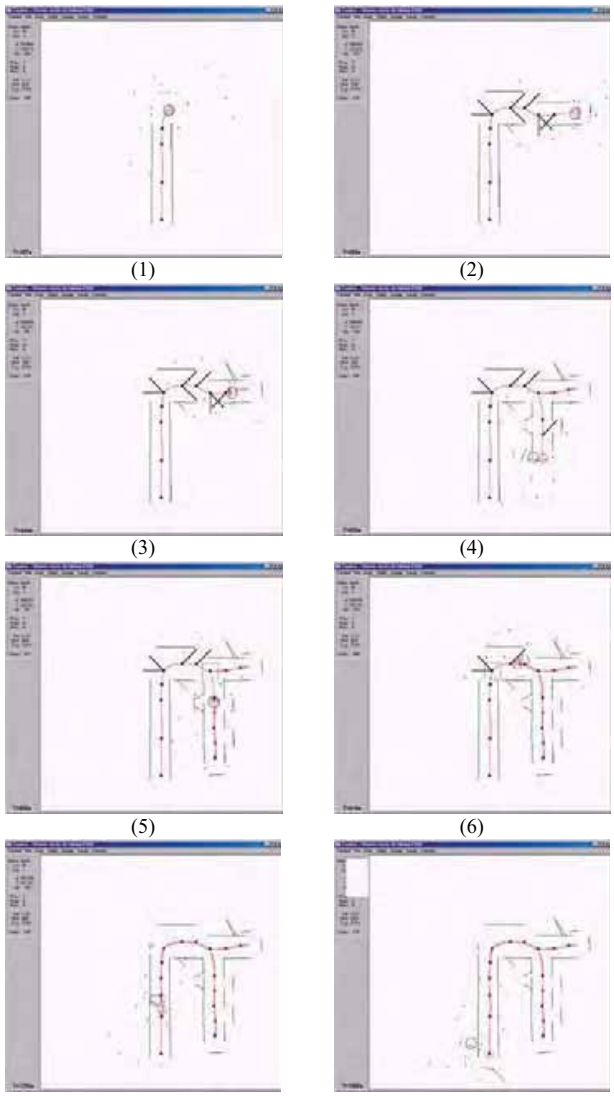


Fig. 15. Snap shots during autonomous SLAM process via Pioneer 2DX mobile robot. (Captured from the robot software "Saphira".) The black robot (a bit bigger) represents the robot position estimated by odometric. The gray robot represents the robot position estimated by SLAM process.

In addition in this chapter, a metric topological map model is advocated for facilitating the path planning process during the autonomous exploration. Moreover, the map model extracted from an EAFC map building algorithm (metric map model) is aimed to generate the navigation point or node on the navigation path. Therefore, a hybrid map model is proposed for autonomous map building in an unknown indoor environment. An autonomous map building algorithm was tested in a simulation world (section 4.2). On the other hand, a successful on-line autonomous SLAM experiment (section 4.3) was conducted for a mobile robot to map an indoor and unknown environment.
Basically, this chapter concluded the pervious work: a SLAM problem solved by overlapping sliding window sonar buffer [Ip and Rad 2003] and EAFC feature initialization technique []Ip and Rad] combined with a novel autonomous exploration strategy to formulate an autonomous SLAM mechanism. Experimental studies demonstrated that the mobile robot was able to build a segment-based map and topological map (a list of navigation points) in real time without human intervention.

6. References

1. Berger, J.O., Statistical Decision Theory and Bayesian Analysis. Springer-Verlag, Berlin. Second Edition, 1985.
2. Borenstein, J. and Koren, Y., Obstacle avoidance with ultrasonic sensors, IEEE Transactions on Robotics and Automation 4 (1988) 213–218.
3. Castellanos, J.A. and Tardos, J.D.: *Mobile robot localization and map building: A multisensor fusion approach*. Boston, MA: Kluwer Academic Publishers (1999)
4. Chan, P.T. and Rad A.B., Optimization of Hierarchical and Fused Fuzzy Systems with Genetic Algorithms, Joint Conference on Intelligent Systems 1998 (JCIS'98), Oct. 23-28, Research Triangle Park, North Carolina, U.S.A., Proceedings Vol. 1, 240-243.
5. Chong, K.S. and Kleeman, L. "Feature-Based mapping in real, large scale environments using an ultrasonic array". *The International Journal of Robotics Research*, Vol. 18, No. 1, pp. 3-19 (1999a)
6. Chong, K.S. and Kleeman, L. "Mobile robot map building from an advanced sonar array and accurate odometry". *The International Journal of Robotics Research*, Vol. 18, No. 1, pp. 20-36 (1999b)
7. Crowely, J.L., Navigation for an Intelligent Mobile Robot. IEEE Journal of Robotics and Autonmation 1 (1), pp. 34-41, 1985.
8. Dissanayake, M.W.M.G., Newman, P., Durrant-Whyte, H., Clark, S. and Csorba, M. "A solution to the simultaneous localization and map building (SLAM) problem". *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 3, pp. 229-241. Dec. (2001)
9. Dubrawski, A. and Crowley, J.L., Learning locomotion reflexes: A self-supervised neural system for a mobile robot, Robotics and Autonomous Systems 12 (1994) 133–142.
10. Duckett, T.D., Concurrent map building and self-localization for mobile robot navigation, PhD. Dissertation, In The University of Manchester, 2000.
11. Drumheller, M., Mobile robot localization using sonar. IEEE Transactions on Pattern Analysis and Machine Intelligence 9(2), pp. 325-332, 1987.
12. Edlinger, T. and Weiss, G., Exploration, navigation and self-localisation in an autonomous mobile robot. In Autonomous Mobile Systems, 1995.
13. Gasos, J. and Martin A., A fuzzy approach to build sonar maps for mobile robots, Computers in Industry 32 pp. 151-167, 1996.

14. Gibbons, A. Algorithmic graph theory. Cambridge, England: Cambridge University Press. 1985.
15. Goodridge, S.G., Kay, M.G. and Luo, R.C., Multi-layered fuzzy behavior fusion for reactive control of an autonomous mobile robot, in: Proceedings of the Sixth IEEE International Conference on Fuzzy Systems, Barcelona, 1997, pp. 579–584.
16. Guivant, J.E. and Nebot, E.M. "Optimization of the simultaneous localization and map building algorithm for real time implementation". *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 3, pp 242-257 Dec. (2001)
17. Ip, Y.L., Rad, A.B., Chow, K.M. and Wong, Y.K.: Segment-Based map building using enhanced adaptive fuzzy clustering algorithm for mobile robot applications, *Journal of Intelligent and Robotic Systems* 35, (2002), pp 221-245.
18. Ip, Y.L., Studies on Map Building and Exploration Strategies for Autonomous Mobile Robots (AMR), PhD. Dissertation, In The Hong Kong Polytechnic University, 2003.
19. Ip, Y.L. and Rad, A.B. "Incorporation of Feature Tracking into Simultaneous Localization and Map Building via Sonar Data". *Journal of Intelligent and Robotic Systems* 39: 149-172, 2004.
20. Jamshidi, M., Large-scale systems modelling, Control and Fuzzy Logic, Prentice-Hall, Englewood Cliffs, U.S.A. (1996)
21. Konolige, K., A Gradient Method for Realtime Robot Control, In Proceedings of International conference on Intelligent Robots and Systems IROS 2000.
22. Koren, Y. and Borenstein, J., Potential methods and their inherent limitations for mobile robot navigation, in: Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, 1991, pp. 1938–1404.
23. Lacrose, V. and Titli, A. Fusion and Hierarchy can Help Fuzzy Logic Controller Designers, Proc. of sixth IEEE Intern. Conf. of Fuzzy Systems. pp.1159-1165. (1997)
24. Lee, David. The Map-Building and Exploration Strategies of Simple Sonar-Equipped Mobile Robot. Cambridge University Press, 1996.
25. Leonard, J.J., Rikoski, R.J., Newman, P.M. and Bosse, M. "Mapping partially observable features from multiple uncertain vantage points". *The International Journal of Robotics Research*, Vol. 21, No. 10, pp.943-976 (2002)
26. Li, W., Fuzzy-logic based reactive behavior control of an autonomous mobile system in unknown environments, Engineering Application of Artificial Intelligence 7 (8) (1994) 521–531.
27. Li, W., A hybrid neuro-fuzzy system for sensor based robot navigation in unknown environments, in: Proceedings of the American Control Conference, pp. 2749–2753, 1996.
28. Lin, C.H. and Wang L.L., Intelligent collision avoidance by fuzzy logic control, Robotics and Autonomous Systems 20 (1997) 61–83.
29. Lu, F. and Milios, E. "Globally consistent range scan alignment for environment mapping". *Autonomous Robots* 4, pp 333-349 (1997)
30. Lumelshy, V. and Skewis, T., Incorporating range sensing in the robot navigation function, IEEE Transactions on Systems, Man and Cybernetics 30 (1990) 1058–1069.
31. Maeda, M., Maeda, Y. and Murakami, S., Fuzzy drive control of a autonomous mobile robot, Fuzzy Sets and Systems 39 (2) (1991) 195–204.
32. Mataric, M., Navigating with a rat brain: A neurobiologically inspired model for robot spatial representation. In From Animals to Animats 1, 1991.

33. Millan, J.R., Rapid, safe, and incremental learning of navigation strategies, IEEE Transactions on Systems, Man and Cybernetics 26 (3) (1996) 408–420.
34. Musilek, P., Neural networks in navigation of mobile robots: A survey, Neural Network World 6 (1995) 917–928.
35. Nagrath, I.J., Behera, L., Krishna, K.M. and Rajshekhar, K.D., Real-time navigation of a mobile robot using Kohonen's topology conserving neural network, in: Proceedings of the International Conference on Advanced Robotics, pp. 459–464, 1997.
36. Nilsson, N., Principles of Artificial Intelligence. Palo Alto, CA: Tioga, 1980.
37. Nomad 200 User's Manual. Mountain View, CA: Nomadic Technol., 1996.
38. Pal, P.K. and Kar, A., Mobile robot navigation using a neural net, in: Proceedings of the IEEE International Conference on Robotics and Automation, Nagoya, Japan, pp. 1503–1508, 1995.
39. Pioneer 2 Mobile Robot Operation Manual. May 2001. ActivMedia Robotics, LLC, v7b.
40. Saffiotti, A., Autonomous robot navigation, in: E. Ruspini, P. Bonissone, W. Pedrycz (Eds.), Handbook of Fuzzy Computation, Oxford University Press, Oxford, 1998 (Chapter G6.1).
41. Saffiotti, A., Fuzzy Logic in Autonomous Navigation, In Driankov, D. and Saffiotti, A. eds., Fuzzy Logic Techniques for Autonomous Vehicle Navigation, Physica-Verlag Heidelberg, New York, pp.3-24, 2001
42. Saphira operations and programming manual V6.2, ActivMedia Robotics, 1999.
43. Tardos, J.D., Neria, J., Newman, P.M. and Leonard J.J., Robust Mapping and Localization in Indoor Environments using sonar data. The International Journal of Robotics Research Vol. 21, No. 4, pp.311-330, 2002.
44. Thrun, S., (1998) Learning Metric-topological Maps for Indoor Mobile Robot Navigation, Artificial Intelligence 99 21-71.
45. Tunstel E. and Jamshidi, M., (1997) Intelligent Control and Evolution of Mobile Robot Behavior, Ed. M. Jamshidi, Prentice Hall Inc. 1997, Chapter 1 pp1-24.
46. Tunstel E., H. Danny, Y. Lippincott and M. Jamshidi (1997) Adaptive Fuzzy-Behavior Hierarchy for Autonomous Navigation, Proceeding of the 1997 IEEE International Conference on Robotics and Automation Albuquerque, New Mexico – April 1997, pp829-834.
47. Wang, L.X., A Course in Fuzzy Systems and Control. Prentice-Hall International, Inc., 1997.
48. Xu, W.L., Tso, S.K. and Fung, Y.H., Fuzzy reactive control of a mobile robot incorporating a real/virtual target switching strategy, Robotics and Autonomous Systems 23 (1998) 171–186.
49. Xu, W.L. and Tso, S.K., Sensor-Based Fuzzy reactive navigation of a mobile robot through local target switching, IEEE Transactions on System, Man and Cybernetics-Part C: Application and Reviews, Vol. 29, No. 3, pp. 451-459 (1999).
50. Yamauchi, B., A frontier-based approach for autonomous exploration. In Proceedings of 1997 IEEE International Symposium on Computational Intelligence in Robotics and Autonmation, 1997.
51. Zhang, M., Peng, S. and Meng, Q., Neural network and fuzzy logic techniques based collision avoidance for a mobile robot, Robotica 15 (1997) 627–632.

24

Motion Planning and Reconfiguration for Systems of Multiple Objects

Adrian Dumitrescu[1]
University of Wisconsin–Milwaukee
U.S.A.

1. Abstract

This chapter surveys some recent results on motion planning and reconfiguration for systems of multiple objects and for modular systems with applications in robotics.

1 Introduction

In this survey we discuss three related reconfiguration problems for systems of multiple objects.

(I) Consider a set of n pairwise disjoint objects in the plane that need to be brought from a given start (initial) configuration S into a desired goal (target) configuration T. The *motion planning* problem for such a system is that of computing a sequence of object motions (schedule) that achieves this task. If such a sequence of motions exists, we say that the problem is *feasible* and say that it is *infeasible* otherwise.

The problem is a simplified version of a multi-robot motion planning problem [23], in which a system of robots, whose footprints are, say disks, are operating in a common workplace. They have to move from their initial positions to a set of specified target positions. No obstacles other than the robots themselves are assumed to be present in the workplace; in particular, the workspace is assumed to extend throughout the entire plane. In many applications, the robots are indistinguishable so any of them can occupy any of the specified target positions; that is, the disks are unlabeled. Another application which permits the same abstraction is moving around large sets of heavy objects in a warehouse. Typically, one is interested in minimizing the number of moves and designing efficient algorithms for carrying out the motion plan. There are several types of moves, such as sliding or lifting, which lead to different models that will be discussed in Section 2.

(II) A different kind of reconfiguration problem appears in connection to so-called *self-reconfigurable* or *metamorphic* modular systems. A *modular* robotic system consists of a number of identical robotic modules that can connect to, disconnect from, and relocate relative to adjacent modules, see examples in [15, 11, 26, 27, 28, 32, 34, 35, 36, 39]. Typically, the modules have a regular symmetry so that they can be packed densely, with small gaps between them. Various practical realizations are under way at different sites. Such a system can be viewed as a large swarm of physically connected robotic modules that behave collectively as a single entity.

---

[1] Computer Science, University of Wisconsin–Milwaukee, Milwaukee, WI 53201-0784, USA. Email: ad@cs.uwm.edu. Supported in part by NSF CAREER grant CCF-0444188.

The system changes its overall shape and functionality by reconfiguring into different formations. In most cases individual modules are not capable of moving by themselves; however, the entire system may be able to move to a new position when its members repeatedly change their positions relative to their neighbors, by rotating or sliding around other modules [10, 26, 38], or by expansion and contraction [32]. In this way the entire system, by changing its aggregate geometric structure, may acquire new functionalities to accomplish a given task or to interact with the environment.
Shape changing in these composite systems is envisioned as a means to accomplish various tasks, such as reconnaissance, exploration, satellite recovery, or operation in constrained environments inaccessible to humans,(e.g., nuclear reactors, space or deep water). For another example, a self-reconfigurable robot can aggregate as a snake to traverse a tunnel and then reconfigure as a six-legged spider to move over uneven terrain. A novel useful application is to realize self-repair: a self-reconfigurable robot carrying some additional modules may abandon the failed modules and replace them with spare units [32]. It is usually assumed that the modules must remain connected all (or most) of the time during reconfiguration.
The motion planning problem for such a system is that of computing a sequence of module motions that brings the system in a given initial configuration I into a desired goal configuration G. Reconfiguration for modular systems acting in a grid-like environment, and where moves must maintain connectivity of the whole system has been studied in [18, 19, 20], focusing on two basic capabilities of such systems: reconfiguration and locomotion. We present details in Section 3.
(III) In many cases, the problem of bringing a set of pairwise disjoint objects (in the plane or in the space) to a desired goal configuration, admits the following abstraction: we have an underlying finite or infinite connected graph, the start configuration is represented by a set of n chips at n start vertices and the target configuration by another set of n target vertices. A vertex can be both a start and target position. The case when the chips are labeled or unlabeled give two different variants of the problem. In one move a chip can follow an arbitrary path in the graph and end up at another vertex, provided the path (including the end vertex) is free of other chips [13].
The motion planning problem for such a system is that computing a sequence of chip motions that brings the chips from their initial positions to their target positions. Again, the problem may be feasible or infeasible. We address multiple aspects of this variant in Section 4. We note that the three (disk) models mentioned earlier do not fall in the above graph reconfiguration framework, because a disk may partially overlap several target positions.

2 Models of reconfiguration for systems of objects in the plane

There are several types of moves that make sense to study, as dictated by specific applications, such as:

1. *Sliding model*: one move is sliding a disk to another location in the plane without intersecting any other disk, where the disk center moves along an arbitrary (open) continuous curve [5].
2. *Lifting model*: one move is lifting a disk and placing it back in the plane anywhere in the free space, that is, at a position where it does not intersect (the interior of) any other disk [6].
3. *Translation model*: one move is translating a disk in the plane along a fixed direction without intersecting any other disk [1].

It turns out that moving a set of objects from one place to another is related to certain *separability* problems [14, 9, 21, 22]; see also [31]. For instance, given a set of disjoint polygons in the plane, may each be moved "to infinity" in a continuous motion in the plane without colliding with the others? Often constraints are imposed on the types of motions allowed, e.g., only translations, or only translations in a fixed set of directions. Usually only one object is permitted to move at a time. Without the convexity assumption on the objects, it is easy to show examples when the objects are interlocked and could only be moved "together" in the plane; however they could be easily separated using the third dimension, i.e., in the lifting model.

Model	Type	Lower bound	Upper bound
Translating disks	Congruent	$\lfloor 8n/5 \rfloor$	$2n-1$
	Arbitrary	$2n$	$2n$
Sliding disks	Congruent	$16n/15 - o(n)$	$3n/2 + o(n)$
	Arbitrary	$2n - o(n)$	$2n-1$
Lifting disks	Congruent	$n + \Omega(n^{1/2})$	$n + O(n^{2/3})$
	Arbitrary	$\lfloor 5n/3 \rfloor$	$9n/5$
Sliding chips (grid)	Unlabeled	n	n
	Labeled	$3n/2$	$7n/4$
Lifting chips (grid)	Unlabeled	n	n
	Labeled	$3n/2$	$3n/2$

Table 1: Comparison summary: number of moves for disks in the plane/ chips in the grid.

It can be shown that for the class of disks, the reconfiguration problem in each of these models is always feasible [1, 5, 6, 9, 21, 22]. This follows essentially from the feasibility in the sliding model and the translation model, see Section 2.1. For the more general class of convex objects, one needs to allow rotations. For simplicity we will restrict ourselves mostly to the case of disks. We are thus lead to the following generic question: Given a pair of start and target configurations, each consisting of n pairwise disjoint disks in the plane, what is the minimum number of moves that suffice for transforming the start configuration into the target configuration for each of these models?

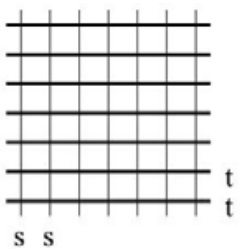


Fig. 1. $2n$ – 1 moves are necessary (in either model) to bring the n segments from vertical position to horizontal position.

If no target disk coincides with a start disk, so each disk must move at least once, obviously at least n moves are required. In general one can use (a variant of) the following simple *universal* algorithm for the reconfiguration of n objects using $2n$ moves. To be specific,

consider the lifting model. In the first step (n moves), move all the objects away anywhere in the free space. In the second step (n moves), bring the objects "back" to target positions. For the class of segments (or rectangles) as objects, it is easy to construct examples that require $2n - 1$ moves for reconfiguration, in any of the three models, even for congruent segments, as shown in Figure 1. A first goal is to estimate more precisely where in the interval $[n, 2n]$ the answer lies for each of these models. The best current lower and upper bounds on the number of moves necessary in the three models mentioned can be found in Table 1. It is quite interesting to compare the bounds on the number of moves for the three models, translation, sliding and lifting, with those for the graph variants discussed in Section 4. Table 1 which is constructed on the basis of the results in [1, 5, 6, 13] facilitates this comparison.

Some remarks are in order. Clearly, any lower bound (on the number of moves) for lifting is also valid for sliding, and any upper bound (on the number of moves) for sliding is also valid for lifting. Another observation is that for lifting, those objects whose target position coincides with their start position can be safely ignored, while for sliding this is not true. A simple example appears in Figure 2: assume that we have a large disk surrounded by $n - 1$ smaller ones. The large disk has to be moved to another location, while the $n - 1$ smaller disks have to stay where they are. One move is enough in the lifting model, while $n - 1$ are needed in the sliding model: one needs to make space for the large disk to move out by moving out about half of the small disks and then moving them back in to the same positions.

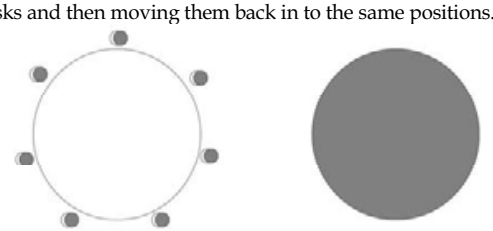


Fig. 2. One move is enough in the lifting model, while $n - 1$ are needed in the sliding model for this pair of start and target configurations with n disks each (here $n = 1$). Start disks are white and target disks are shaded.

A move is a *target move* if it moves a disk to a final target position. Otherwise, it is a *non-target* move. Our lower bounds use the following argument: if no target disk coincides with a start disk (so each disk must move), a schedule which requires x non-target moves, must consist of at least $n + x$ moves.

2.1 The sliding model

It is not difficult to show that, for the class of disks, the reconfiguration problem in the sliding model is always feasible. More generally, the problem remains feasible for the class of all convex objects using sliding moves which follows from Theorem 1 below. This old result appears in the work of Fejes Tóth and Heppes [21], but it can be traced back to de Bruijn [9]; some algorithmic aspects of the problem have been addressed more recently by Guibas and Yao [22].

Theorem 1 [9, 21, 22] *Any set of* n *convex objects in the plane can be separated via translations all parallel to any given fixed direction, with each object moving once only. If the topmost and bottommost points of each object are given (or can be computed in O(n log n) time), an ordering of*

the moves can be computed in O(n log n)time.

The universal algorithm mentioned earlier can be adapted to perform the reconfiguration of any set of n convex objects. It performs $2n$ moves for the reconfiguration of n disks. In the first step (n moves), in decreasing order of the x-coordinates of their centers, slide the disks initially along a horizontal direction, one by one to the far right. Note that no collisions can occur. In the second step (n moves), bring the disks "back" to target positions in increasing order of the x-coordinates of their centers. (General convex objects need rotations and translations in the second step). Already for the class of disks, Theorem 3 shows that one cannot do much better in terms of the number of moves.

Theorem 2 [5] *Given a pair of start and target configurations S and T, each consisting of n congruent disks,* $\frac{3n}{2}+O(\sqrt{n\log n})$ *sliding moves always suffice for transforming the start configuration into the target configuration. The entire motion can be computed in* $O(n^{3/2}(\log n)^{-1/2})$ *time. On the other hand, there exist pairs of configurations that require* $\left(1+\frac{1}{15}\right)n-O(\sqrt{n})$ *moves for this task.*

We now briefly sketch the upper bound proof and the corresponding algorithm in [5] for congruent disks. First, one shows the existence of a line bisecting the set of centers of the start disks such that the strip of width 6 around this line contains a small number of disks. More precisely the following holds:

Lemma 1 [5] *Let S be a set of n pairwise disjoint unit (radius) disks in the plane. Then there exists a line* ℓ *that bisects the centers of the disks such that the parallel strip of width 6 around* ℓ *(that is,* ℓ *runs in the middle of this strip) contains entirely at most* $O(\sqrt{n\log n})$ *disks.*

Let S′ and T′ be the centers of the start disks and target disks, respectively, and let ℓ be the line guaranteed by Lemma 1. We can assume that ℓ is vertical. Denote by $s_1 = \lceil n/2 \rceil$ and $s_2 = \lfloor n/2 \rfloor$ the number of centers of start disks to the left and to the right of ℓ (centers on ℓ can be assigned to the left or right). Let $m = O(\sqrt{n\log n})$ be the number of start disks contained entirely in the vertical strip of width 6 around ℓ. Denote by t_1 and t_2 the number of centers of target disks to the left and to the right of ℓ, respectively. By symmetry we can assume that $t_1 \le n/2 \le t_2$

Let R be a region containing all start and target disks, e.g., the smallest axis-aligned rectangle that contains all disks. The algorithm has three steps. All moves in the region R are taken along horizontal lines, i.e., perpendicularly to the line ℓ. The reconfiguration procedure is schematically shown in Figure 3. This illustration ignores the disks/targets in the center parallel strip.

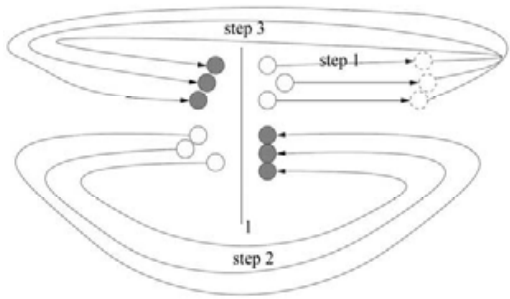


Fig. 3. Algorithm with three steps for sliding congruent disks. The start disks are white and the target disks are shaded.

STEP 1 Slide to the far right all start disks whose centers are to the right of ℓ and the (other)

start disks in the strip, one by one, in decreasing order of their *x*-coordinates (with ties broken arbitrarily). At this point all $t_2 \geq n/2$ target disk positions whose centers lie right of are free.
STEP 2 Using all the $s_1' \leq s_1 \leq n/2$ remaining disks whose centers are to the left of ℓ, in increasing order of their *x*-coordinates, fill free target positions whose centers are right of ℓ, in increasing order of their *x*-coordinates: each disk slides first to the left, then to the right on a wide arc and to the left again in the end. Note that $s_1' \leq n/2 \leq t_2$. Now all the target positions whose centers lie left of ℓ are free.
STEP 3 Move to place the far away disks: first continue to fill target positions whose centers are to the right of ℓ, in increasing order of their *x*-coordinates. When done, fill target positions whose centers are left of ℓ, in decreasing order of their *x*-coordinates. Note that at this point all target positions whose centers lie left of ℓ are free.
The only non-target moves are those done in STEP 1 and their number is $n/2 + O(\sqrt{n \log n})$, so the total number of moves is $3n/2 + O(\sqrt{n \log n})$.
The first simple idea in constructing a lower bound is as follows: The target configuration consists of a set of *n* densely packed unit (radius) disks contained, for example, in a square of side length $\approx 2\sqrt{n}$. The disks in the start configuration enclose the target positions using concentric rings, that is, $\Theta(\sqrt{n})$ rings, each with $\Theta(\sqrt{n})$ start disk positions, as shown in Figure 4. Now observe that for each ring, the first move which involves a disk in that ring must be a non-target move. The number of rings is $\Theta(\sqrt{n})$, from which the lower bound $n + \Omega(\sqrt{n})$ follows.

Fig. 4. A lower bound of $n+\Omega(\sqrt{n})$ moves: $\Theta(\sqrt{n})$ rings, each with $\Theta(\sqrt{n})$ start disk positions. Targets are densely packed in a square formation enclosed by the rings.

This basic idea of a cage-like construction can be further refined by redesigning the cage[5]. The new design is more complicated and uses "rigidity" considerations which go back to the stable disk packings of density 0 of K. Böröczky[7]. A packing *C* of unit (radius) disks in the plane is said to be *stable* if each disk is kept fixed by its neighbors [8]. More precisely, *C* is stable if none of its elements can be translated by any small distance in any direction without colliding with the others. It is easy to see that any stable system of (unit) disks in the plane must have infinitely many elements. Somewhat surprisingly, K. Böröczky [7] showed that there exist stable systems of unit disks with arbitrarily small density. These can be adapted for the purpose of constructing a lower bound in the sliding model for congruent disks. The details are quite technical, and we only sketch here the new cage-like constructions shown in Figure 5.
Let us refer to the disks in the start (resp. target) configuration as white (resp. black) disks.

Now fix a large n, and take n Use $O(\sqrt{n})$ of them to build three junctions connected by three "double-bridges" to enclose a triangular region that can accommodate n tightly packed nonoverlapping black disks. Divide the remaining white disks into three roughly equal groups, each of size $\frac{n}{3} - O(\sqrt{n})$ and rearrange each group to form the initial section of a "one-way bridge" attached to the unused sides of the junctions. Each of these bridges consists of five rows of disks of "length" roughly $\frac{n}{15}$ where the length of a bridge is the number of disks along its side. The design of both the junctions and the bridges prohibits any target move before one moves out a sequence of about $\frac{1}{5} \cdot \frac{n}{3} = \frac{n}{15}$ white adjacent disks starting at the far end of one of the one-way bridges. The reason is that with the exception of the at most 3 × 4 = 12 white disks at the far ends of the truncated one-way bridges, every white disk is fixed by its neighbours. The total number of necessary moves is at least $\left(1 + \frac{1}{15}\right) n - O(\sqrt{n})$ for this triangular ring construction, and at least $\left(1 + \frac{1}{30}\right) n - O(\sqrt{n})$ for the hexagonal ring construction.
For disks of arbitrary radii, the following result is obtained in [5]:

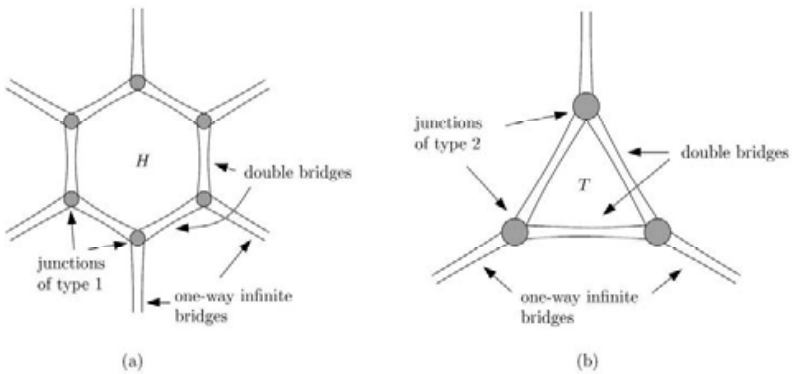


Fig. 5. Two start configurations based on hexagonal and triangular cage-like constructions. Targets are densely packed in a square formation enclosed by the cage.
For disks of arbitrary radii, the following result is obtained in [5]:

Theorem 3 *[5] Given a pair of start and target configurations, each consisting of ¯disks of arbitrary radii, 2n sliding moves always suffice for transforming the start configuration into the target configuration. The entire motion can be computed in O(n* log *n) time. On the other hand, there exist pairs of configurations that require 2n – o(n)moves for this task, for every sufficiently large n.*

Fig. 6. A simple configuration which requires about $3n/2$ moves (basic step for the recursive construction).

The upper bound follows from the universal reconfiguration algorithm described earlier. The lower bound is a recursive construction shown in Figure 7. It is obtained by iterating recursively the basic construction in Figure 6, which requires about $3n/2$ moves: note that the target positions of the n – 1 small disks lie inside the start position of the large disk. This means that no small disk can reach its target before the large disk moves away, that is, before roughly half of the n – 1 small disks move away. So about $3n/2$ moves in total are necessary.

In the recursive construction, the small disks around a large one are replaced by the "same" construction scaled (see Figure 7). All disks have distinct radii so it is convenient to think of them as being labeled. There is one large disk labeled 0, and $2m + 1$ groups of smaller disks around it close to the vertices of a regular $(2m + 1)$-gon ($m \geq 1$). The small disks on the last level or recursion have targets inside the big ones they surround (the other disks have targets somewhere else). Let $m \geq 1$ be fixed. If there are levels in the recursion, about $n/2 + n/4 + \cdots + n/2^k$ non-target moves are necessary. The precise calculation for $m = 1$ yields the lower bound $2n - O(n^{\log_3 2}) = 2n - O(n^{0.631})$, see [5].

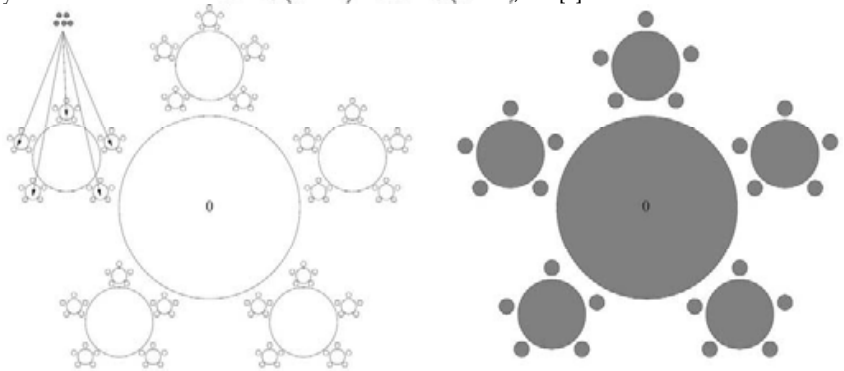


Fig. 7. Recursive lower bound construction for sliding disks of arbitrary radii: $m = 2$ and $k = 3$.

2.2 The translation model

This model is a constrained variant of the sliding model, in which each move is a translation along a fixed direction; that is, the center of the moving disk traces a line segment. With some care, one can modify the universal algorithm mentioned in the introduction, and find a suitable order in which disks can be moved "to infinity" and then moved "back" to target position via translations all almost parallel to any given fixed direction using $2n$ translation moves [1].

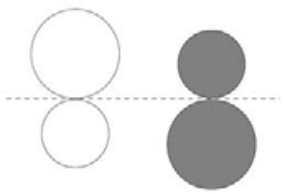


Fig. 8. A two-disk configuration that requires 4 translation moves.

That this bound is best possible for arbitrary radii disks can be easily seen in Figure 8. The two start disks and the two target positions are tangent to the same line. Note that the first move cannot be a target move. Assume that the larger disk moves first, and observe that its location must be above the horizontal line. If the second move is again a non-target move, we have accounted for 4 moves already. Otherwise, no matter which disk moves to its target position, the other disk will require two more moves to reach its target. The situation when the smaller disk moves first is analogous. One can repeat this basic configuration with two disks, using different radii, to obtain configurations with an arbitrary large (even) number of disks, which require $2n$ translation moves.

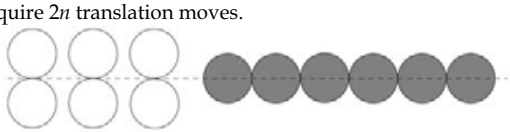


Fig. 9. A configuration of 6 congruent disks that requires 9 translation moves.

Theorem 4 [1] *Given a pair of start and target configurations, each consisting of ¯disks of arbitrary radii,* $2n$ *translation moves always suffice for transforming the start configuration into the target configuration. On the other hand, there exist pairs of configurations that require* $2n$ *such moves.*
For congruent disks, the configuration shown in Figure 9 requires $3n/2$ moves, since from each pair of tangent disks, the first move must be a non-target move. The best known lower bound, $[8n/5]$, is from [1]; we illustrate it in Figure 10. The construction is symmetric with respect to the middle horizontal line. Here we have groups of five disks each, where to move one group to some five target positions requires eight translation moves. In each group, the disks S_2, S_4 and S_5 are pairwise tangent, and S_1 and S_3 are each tangent to S_2; the tangency lines in the latter pairs are almost horizontal converging to the middle horizontal line. There are essentially only two different ways for "moving" one group, each requiring three non-target moves: (i) S_1 and S_3 move out, S_2 moves to destination, S_4 moves out, S_5 moves to destinations followed by the rest. (ii) S_4, S_5 and S_2 move out (to the left), S_1 and S_3 move to destinations followed by the rest.

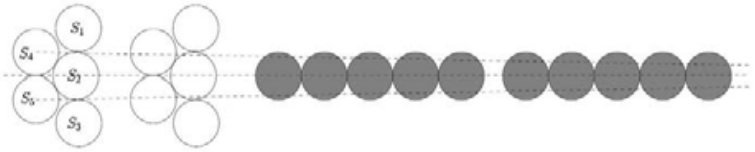


Fig. 10. Reconfiguration of a system of 10 congruent disks which needs 16 translation moves. Start disks are white and target disks are shaded.
Theorem 5 [1] *Given a pair of start and target configurations, each consisting of n congruent disks,* $2n - 1$ *translation moves always suffice for transforming the start configuration into the target configuration. On the other hand, there exist pairs of configurations that require* $[8n/5]$ *such moves.*

2.3 The lifting model

For systems of n congruent disks, one can estimate the number of moves that are always sufficient with relatively higher accuracy. The following result is obtained in [6]:
Theorem 6 [6] *Given a pair of start and target configurations S and T, each with n congruent disks, one can move disks from S to T using* $n + O(n^{2/3})$ *moves in the lifting model. The entire motion can be computed in* $O(n \log n)$ *time. On the other hand, for each n, there exist pairs of configurations*

which require $n + \Omega(n^{1/2})$ *moves for this task.*

The lower bound construction is illustrated in Figure 11 for n = 25. Assume for simplicity that $n = m^2$ where m is odd. We place the disks of T onto a grid X x X of size m x m where X = {2, 4, …, 2m} We place the disks of S onto a grid of size (m – 1) x (m – 1) so that they overlap with the disks from T. The grid of target disks contains 4m – 4 disks on its boundary. We "block" them with 2m – 2 start disks in S by placing them so that each start disk overlaps with two boundary target disks as shown in the figure. We place the last start disk somewhere else, and we have accounted for $(m-1)^2 + (2m-2) + 1 = m^2$ start disks. As proved in [6], at least $n + [m/2]$ moves are necessary for reconfiguration (it can be verified that this number of lifting moves suffices for this construction).

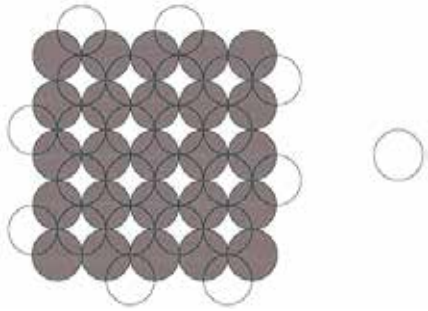


Fig. 11. A pair of start and target configurations, each with n = 25 congruent disks, which require 27 lifting moves. The start disks are white and the target disks are shaded.

The upper bound $n + O(n^{2/3})$ of is technically somewhat more complicated. It uses a binary space partition of the plane into convex polygonal (bounded or unbounded) regions satisfying certain properties. Once the partition is obtained, a *shifting* algorithm moves disks from some regions to fill the target positions in other regions, see [6] for details. Since the disks whose target position coincides with their start position can be safely ignored in the beginning, the upper bound yields an efficient algorithm which performs a number of moves close to the optimum (for large n).

For arbitrary radius disks, the following result is obtained in [6]:

Theorem 7 [6] *Given a pair of start and target configurations S and T, each with n disks with arbitrary radii, 9n/5 moves always suffice for transforming the start configuration into the target configuration. On the other hand, for each n, there exist pairs of configurations which require [5n/3]moves for this task.*

The lower bound is very simple. We use disks of different radii (although the radii can be chosen very close to the same value if desired). Since all disks have distinct radii, one can think of the disks as being labeled. Consider the set of three disks, labeled 1, 2, and 3 in Figure 12. The two start and target disks labeled i are congruent, for i= 1,2,3. To transform the start configuration into the target configuration takes at least two non-target moves, thus five moves in total. By repeatedly using groups of three (with different radii), one gets a lower bound of 5n/3 moves, when n is a multiple of three, and [5n/3] in general.

We now explain the approach in [6] for the upper bound for ¯disks of arbitrary radii. Let S = {S_1, …, S_n} and T = {t_1, …, t_n} be the start and target configurations. We assume that for each i, disk S_i is congruent to disk t_i i.e., t_i is the target position of S_i; if the correspondence

$s_i \to t_i$ is not given (but only the two sets of disks), it can be easily computed by sorting both *S* and *T* by radius.

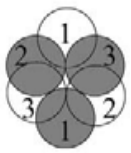


Fig. 12. A group of three disks which require five moves to reach their targets; part of the lower bound construction for lifting disks of arbitrary radii. The disks are white and their targets are shades.

In a directed graph $D = (V,E)$, let $d_v = d_v^+ + d_v^-$ denote the degree of vertex v, where d_v^+ is the out-degree of v and d_v^- is the in-degree of v. Let $\beta(D)$ be the maximum size of a subset V' of V such that $D[V']$, the subgraph induced by V', is acyclic. In [6] the following inequality is proved for any directed graph:

$$\beta(D) \geq \max\left(\sum_{v\in V}\frac{1}{d_v^+ + 1}, \sum_{v\in V}\frac{1}{d_v^- + 1}\right).$$

For a disk ω, let $\mathring{\omega}$ denote the interior of ω. Let S be a set of k pairwise disjoint red disks, and T be a set of l pairwise disjoint blue disks. Consider the bipartite red-blue disk intersection graph $G = (S,T,E)$, where $E = \{(s,t) : s \in S, t \in T, \mathring{s} \cap \mathring{t} \neq \emptyset\}$. Using the triangle inequality (among sides and diagonals in a convex quadrilateral), one can easily show that any red-blue disk intersection graph $G = (S,T,E)$ is planar, and consequently $|E| \leq 2(|S| + |T|) - 4 = 4n - 4$. We think of the start and target disks being labeled from 1 to n, so that the target of start disk i is target disk i. Consider the directed *blocking graph* $D = (S,F)$ on the set S of n start disks, where

$$F = \{(s_i, s_j) : i \neq j \text{ and } \mathring{s_i} \cap \mathring{t_j} \neq \emptyset\}.$$

If $(s_i, s_j) \in F$ we say that disk i *blocks* disk j. (Note that $s_i \cap t_i \neq \emptyset$ does not generate any edge in D.) Since if $(s_i, s_j) \in F$ then $(s_i, t_j) \in E$, we have $|F| \leq |E| \leq 4n - 4$. The algorithm first eliminates all the directed cycles from D using some non-target moves, and then fills the remaining targets using only target moves. Let

$$d^+ = \frac{\sum_{v\in S} d_v^+}{n} = \frac{|F|}{n}$$

be the average out-degree in D. We have $d^+ \leq 4$, which further implies (by Jensen's inequality):

$$\beta(D) \geq \sum_{v\in S}\frac{1}{d_v^+ + 1} \geq \frac{n}{d^+ + 1} \geq \frac{n}{5}.$$

Let $S' \subset S$ be a set of disks of size at least $n/5$ and whose induced subgraph is acyclic in D. Move out far away the remaining set S'' of at most $4n/5$ disks, and note that the far away disks do not block any of the disks in S'. Perform a topological sort on the acyclic graph $D[S']$ induced by S', and fill the targets of these disks in that order using only target moves. Then fill the targets with the far away disks in any order. The number of moves is at most

$n + 4n/5 = 9n/5$, as claimed. Figure 13 shows the bipartite intersection graph *G* and the directed blocking graph *D* for a small example, with the corresponding reconfiguration procedure explained above. Similar to the case of congruent disks, the resulting algorithm performs a number of moves that is not more than a constant times the optimum (with ratio 9/5).

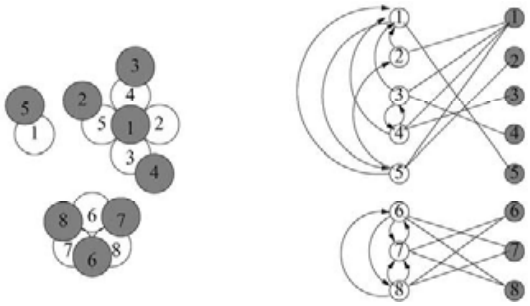


Fig. 13. The bipartite intersection graph *G* and the directed blocking graph *D*. Move out: 4, 5, 7, 8; no cycles remain in *D*. Fill targets: 3,2,1,6, and then 4,5,7,8. The start disks are white and the target disks are shaded.

2.4 Further questions

Here are some interesting remaining questions pertaining to the sliding and translation models:

(1) Consider the reconfiguration problem for congruent squares (with arbitrary orientation) in the sliding model. It can be checked that the $3n/2 + o(n)$ upper bound for congruent disks still holds in this case, however the $16n/15 - o(n)$ lower bound based on stable disk packings cannot be used. Observe that the $n + \Omega(n^{1/2})$ lower bound for congruent disks in the lifting model (Figure 11) can be realized with congruent (even axis-aligned) squares, and therefore holds for congruent squares in the sliding model as well. Can one deduce better bounds for this variant?

(2) Derive a $(2 - \delta)n$ upper bound for the case of congruent disks in the translation model (where δ is a positive constant), or improve the $\lfloor 8n/5 \rfloor$ lower bound.

(3) Consider the reconfiguration problem for congruent *labeled* disks in the sliding model. It is easy to see that $\lfloor 5n/3 \rfloor$ the lower bound for arbitrary disks holds, since the construction in Figure 12 can be realized with congruent disks. Find a $(2 - \delta)n$ upper bound (where δ is a positive constant), or improve the $\lfloor 5n/3 \rfloor$ lower bound.

3. Modular and reconfigurable systems

In [20] a number of issues related to motion planning and analysis of rectangular metamorphic robotic systems are addressed. A distributed algorithm for reconfiguration

that applies to a relatively large subclass of configurations, called horizontally convex configurations is presented. Several fundamental questions in the analysis of metamorphic systems were also addressed. In particular the following two questions have been shown to be decidable: (*i*) whether a given set of motion rules maintains connectivity; (*ii*) whether a goal configuration is reachable from a given initial configuration (at specified locations).

For illustration, we present the *rectangular model* of metamorphic systems introduced in [18, 19, 20]. Consider a plane that is partitioned into a integer grid of square cells indexed by their center coordinates in the underlying *x-y* coordinate system. This partition of the plane is only a useful abstraction: the module-size determines the grid size in practice, and similarly for orientation.

At any time each cell may be empty or occupied by a module. The *reconfiguration* of a metamorphic system consisting of *n* modules is a sequence of configurations (distributions) of the modules in the grid at discrete time steps t=0,1,2…, see below. Let V_t be the configuration of the modules at time *t*, where we often identify V_t with the set of cells occupied by the modules or with the set of their centers. We are only interested in configurations that are connected, i.e., for each *t*, the graph $G_t = (V_t, E_t)$ must be connected, where for any t, E_t, is the set of edges connecting pairs of cells in V_t that are side-adjacent. The following two generic motion rules (Figure 14) define the *rectangular model* [18, 19, 20]. These are to be understood as possible in all axis parallel orientations, in fact generating 16 rules, eight for rotation and eight for sliding. A somewhat similar model is presented in [10].

- *Rotation*: A module *m* side-adjacent to a stationary module *f* rotates through an angle of 90° around *f* either clockwise or counterclockwise. Figure 14(a) shows a clockwise *NE* rotation. For rotation to take place, both the target cell and the cell at the corresponding corner of *f* that *m* passes through (*NW* in the given example) have to be empty.
- *Sliding*: Let f_1 and f_2 be stationary cells that are side-adjacent. A module that *m* is side-adjacent to f_1 and adjacent to f_2 slides along the sides of f_1 and f_2 into the cell that is adjacent to *f* and side-adjacent to f_2. Figure 14(b) shows a sliding move in the *E* direction. For sliding to take place, the target cell has to be empty.

The system may execute moves sequentially, when only one module moves at each discrete time step, or concurrently (when more modules can move at each discrete time step). Parallel execution has the advantage to speed up the reconfiguration process. If concurrent moves are allowed, additional conditions have to be imposed, as in [19, 20]. In order to ensure motion precision, each move is guided by one or two modules that are stationary during the same step.

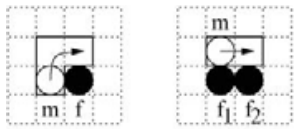


Fig. 14. Moves in the rectangular model: (a) clockwise *NE* rotation and (b) sliding in the *E* direction. Fixed modules are shaded. The cells in which the moves take place are outlined in the figure.

The following recent result settles a conjecture formulated in [20].

Theorem 8 *[18] The set of motion rules of the rectangular model guarantees the feasibility of motion planning for any pair of connected configurations V and V' having the same number of modules. That is, following the above rules, V and V' can always be transformed into each other so that all intermediate configurations are connected.*

The algorithm is far from being intuitive or straightforward. The main difficulties that have to be overcome are: dealing with holes and avoiding certain deadlock situations during reconfiguration. The proof of correctness of the algorithm and the analysis of the number of moves (cubic in the number of modules, for sequential execution) are quite involved. At the moment, this is for reconfiguration without the presence of obstacles!

We refer to a set of modules that form a straight line chain in the grid, as a *straight chain*. It is easy to construct examples so that neither sliding nor rotation alone can reconfigure them to straight chains. Conform with Theorem 8, the motion rules of the rectangular model (rotation and sliding, Figure 14) are sufficient to guarantee reachability, while maintaining the system connected at each discrete time step. This has been proved earlier for the special class of horizontally convex systems [20].

A somewhat different model can be obtained if, instead of the connectedness requirement at each time step, one imposes the so-called *single backbone* condition [20]: a module moves (slides or rotates) along a single connected backbone (formed by the other modules). If concurrent moves are allowed, additional conditions have to be imposed, as in [20]. A subtle difference exists between requiring the configuration to be connected at each discrete time step and requiring the existence of a connected backbone along which a module slides or rotates. A one step motion that does not satisfy the single backbone condition appears in Figure 15: the initial connected configuration practically disconnects during the move and reconnects at the end of it. Our algorithm has the nice property that the single backbone condition is satisfied during the whole procedure.

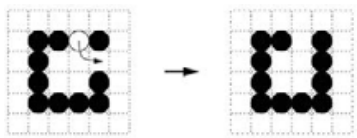
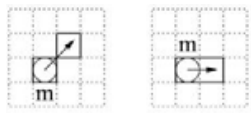


Fig. 15. A rotation move which temporarily disconnects the configuration.

We now briefly discuss another rectangular model for which the same property holds. The following two generic motion rules (Figure 16) define the *weak rectangular model*. These are to be understood as possible in all axis-parallel orientations, in fact generating eight rules, four diagonal moves and four side moves (axis-parallel ones). The only imposed condition is that the configuration must remain connected at each discrete time step.

- *Diagonal move*: A module *m* moves diagonally to an empty cell corner-adjacent to *cell*(*m*).
- *Side move*: A module *m* moves to an empty cell side-adjacent to *cell*(*m*).



(a) (b)

Fig. 16. Moves in the weak rectangular model: (a) *NE* diagonal move and (b) side move in the *E* direction. The cells in which the moves take place are outlined in the figure.

The same result as in Theorem 8 holds for this second model [18]; however, its proof and corresponding reconfiguration algorithm, are much simpler. It remains to be seen how these two models compare in a practical realization.

Theorem 9 [18] *The set of motion rules of the weak rectangular model guarantees the feasibility of motion planning for any pair of connected configurations having the same number of modules.*

A different variant of inter-robot reconfiguration is useful in applications for which there is no clear preference between the use of a single large robot versus a group of smaller ones [12]. This leads to the merging of individual smaller robots into a larger one or the splitting of a large robot into smaller ones. For example, in a surveillance or rescue mission, a large robot is required to travel to a designated location in a short time. Then the robot may create a group of small robots which are to explore in parallel a large area. Once the task is complete, the robots might merge back into the large robot that carried them.

As mentioned in [17], there is considerable research interest in the task of having one autonomous vehicle follow another, and in general in studying robots moving in formation. [19] examines the problem of dynamic self-reconfiguration of a modular robotic system to a formation aimed at reaching a specified target position as quickly as possible. A number of fast formations for both rectangular and hexagonal systems are presented, achieving a constant ratio guarantee on the time to reach a given target in the asymptotic sense.

For example in the rectangular model, for the case of even $n \geq 4$, there exist snake-like formations having a speed of $\frac{1}{3}$. Fig. 17 shows the case $n = 20$, where the formation at time 0 reappears at time 3, translated diagonally by one unit. Thus by repeatedly going through these configurations, the modules can move in the *NE* direction at a speed of $\frac{1}{3}$.

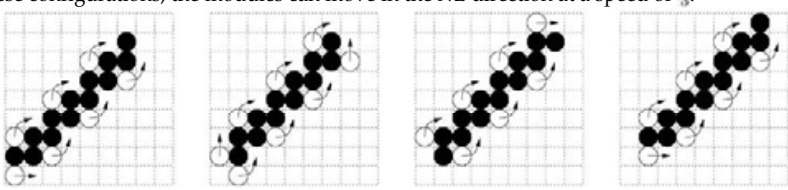


Fig. 17. Formation of 20 modules moving diagonally at a speed of $\frac{1}{3}$ (diagonal formation).

We conclude this section with some remaining questions on modular and reconfigurable systems related to the results presented:

(1) The reconfiguration algorithm in the rectangular model takes fewer than $2n^3$ moves in the worst case (with the current analysis). On the other hand, the reconfiguration of a vertical chain into a horizontal chain requires only $\Theta(n^2)$ moves, and it is believed that no other pair of configurations requires more. This has been shown to hold in the weak rectangular model, but it remains open in the first model.
(2) Extend the algorithm(s), for reconfiguration under the presence of obstacles.
(3) Study weather the analogous rules of rotation and sliding in three dimensions permit the feasibility of motion planning for any pair of connected configurations having the same number of modules.

4 Reconfigurations in graphs and grids

In certain applications, objects are indistinguishable, therefore the chips are unlabeled; for instance, a modular robotic system consists of a number of identical modules (robots), each having identical capabilities [18, 19, 20]. In other applications the chips may be labeled. The variant with unlabeled chips is easier and always feasible, while the variant with labeled chips may be infeasible: a classical example is the 15-puzzle on a 4 x 4 grid — introduced by Sam Loyd in 1878 — which admits a solution if and only if the start permutation is an even permutation [24, 33] Most of the work done so far concerns labeled versions of the reconfiguration problem, and we give here only a short account.

For the generalization of the 15-puzzle on an arbitrary graph (with $k = v - 1$ labeled chips in a connected graph on v vertices), Wilson [37] gave an efficiently checkable characterization of the solvable instances of the problem. Kornhauser et al. have extended his result to any $k \leq v - 1$ and provided bounds on the number of moves for solving any solvable instance [25].

Ratner and Warmuth have shown that finding a solution with minimum number of moves for the (N x N) extension of the 15-puzzle is intractable [30], so the reconfiguration problem in graphs with labeled chips is NP-hard.

Auletta et al. gave a linear time algorithm for the *pebble motion on a tree* [3]. This problem is the labeled variant of the same reconfiguration problem studied in [13], however each move is along one edge only.

Papadimitriou et al. studied a problem of *motion planning on a graph* in which there is a mobile robot at one of the vertices s, that has to reach to a designated vertex t using the smallest number of moves, in the presence of obstacles (pebbles) at some of the other vertices [29]. Robot and obstacle moves are done along edges, and obstacles have no destination assigned and may end up in any vertex of the geraph. The problem has been shown to be NP-complete even for planer graphs, and a ration $O(\sqrt{n})$ polynomial time approximation algorithm was given in [29].

The following results are shown in [13] for the "chips in graph" reconfiguration problem described in part (III) of Section 1.

(1) The reconfiguration problem in graphs with unlabeled chips U-GRAPH-RP is NP-hard, and even APX-hard.
(2) The reconfiguration problem in graphs with labeled chips L-GRAPH-RP is APX-hard.
(3) For the infinite planar rectangular grid, both the labeled and unlabeled variants L-GRID-RP and U-GRID-RP are NP-hard.
(4) There exists a ratio 3 approximation algorithm for the unlabeled version in graphs U-GRAPH-RP. Thereby one gets a ratio 3 approximation algorithm for the unlabeled version U-GRID-RP in the (infinite) rectangular grid.
(5) It is shown that n moves are always enough (and sometimes necessary) for the reconfiguration of n unlabeled chips in graphs. For the case of trees, a linear time algorithm which performs an optimal (minimum) number of moves is presented.
(6) It is shown that $7n/4$ moves are always enough, and $3n/2$ are sometimes necessary, for the reconfiguration of n labeled chips in the infinite planar rectangular grid (L-GRID-RP).

Next, we give some details showing that in the infinite grid, n moves always suffice for the reconfiguration of n unlabeled chips, and of course it is easy to construct tight examples. The result holds in a more general graph setting (item (5) in the above list): Let G be a

connected graph, and let V and V' two -element subsets of its vertex set $V(G)$. Imagine that we place a chip at each element of V and we want to move them into the positions of V' (V and V' may have common elements). A move is defined as shifting a chip from v_1 to v_2 ($v_1, v_2 \in V(G)$) along a path in G so that no intermediate vertices are occupied.

Theorem 10 [13] *In G one can get from any n-element initial configuration V to any n-element final configuration V′ using at most n moves, so that no chip moves twice.*

It is sufficient to prove the theorem for trees. We argue by induction on the number of chips. Take the smallest tree T containing V and V', and consider an arbitrary leaf l of T. Assume first that the leaf l belongs to V: say $l=v$. If v also belongs to V', the result trivially follows by induction, so assume that this is not the case. Choose a path P in T, connecting v to an element v' of V' such that no internal point of P belongs to V'. Apply the induction hypothesis to $V\setminus\{v\}$ and $V'\setminus\{v'\}$ to obtain a sequence of at most $n-1$ moves, and add a final (unobstructed) move from to.

The remaining case when the leaf l belongs to V' is symmetric: say $l=v'$; choose a path P in T, connecting v' to an element v of V such that no internal point of P belongs to V. Move first v to v' and append the sequence of at most $n-1$ moves obtained from the induction hypothesis applied to $V\setminus\{v\}$ and $V'\setminus\{v'\}$. This completes the proof.

Theorem 10 implies that in the infinite rectangular grid, we can get from any starting position to any ending position of the same size n in at most n moves. It is interesting to compare this to the problem of sliding congruent unlabeled disks in the plane, where one can come up with cage-like constructions that require about $\frac{16n}{15}$ moves [5], as discussed in Section 2.1. Here are some interesting remaining questions on reconfigurations in graphs and grids:

(1) Can the ratio approximation algorithm for the unlabeled version in graphs U-GRAPH-RP be improved? Is there an approximation algorithm with a better ratio for the infinite planar rectangular grid?

(2) Close or reduce the gap between the $3n/2$ lower bound and the $7n/4$ upper bound on the number of moves for the reconfiguration of n labeled chips in the infinite planar rectangular grid.

5 Conclusion

The different reconfiguration models discussed in this survey have raised new interesting mathematical questions and revealed surprising connections with other older ones. For instance the key ideas in the reconfiguration algorithm in [18] were derived from the properties of a system of *maximal cycles*, similar to those of the block decomposition of graphs [16].

The lower bound configuration with unit disks for the sliding model in [5] uses "rigidity" considerations and properties of *stable* packings of disks studied a long time ago by Böröczky [7]; in particular, he showed that there exist stable systems of unit disks with arbitrarily small density. A suitable modification of his construction yields our lower bound.

The study of the lifting model offered other interesting connections: the algorithm for unit disks given in [6] is intimately related to the notion of *center point* of a finite point set, and to the following fact derived from it: Given two sets each with n pairwise disjoint unit disks, there exists a binary space partition of the plane into polygonal regions each containing roughly the same small number ($\approx n^{2/3}$) of disks and such that the total number of disks

intersecting the boundaries of the regions is small ($\approx n^{2/3}$). The reconfiguration algorithm for disks of arbitrary radius relies on a new lower bound on the maximum order of induced *acyclic* subgraphs of a directed graph [6], similar to the bound on the *independence number* of an undirected graph given by Turán's theorem [2].

The ratio 3 approximation algorithm for the unlabeled version in graphs is obtained by applying the local ratio method of Bar-Yehuda [4] to a graph H constructed from the given graph G. Regarding the various models of reconfiguration for systems of objects in the plane, we currently have combinatorial estimates on the number of moves that are necessary in the worst case. From the practical viewpoint one would like to covert these estimates into approximation algorithms with a good ratio guarantee. As shown for the lifting model, the upper bound estimates on the number of moves give good approximation algorithms for large values of n. However further work is needed in this direction for the sliding model and the translation model in particular.

6 References

[1] M. Abellanas, S. Bereg, F. Hurtado, A. G. Olaverri, D. Rappaport, and J. Tejel, Moving coins, *Computational Geometry: Theory and Applications*, 34(1), 2006, 35–48.

[2] N. Alon and J. Spencer, *The Probabilistic Method*, second edition, Wiley, New York, 2000.

[3] V. Auletta, A. Monti, M. Parente, and P. Persiano, A linear-time algorithm for the feasibility of pebble motion in trees, *Algorithmica*, 23 (1999), 223–245.

[4] R. Bar-Yehuda, One for the price of two: a unified approach for approximating covering problems, *Algorithmica*, 27 (2000), 131–144.

[5] S. Bereg, A. Dumitrescu, and J. Pach, Sliding disks in the plane, *Proceedings of Japan Conference on Discrete and Computational Geometry 2004* (J. Akiyama, M. Kano and X. Tan, editors), LNCS vol. 3742, Springer, 2005, 37–47.

[6] S. Bereg and A. Dumitrescu, The lifting model for reconfiguration, *Discrete & Computational Geometry*, accepted. A preliminary version in *Proceedings of the 21st Annual Symposium on Computational Geometry*, (SOCG '05), Pisa, Italy, June 2005, 55–62.

[7] K. Böröczky, Über stabile Kreis-und Kugelsysteme (in German), *Ann. Univ. Sci. Budapest. Eös otvösect. Math.* 7 (1964), 79–82.

[8] P. Braß, W. Moser, and J. Pach, *Research Problems in Discrete Geometry*, Springer, New York, 2005.

[9] N. G. de Bruijn, Aufgaben 17 and 18 (in Dutch), Nieuw Archief voor Wiskunde 2 (1954), 67.

[10] Z. Butler, K. Kotay, D. Rus and K. Tomita, Generic decentralized control for a class of selfreconfigurable robots, *Proceedings of the 2002 IEEE International Conference on Robotics and Automation* (ICRA '02), Washington, May 2002, 809–816.

[11] A. Casal and M. Yim, Self-reconfiguration planning for a class of modular robots, *Proceedings of the Symposium on Intelligent Systems and Advanced Manufacturing*, (SPIE'99), 246–256.

[12] A. Castano and P. Will, A polymorphic robot team, in *Robot Teams: From diversity to Polymorphism*, edited by T. Balch and L.E. Parker, pp. 139–160, A K Peters, 2002.

[13] G. C˘alinescu, A. Dumitrescu, and J. Pach, Reconfigurations in graphs and grids, *Proceedings of Latin American Theoretical Informatics Symposium* (LATIN '06),

Valdivia, Chile, March 2006, LNCS vol. 3887, Springer, 2006, 262–273.

[14] B. Chazelle, T. A. Ottmann, E. Soisalon-Soininen, and D. Wood, The complexity and decidability of separation, *Proceedings of the 11th International Colloquium on Automata, Languages and Programming*, (ICALP '84), LNCS vol. 172, Springer, 1984, pp. 119–127.

[15] G. Chirikjian, A. Pamecha and I. Ebert-Uphoff, Evaluating efficiency of self-reconfiguration in a class of modular robots, *Journal of Robotic Systems*, 13(5) (1996), 317–338.

[16] R. Diestel, *Graph theory (2nd ed.), Graduate Texts in Mathematics* 173, Springer-Verlag, New York, 2000.

[17] G. Dudek, M. Jenkin, and E. Milios, A taxonomy of multirobot systems, in *Robot Teams: From diversity to Polymorphism*, edited by T. Balch and L.E. Parker, pp. 3–22, A K Peters, 2002.

[18] A. Dumitrescu and J. Pach, Pushing squares around, *Graphs and Combinatorics*, 22(1) (2006), 37–50.

[19] A. Dumitrescu, I. Suzuki, and M. Yamashita, Motion planning for metamorphic systems: feasibility, decidability and distributed reconfiguration, *IEEE Transactions on Robotics and Automation*, 20(3), 2004, 409–418.

[20] A. Dumitrescu, I. Suzuki, and M. Yamashita, Formations for fast locomotion of metamorphic robotic systems, *International Journal of Robotics Research*, 23(6), 2004, 583–593.

[21] L. Fejes Tóth and A. Heppes, Uber stabile K ¨orpersysteme (in German), *Compositio Mathematica*, 15 (1963), 119–126.

[22] L. Guibas and F. F. Yao, On translating a set of rectangles, in *Computational Geometry*, F. Preparata (ed.), pp. 61–67, Vol. 1 of *Advances in Computing Research*, JAI Press, London, 1983.

[23] Y. Hwuang and N. Ahuja, Gross motion planning — a survey, *ACM Computing Surveys*, 25(3) (1992), 219–291.

[24] W. W. Johnson, Notes on the ¯puzzle. I., *American Journal of Mathematics*, 2 (1879), 397–399.

[25] D. Kornhauser, G. Miller, and P. Spirakis, Coordinating pebble motion on graphs, the diameter of permutation groups, and applications, *Proceedings of the 25-th Symposium on Foundations of Computer Science*, (FOCS '84), 241–250.

[26] S. Murata, H. Kurokawa and S. Kokaji, Self-assembling machine, *Proceedings of IEEE International Conference on Robotics and Automation*, (1994), 441–448.

[27] A. Nguyen, L. J. Guibas and M. Yim, Controlled module density helps reconfiguration planning, *Proceedings of IEEE International Workshop on Algorithmic Foundations of Robotics*, 2000.

[28] A. Pamecha, I. Ebert-Uphoff and G. Chirikjian, Useful metrics for modular robot motion planning, *IEEE Transactions on Robotics and Automation*, 13(4) (1997), 531–545.

[29] C. Papadimitriou, P. Raghavan, M. Sudan, and H. Tamaki, Motion planning on a graph, *Proceedings of the 35-th Symposium on Foundations of Computer Science*, (FOCS '94), 511–520.

[30] D. Ratner and M. Warmuth, Finding a shortest solution for the (N x N) extension of the 15-puzzle is intractable, *Journal of Symbolic Computation*, 10 (1990), 111–137.

[31] J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, second

edition, 1998.

[32] D. Rus and M. Vona, Crystalline robots: self-reconfiguration with compressible unit modules, *Autonomous Robots*, 10 (2001), 107–124.

[33] W. E. Story, Notes on the 15 puzzle. II., *American Journal of Mathematics*, 2 (1879), 399–404.

[34] J.E. Walter, J.L. Welch and N.M. Amato, Distributed reconfiguration of metamorphic robot chains, *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing* (PODC'00), July 2000, 171–180.

[35] J.E. Walter, J.L. Welch and N.M. Amato, Reconfiguration of hexagonal metamorphic robots in two dimensions, in *Sensor Fusion and Decentralized Control in Robotic Systems III* (G. T. McKee, P. S. Schenker, editors), *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, Vol. 4196, 2000, 441–453.

[36] X. Wei, W. Shi-gang, J. Huai-wei, Z. Zhi-zhou, and H. Lin, Strategies and methods of constructing self-organizing metamorphic robots, in *Mobile Robots: New Research*, edited by J. X. Liu, pp. 1–38, Nova Science, 2005.

[37] R. M. Wilson, Graph puzzles, homotopy, and the alternating group, *Journal of Combinatorial Theory, Series B*, 16 (1974), 86–96.

[38] M. Yim, Y. Zhang, J. Lamping and E. Mao, Distributed control for 3D metamorphosis, *Autonomous Robots*, 10 (2001), 41–56.

[39] E. Yoshida, S. Murata, A. Kamimura, K. Tomita, H. Kurokawa and S. Kokaji, A motion planning method for a self-reconfigurable modular robot, in *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

25

Symbolic Trajectory Description in Mobile Robotics

Pradel Gilbert[1], Căleanu Cătălin-Daniel[2]
[(1)]*IBISC laboratory,*
University of Evry Val d'Essonne
40 Rue du Pelvoux, 91020 Evry cedex, France
[(2)]*Electronics and Telecommunication Faculty*
Politehnica University of Timisoara
Bd. Vasile Parvan, 2, 300223 Timisoara, Romania

1. Introduction

One main issue for mobile robots is their capacity to go from one point to another autonomously, without getting lost or crashing into another object (Arkin, 1998).
It is based on three concepts:

- planning which computes a trajectory between the two points,
- navigation which gives motion orders to the robot to follow the computed trajectory,
- environment representation which permits the robot to know if it goes in the right direction.

Works presented here are interested in point 3, that is, in acquiring spatial models of the robot's physical environment.
Two different approaches to this problem have emerged. The first one, the metric/quantitative representation of the environment, has some disadvantages. For example, due to incorrigible wheel slippage, dead-reckoning could be unreliable. The non-metric/qualitative approach use perceptual landmarks to generate maps and to localise the robot with respect to these landmarks. Works presented here are interested in the non-metric approach, trying to perform a qualitative description of a stuctured indoor environment.
These problems are tackled by the Simultaneous Localisation And Mapping (SLAM) introduced by Leonard and Durrant-Whyte (Leonard & Durrant-Whyte, 1991) (Smith & Leonard, 1997) in robotics. SLAM is still today a very active field of research (Meyer & Filliat, 2002)(Filliat & Meyer, 2002a,b). This problem is regarded as one of most significant for a true autonomy of the robots. Crucial questions still remain satisfactorily unanswered in spite of great progress in this field and the existence of robust methods to map static, very structured and limited sized environments.
(Kulic and Vukic, 2003) use a robot motion planning based on behavioural cloning. In a first phase, the robot is trained under operator's control to locate unmoving obstacles avoidance through a simulator. In that phase, the evaluated variables are stored in a log file. The

second phase, called learning phase, machine learning program generates the differential equations defining the operator's trajectory, i.e. the clone. Finally, the verifying phase, the robot is controlled by the clone. These developmental phases are repeated changing both problem domain representation and learning system according to the cloning system criterion.

The problem of mapping can be generally regarded as the fact of giving to an autonomous robot the capacity to move in an environment. Thus, the problem of mapping goes further than simple construction of a plan gathering the obstacles in a given zone. Thrun (Thrun, 2002) gives a general survey of the mapping problem. He points out the six key aspects of the mapping problem:

- The effects of the noise in the measurements (Wheel slippage, localisation error introduced by integration of data from wheel encoders, drift of inertial systems are three examples among many others.),
- The high dimensionality of the entities that are being mapped (How many parameters describe the environment, its major topological elements like corridors, crossings, doors, rooms, etc.?),
- The correspondence problem, also known as the data association problem (Do the measurements made by the sensors at different points in time in the environment correspond to the same object?),
- The perceptual aliasing (Two different places from the environment can be perceived in an identical way by the sensors.),
- The environment changes over time,
- The robotic exploration, that is the task of generating robot motion in the pursuit of building a map.

This chapter is organised as follow. Section 2 gives an overview of the works in the field of environment representation. Section 3 briefly presents the test-bed perception system. Sections 4 and 5 detail our approach in the digitised construction of the environment from the distance measurements, the extraction of the landmarks and explain the fresco construction and its validation. In section 6, we propose a method to represent robot's trajectory based on series of landmarks called frescoes and different methods to select the most salient of them with which it is possible to describe the environment. Section 7 shows and discusses the experimental results. We conclude with ways to improve the method.

2. Related works

Related works can be found in the fields of Image Based Navigation systems, shape understanding using sensor data, vision based homing. Vision for mobile robot navigation did have specific development during the last twenty years. (DeSouza & Kak, 2002) give a complete survey of the different approaches. For indoor navigation, systems are classified in three groups: map-based navigation using predefined geometric and/or topological models, map-building-based navigation constructing by themselves geometric and/or topological models, and mapless navigation using only object recognition and actions associated to these objects (Gaussier & al. 1997).

Kuipers' works (Kuipers & Byan, 1991) defined symbols as distinct places situated at equal distances from the nearby obstacles. Connections between these places link symbols and represent free path (Choset & Nagatani, 2001). Fig. 1 shows an example of the Voronoii

graph of an environment. The labelled vertices represent the symbols while edges connecting the symbols are the path the robot can use.

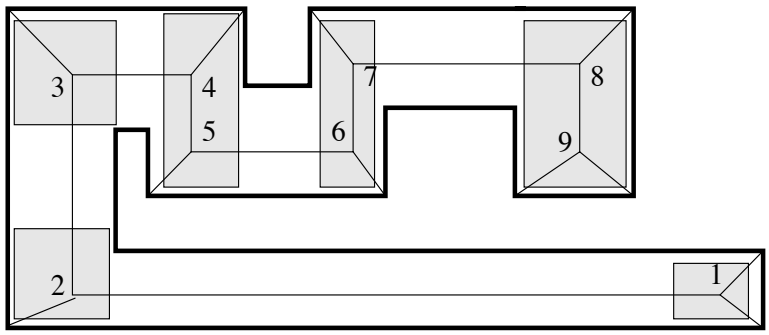


Fig. 1. Voronoii diagram with labelled vertices.

Indeed, assume that the robot has to move in this simple environment (Fig. 1) according to a mission given by the user, if the robot goes from label 1 to label 9, the most important areas are those filled, where the robot changes its direction. Between them, when there are no changes in the environment, it is useless to preserve the whole set of information from this part of the way. On the contrary, it is necessary to find a method of mapping of the filled zones which can describe them unambiguously.

In Image Based Navigation systems, several great classes of systems can be identified from the literature. The first one uses conventional telemeters and vision to find and identify objects in the environment (Wichert, 1996). The second one is the class of the systems coupling more or less directly sensor data to motor control thanks to a supervised learning process. Among them neural networks systems used as classifiers are noticeable. These systems begin to classify the environment into global classes such as "corridor, corner, room, crossing ..." (Al Alan & al. ,1995) (Pomerleau, 1993) are often followed by a second processing unit that outputs a navigation command. In addition to restrictions related to the supervised learning, these classes give only a global description and are of least interest in cluttered and complex environments. The third class includes the systems which compare current sensor data and predefined models both at a low level (edges, planes ...) – see (Kim & Neviata, 1994) - and at a high level (door, room, object ...). These systems use mainly vision sensors (cameras) that provide a huge amount of data that must be reduced to be processed in real time. The elements extracted from the data are compared to reference models known a priori. The fourth class evoked here includes the systems trying to geometrically build environment models before deciding an optimised path plan (Crosnier, 1999).

In the field of shape understanding using sensor data, environment interpretation stresses the use of natural landmarks to ease the navigation and the pose estimation of a mobile robot. Among other works, one can pinpoint (Simhon & Dudek, 1998a) which is interested in defining islands of reliability for exploration. He proposes strategies to couple navigation and sensing algorithms through hybrid topological metric maps. (Oore & al., 1997) consider the problem of locating a robot in an initially unfamiliar environment from visual input. In the same way, (MacKenzie & Dudek, 1994) involve a methodology to bind raw noisy sensor data to a map of object models and an abstract map made of discrete places of interest.

Several implementations of vision based homing systems are presented in (Franz & al., 1997). A method aiming at highlighting salient features as, for example, landmarks between these two views and deriving a decision is used in (Hong, 1991). In these works, a homing system extracts landmarks from the view and allows a robot to move to home location using sequence target locations situated en route between its current location and home. Other works are biologically inspired. (Judd & Collett, 1998) showed that ants store series of snapshots at different distances from their goal to use them for navigating during subsequent journeys. Judd and Collett experimented their theory with a mobile robot navigating through a corridor, homing successive target locations. Weber (Weber & al., 1999) proposes an approach using the bearings of the features extracted of the panoramic view leading to a robust homing algorithm. This algorithm pairs two landmarks situated into two snapshots to derive the homing direction. The bearings-pairing process uses a list of preferences similar to neighbourhood rules.

Symbolic processing methods are described in Tedder's works (Tedder & Hall, 2001). This formal approach is often called structural or syntactic description and recognition. The general method for perception and interpretation proposes to symbolically represent and manipulate data in a mapping process. (Tedder & Hall, 2001) solve the problem in modelling the 3D environment as symbolic data and in processing all data input on this symbolic level. The results of obstacle detection and avoidance experiments demonstrate that the robot can successfully navigate the obstacle course using symbolic processing control. These works use a laser range finder. A way for defining suitable landmarks from an environment as the robot travels is a research problem pointed out by Fleisher and al. in (Fleisher and al., 2003). An automatic landmark selection algorithm chooses as landmarks any places where a trained sensory anticipation model makes poor predictions. The landmark detection system consists of a sensory anticipation network and a method of detecting when the difference between the prediction of the next sensor values and the current measured values can reveal the presence of a landmark. This model has been applied to the navigation of a mobile robot. An evaluation has been made according to how well landmarks align between different runs on the same route. These works show that the robot is able to navigate reliably using only odometry and landmark category information.

In (Lamon & al., 2001), a method is proposed for creating unique identifiers called fingerprint sequences for visually distinct significant features in panoramic images. This localisation system proves that the actual position of a robot in an environment can be recovered by constructing a fingerprint sequence and comparing it with a database of known fingerprints.

The proposed work goes on the way proposed by (Tedder & Hall, 2001) and (Lamon & al., 2001). According to these works, our contribution applies mainly on a method to extract clues of interest among raw distance data delivered by a 2D panoramic laser range finder installed on the robot. These clues of interest, i.e. the landmarks, are gathered in a sequence that we call a fresco. We consider that the trajectory of the robot can be described by the set of the frescoes. To do that, we have to select the frescoes that bring new information. The originality of this work stays in the simple but efficient criteria used for the construction and the validation of the fresco but mainly to select the most pertinent frescoes along the route of the robot. In addition to this qualitative approach, one must consider that the system will have to be embarked on a vehicle, which vibrates, runs at variable speeds on a non-uniform ground. This leads to constraints of speed, size, robustness, compactness and cost, implying

various choices both at the design and at the development levels of the system. The methods used have been chosen as simple as possible to reduce the cost and the complexity of the processing. Nevertheless the method must be robust compared with the robot movements, the sensor accuracy and the variations of the complexity of the environment.

3. Test-bed perception system

The application field of this work is a middle-cost mobile robot sent in an apartment to do service for a user. Hence, the environment is of a structured and not engineered indoor type environment. At this point, the problem is two-fold. Firstly, through the Human-Machine Interface (HMI), a mission must be entered and its development must be explained to the user. Secondly, the robot has to be programmed to execute the mission. Building a description of the route as close as a human could do has at least two advantages. This description, on one hand, is requested by the HMI and, on the other hand, at the execution level, it can be a way to take into account the stumbling blocks highlighted by the conventional navigation systems.
The size of the non holonomous robot is (width x length) 0.50m x 0.75m. Its linear and angular speeds are up to 1 ms^{-1} and 2.45 $rads^{-1}$. Placed at the geometrical centre of the robot with practical/maximum ranges equal to 3m/10m, a panoramic 2D telemeter captures a circular environment. It has been decided to consider a 36m^2 squared environment to ease the reconstruction process (measurements at the corners are valid according to the maximum range of the telemeter). Only 256 measurements over the 1024 the telemeter is able to deliver are used by the fresco construction process. At a 1ms^{-1} speed, the translation displacement error remains lower than 10cm for one complete rotation of the telemeter. In 100 ms, the rotation of the robot remains lower than 23°. Experiments in the following have been made with measurements coming from both a simulated laser range finder and the real telemeter.
We will then consider that:

- there is a lack of accuracy of the telemetry measurements due to the vibrations caused by the jolts,
- most part of the environment is composed of co-operative targets (low material absorption coefficient, acceptable level of the reflected signal up to a 80° incident angle),
- reference position of the laser coincides with the main axis of the robot,
- data sequencing compensates the effects of the clockwise (CW) or counter clockwise (CCW) rotations of the robot so that the 256 horizontal distance information are regularly arranged on 360°,
- precision is greater than 20 cm for every measurements.

According to these considerations, we chose to digitise the environment on a 32 x 32 cells grid which covers the area seen by the telemeter, each cell representing a 0.1875m x 0.1875m square. The terms "grid" or "cellular space" will be considered as equivalent in the following.

4. Representation construction

4.1 Cyclic representation and cellular space

Landmarks such as "Opening, Closure, End_of_Closure, Angle_of_Closures" used to build the qualitative description of the environment from the measurements. According to the

sequential aspect of the data delivered by the laser range finder, the landmarks extraction order corresponds to the measurements order. The robot refers to two main axis: the "lengthwise axis" corresponds to the forward and rear directions of displacement, the "crosswise axis" is perpendicular to the lengthwise axis at the robot geometrical centre.
The fresco construction is divided into two main steps:

- The construction of the reliable digitised environment: cellular space building, signature extraction, crosswise, lengthwise and diagonal segments extraction, refining, reorientation.
- The landmarks extraction: Opening, Closure, End_of_Closure and Angle_of_Closures extraction, fresco construction, fresco validation.

The cellular space appears in fig. 4a.

4.2 Conventions used in the cellular space

The method uses evolution laws in the cellular space that act on every cells. For a cell called CELL the neighbourhood conventions use standard Von Neuman neighbourhood. For example, CELL_W, CELL_E, CELL_N, CELL_S are the names of the cells situated westbound, eastbound, northbound, southbound. We add the word Great to name the cells in the second neighbourhood layer (Great West: CELL_GW, Great East: CELL_GE …). The quadrants are numbered counter clockwise in relation to the lengthwise axis: quadrant 0 is the front right one.

4.3 Construction of the digitised description

Fig. 2 summarises the operations leading to the construction of a reliable cellular space (Pradel and al., 1994).

(a) Generation of the digitised environment: the very first operation performed consists in the lay-down of the distance measurements onto the grid to create the initial cellular spaces. They perform the same operations on the distance measurements issued from the sensor (part 1) and on the 45° shifted measurements set (part 2). On the grid, black cells represent the range finder impacts. Noise introduced in the measurements (measurements are made while the robot is moving) appears mainly on the form of cells agglomerations (fig. 4a). Agglomerations also occur when measurements belong to the border between adjacent cells. Elimination of agglomerations (fig. 4a, b) is performed keeping only the cells situated the closest to the robot for obvious safety reasons. The method adopted for this elimination uses evolution laws close to those used in cellular automata.

(b) Segmentation of the cellular space: the next operation is the extraction of the segments corresponding to the obstacles from the cellular space. Four directions are considered. In addition with the lengthwise (fig. 3a) and crosswise axis (fig. 3c), a search for the segments is made onto the two diagonals (fig. 3d, f). The extraction laws leave alive a cell owning a neighbour alive in the considered direction.

(c) Reorientation of the cellular space: as shown in fig. 4a, another origin of noise is bound to the oblique walls. These digitised oblique walls take the form of small adjacent segments with junctions without real significance. To eliminate these oblique walls and the noise they introduce we decided to use a second grid on which the measurements are laid with a 45° angular shift (Part 3). Superfluous data elimination and segmentation are also applied on this second grid.

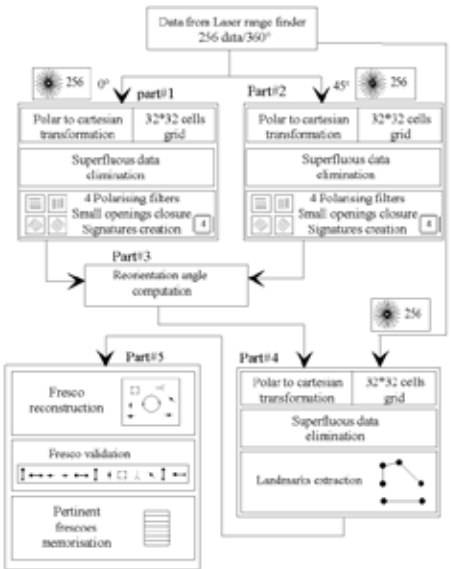


Fig. 2. lock diagram showing the operations performed in the construction of the digitised environment.

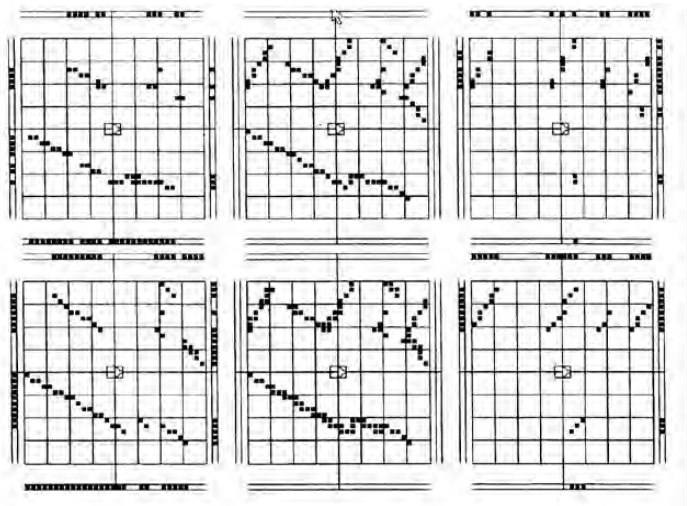


Fig. 3. Extraction of segments in the 4 filtering directions:
a (upper left): Lengthwise segmentation,
b (upper centre): Refined environment,

c (upper right): Crosswise segmentation,
d (lower left): First diagonal segmentation,
e (lower centre): Initial measurements,
f (lower right): Second diagonal segmentation

A search for the longest and the shortest continuous segments is performed (Bras & al., 1995) among the projections of the environment on the crosswise and lengthwise axis in each quadrant of the cellular space according to a filtering direction (lengthwise, crosswise, diagonal 1 and 2). A reorientation angle is then computed according to:

$$\Theta = \arctan\frac{\text{ShortestCrosswiseSegment}}{\text{LongestLengthwisesegment}} \pm n * \frac{\pi}{4} \quad (1)$$

According to the direction in which the longest segment is found (i.e. the most plausible reference in the environment), adequate choices for the sign and the value of n (n in {0, 1}) lead the robot to be reoriented parallel to the longest segment ($0 \leq \Theta \leq \pi/3$) or perpendicularly to it ($\pi/3 < \Theta \leq \pi/2$). The reoriented cellular space is re-built from the initial measurements according to the reorientation angle. Fig. 4c shows the benefits of the reorientation. The reoriented cellular space is then considered as reliable and will allow the landmarks to be extracted.

(d) Landmarks extraction: as told in the introduction, the environments are described using a fresco made of ordered series of landmarks: "Opening", "Wall" also called "Closure" and "Corner" also called "Angle_of_Closures". Let us note that an "Angle_of_Closures" must be neighboured by two "End_of_Closure" landmarks. The landmarks extraction first considers the "Opening" elements that are directly extracted from the reoriented signatures.

The "Angle_of_Closures" and "End_of_Closure" landmarks are extracted from the reoriented cellular space by the following laws. The first operation consists in the "Angle_of_Closures" extraction by the following equation that is applied to every cell in the grid:

Angle_of_Closures = ((CELL & CELL_W) | (CELL & CELL_E)) & ((CELL | CELL_N) | (CELL & CELL_S)) & neg CELLdiag

with: CELLdiag meaning that the logical state of the cell is true if it belongs to a diagonal. Operators & (logical AND) and neg (logical NOT) are applied on the states of the cells.

The first ligne of this equation checks if the cell has east or west neighbours while the second line checks north and south neighbours. Therefore a cell is considered as an Angle_of_Closures if it has at least a crosswise and a lengthwise neighbour.

The second operation aims at extracting the "Lenghtwise End_of_Closure" and "Crosswise End_of_Closure" landmarks. These operations are allowed if and only if the cell does not belong to the two diagonals and is not an "Angle_of_Closures".

Fig. 4d and 4e show the "Angle_of_Closures" and "End_of_Closure" landmarks positioned on the grids. To each landmark are associated three qualitative attributes representing three properties of landmarks. The off-sight attribute is set when the landmark stands on the cellular space border. The position attribute can take the following values: crosswise, diagonal, lengthwise according its position. The certainty attribute is introduced to take into account landmarks that could come from a possible noise introduced in the digitisation process not detected by the previous laws or a still possible bad reorientation. It is false for every landmark (for instance, diagonal "End_of_Closure", "45°_angles") whose evolution cannot be known.

5. Fresco construction

The first step of the fresco construction gathers the landmarks space into ordered series of semantic clues and describes the environment by positioning landmarks in respect to each others. Each landmark has exactly two neighbours (the last landmark in the list has the first one as second neighbour). Building the fresco is made using the symbols presented in Table 1 which gathers the landmarks identity and attributes. The landmarks identity and attributes have been chosen according to the indoor environment in which the robot moves. This operation mainly aims at eliminating the notion of distance to the profit of a spatial series and highlights the qualitative representation of the environment. An example of fresco is given in fig. 4f. The robot is situated in the middle of the environment. To each landmark are associated three qualitative attributes representing three properties of landmarks. The off-sight attribute is set when the landmark stands close to or beyond the end of the sensor range. The position attribute can take the following values: crosswise, diagonal or lengthwise according its position related to the lenghtwise and crosswise robot axis. The certainty attribute is introduced to take into account landmarks whose evolution can be forecast. It is false for every landmark (for instance, diagonal "End_of_Closure", "45°_angles") that could come from a possible noise introduced in the digitisation process and whose evolution cannot be known (Pradel & al., 2000), (Pradel & Bras, 2001).

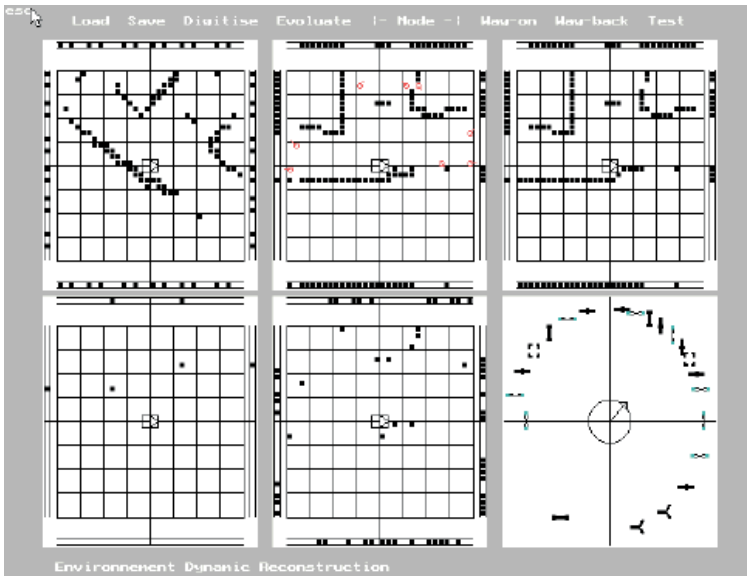


Fig. 4. Example of the digitised constructions:
a (upper left): real world from raw measurements; b (upper centre): reoriented cellular space; c (upper right): refined space after superfluous data elimination;
d (lower left): Angles_of_Closure extraction, e (lower centre): End_of_Closure extraction; f (lower right): fresco construction

Symbol	Landmark	Position	Off-sight	Certainty
	Angle_of_Closure			True
	End_of_Closure	lengthwise		True
	End_of_Closure	lengthwise	off_sight	False
	End_of_Closure	crosswise		True
	End_of_Closure	crosswise	off_sight	False
	End_of_Closure	diagonal1		False
	End_of_Closure	diagonal1	off_sight	False
	End_of_Closure	diagonal2		False
	End_of_Closure	diagonal2	off_sight	False
	45°Angle	lengthwise		False
	45°Angle	crosswise		False
	Opening	lengthwise		True
	Breakthrough	lengthwise		True
	Opening	crosswise		True
	Breakthrough	crosswise		True

Table 1. Landmarks used in the fresco construction.

The second step focuses on the fresco validation. Assuming that there is only one description for one environment, strict laws of neighbourhood are defined. Fig. 5 shows these neighbourhood laws that can be interpreted as a set of logical assertions. An Angle_of_Closure can only have as neighbours Angle_of_Closures or End_of_Closures. For each landmark, the neighbourhood is checked. Every time a fresco is built, the whole set of these rules is applied in order to validate the fresco. If one rule failed, the fresco is not valid.

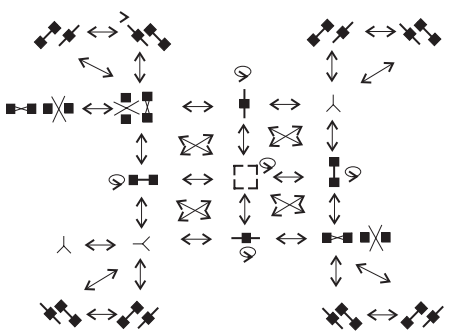


Fig. 5. Landmarks neighbourhood rules.

The validation fails mainly due to a bad landmark extraction process in a very noisy cellular space or a bad reorientation. Making the necessary corrections in the extraction laws to solve these seldom failing cases leads to an increasing of the complexity of the evolution laws, increasing not really justified by the low frequency of the failures. We consider that the loss of a fresco is not an important drawback: a failure in the validation of the fresco will

be corrected by the next valid one with only slight effects on the mission of the robot and the effect of this loss is very attenuated because the process of transitions detection and environment memorisation eliminates a greater part of the frescoes. When it is validated, the fresco appears as shown in fig. 4f. A fresco will contain at most 64 landmarks symbols organised into 4 sectors of 16 symbols at most.

6. Symbolic trajectory description using frescoes

Building the symbolic description of the route followed by the robot is three-fold:

- how to build the qualitative descriptions (frescoes) in accordance with the robot's sensors ?
- how to describe the route by a sequence of the most pertinent frescoes ?
- how to use these frescoes with the control-command level of the robot ?

This section deals with the second point. The choice of the most salient frescoes is made using different criteria described in the following sections. Every time the laser range finder scans the environment, a fresco is built. In our case, the fresco built-in period is 300ms. Hence, if all frescoes are stored their number grows quickly and some of them are not useful. Storing all the frescoes when the robot runs in a corridor is a trivial example. All frescoes are very similar excepted at both ends. If only few frescoes are useful, how then is it possible to select them? Is a specific sequence of frescoes able to describe a part of the environment? Answering, at least partially, to these questions is the aim of this section.

Following a specific path, the total number of stacked frescoes could be large enough. Moreover, successive frescoes could be identical or slightly different. Therefore, a selection of meaningful frescoes, which offers a thoroughly environment description, is absolutely necessary. Based on these salient selected frescoes, the robot also should be able to find a return path. In local homing for example, an agent returns to a previously visited location by moving to maximize the correspondence between what it sees currently and a remembered view from the target.

In dealing with frescoes, which are basically a collection of symbolic strings, we were inspired by different methods, such as those used in spell checking, optical character recognition (OCR), molecular biology for DNA or amino-acid sequences study (Altschul, 1991), (Karlin, 1990) or computational intelligence.

The first two criteria proposed to evaluate a kind of distance between frescoes are called resemblance and barycentre. A new fresco is considered as bringing new information if its distance to the previous stored one regarding one of the criteria is greater than a threshold. The two next sections describe these criteria. A systematic study gives an evaluation of the thresholds to use to make the criteria effective.

6.1 Resemblance method

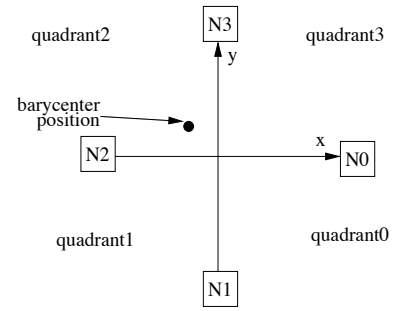
This criterion uses a nearby principle of that presented in (Hong, 1991). A correlation function allows calculating the resemblance between two frescoes. This criterion has been tested in the same environment as that used for the construction and the validation of the frescoes. The use of this criterion shows that the landmarks that are not certain make very difficult the evaluation of the resemblance so only the certain elements were kept. The resemblance between two consecutive frescoes is calculated by taking into account the difference between the number of certain landmarks in the corresponding quadrants. The resemblance between two frescoes is calculated from the difference between the number of

landmarks in respective quadrants of two consecutive frescoes. The comparison of this difference with a reference threshold indicates if the current fresco should be kept or rejected because not bringing enough information.
The resemblance between two consecutive frescoes i and j is calculated as:

$$r_{ij} = |N0_i\text{-}N0_j| + |N1_i\text{-}N1_j| + |N2_i\text{-}N2_j| + |N3_i\text{-}N3_j| \quad (2)$$

where
Nk_i, k = 1 ... 4 represents the number of landmarks in quadrant k of the i-th fresco Nk_j, k = 1 ... 4 represents the number of landmarks in quadrant k of the j-th fresco .
If the resemblance r_{ij} is greater then an a priori specified threshold then the j-th fresco will be selected and memorized as sufficiently different from the rest.

6.2 Barycenter method



N0: number of certain landmarks in quadrant0
N1: number of certain landmarks in quadrant1
N2: number of certain landmarks in quadrant2
N3: number of certain landmarks in quadrant3

Fig. 6. Barycenter computation between certain landmarks.

This criterion is inspired by the distance of Hausdorff which measures the distance between two sets (Ahuactzin & al., 1995), (Huttenlocher & al., 1993). In our case, this notion was very simplified to respect real-time constraints. It takes into account only the number of certain landmarks in every quadrant. The landmarks are positioned as indicated on the fig. 6 and the barycentre is positioned at the following coordinates:

$$xref = \frac{N0_i - N2_i}{Ntot_i}; x = \frac{N0_j - N2_j}{Ntot_j} \quad (3)$$

$$yref = \frac{N1_i - N3_i}{Ntot_i}; y = \frac{N1_j - N3_j}{Ntot_j} \quad (4)$$

$$bary_{ij} = \sqrt{(x_{ref} - x)^2 + (y_{ref} - y)^2} \quad (5)$$

where
Nk_i, k = 1 ... 4 is the number of landmarks in quadrant k of the i-th fresco,
Nk_j, k = 1 ... 4 is the number of landmarks in quadrant k of the fresco,

$Ntot_i$ and $Ntot_j$ are the total numbers of certain landmarks in the i-th/j-th frescoes respectively.
Any variation of the number of elements in a quadrant implies a movement of the barycentre. If this displacement is greater then an a priori specified threshold then the j-th fresco will be selected and memorized.

6.3 Distances based methods

Distance is usually, but not necessarily, defined on a vector space. For strings, there are also some ways for quantifying how much two strings differs, as we will see in the next sections. These metric functions attempt to ascribe a numeric value to the degree of dissimilarity between two strings.

(a) Hamming distance method: the Hamming distance (HD) could be defined only for strings of the same length (Gusfield, 1997). For two strings, S_1 and S_2, the Hamming distance $HD(S_1, S_2)$ represents the number of places in which the two strings differ, (Lamon, 2001) have different characters as shown in the following example: HD ('ABCD', 'ACDB') = 3

(b) Levenshtein distance method: the Levenshtein distance (LD) realizes a more complex evaluation of two strings than the Hamming distance. It could operate with strings not necessary of the same length and represents the minimum number of elementary transformations (insertion, deletion and substitution of a symbol) needed to transform one string into another (Levenshtein, 1966):

$$LD(S_1, S_2) = \min(N_{ins} + N_{del} + N_{subst}) \tag{6}$$

Closely related to it is the weighted Levensthein distance (WLD) also known as edit distance, where different costs are assigned to each edit operation (Kohonen, 1988) (Wagner, 1974):

$$WLD(S_1, S_2) = \min(w_{ins}N_{ins} + w_{del}N_{del} + w_{subst}N_{subst}) \tag{7}$$

(c) N-Gram method: an N-gram is a substring of N consecutive symbols. Let N_1 and N_2 be the number of N-grams in strings S_1 and S_2, respectively let m be the number of matching N-grams. If one string is longer than the other, the unmatched N-grams are also counted as differences. The feature distance (FD) is defined then as (Kohonen, 1987):

$$FD(S_1, S_2) = \max(N_1,N_2) - m(S_1, S_2) \tag{8}$$

6.4 Similarity based methods

Finding similarities in character strings is an important problem in text processing and data mining. It has applications in genetics research as well, since strands of DNA can be expressed as very long strings of the characters.
A similarity measure is simpler than a distance. For strings S_1, S_2, finding similarities in character strings is an important problem in text processing and data mining. It has applications in genetics research as well, since strands of DNA can be expressed as very long strings of characters.
A similarity measure is simpler than a distance. For strings S_1, $S_2 \in S$, any function $s : S^2 \to \Re$ can be declared similarity. For strings, similarity is closely related to alignment.
(a)Cross correlation matching method

This function is commonly used in signal processing. For symbols, the function compares string S_1 (of length m) with S_2 (of length l = n ≥ m) and produces a cross correlation similarity vector, CCS, of length (l = m + n-1) with elements CCS_i (with i = 0, 1 … l-1) given by (Gusfield, 1997) (Haykin, 1999):

$$CCS_i(s_1, s_2) = \begin{cases} \sum_{j=0}^{i} S(s_{1j}, s_{2(n-1)-i-j}), if \;\; i = 0...m-1 \\ \sum_{j=0}^{m-1} S(s_{1j}, s_{2(n-1)-i-j}), if \;\; i = m...n-1, m \neq n \\ \sum_{j=0}^{(l-1)-i} S(s_{1(n-1)-(l-1-i)+j}, s_{2j}), if \;\; i = n...l-1 \end{cases} \quad (9)$$

where:

$$S(x, y) = \begin{cases} 1 \; if \; x = y \\ 0 \; if \; x \neq y \end{cases} \quad (10)$$

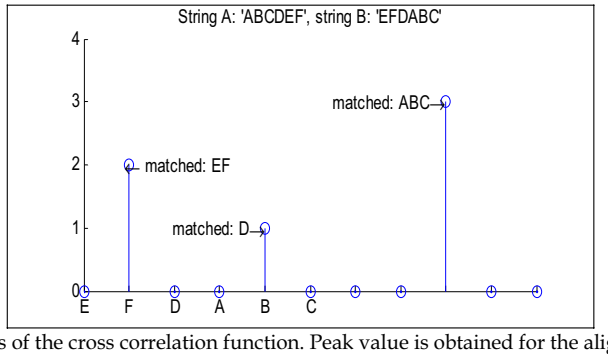


Fig. 7. Results of the cross correlation function. Peak value is obtained for the alignment of "ABC" tri-gram.

Fig. 7 gives an example of the results given by the cross correlation matching method for two strings of length equal to 6.

6.5 Neural network based method

Speaking in a neural network terminology, finding the salient frescoes is equivalent with finding prototype vectors. Self Organizing Feature Map-Neural Networks, SOFM-NN, tries to place or adapt neurons in such a way that they serve as good prototypes of input data for which they are sensitive.

(a) Classic SOFM-NN: these networks are based on unsupervised competitive learning and winner-takes-all neurons (Haykin, 1999). During the training phase a SOFM-NN creates a topologic correspondence between the spatial location of the neurons in the output layer and the intrinsic features of the input patterns. If there is some similarity between input vectors then neighbours neurons will fire. If two input patterns are different than output

neurons situated at considerable distance or spatial location will respond. For prototype vectors calculus usually Euclidian distance is used, as elements having the smallest sum of squared distance over the data set.
The principal problem is that classic SOFM-NN training algorithm is defined for numbers and not for strings. There are numerous ways for string to numbers conversion and vice versa (Aha & al., 1991) (Blanzieri, 1999). For our particular case, the maximum number of symbols within a fresco is 16, hex coded. So the NN input vector could be constructed by means of:

- Direct coding: each symbol had its own binary equivalent (0 = 0000, 1 = 00001 … F = 1111),
- Exclusive coding that is, the symbol is coded with an unary vector with all the components but the i-th set to zero (0 = 000 … 0001, 1 = 00 … 010, … , F = 10 … 000).

Finally, a fresco will be represented as a binary vector composed by concatenation of each binary coded constituent string.
(b) Symbolic SOFM-NN: based on distance measure for strings and calculating the prototype as a mean or median value, SOFM-NN for strings have been defined (Kohonen, 1998). These SOFM-NN are organized as a symbol strings array, whereby the relative locations of the strings on the SOFM ought to reflect some distance measure (e.g. LD, FD) between the strings. The idea behind the pure symbolic SOFM-NN is to define similarities or distances between data objects. In our application data objects are represented by symbolic strings. Based on these similarities/distances, finding representative prototypes (for our application, meaningful frescoes) will be the next step.
In training of pure symbolic SOFM-NN, two steps are repeated:

- Find best-matching unit (BMU) for each data item, and add the data item to the list of its best-matching unit; BMU is found using the defined similarity/distance measure,
- Update the models in SOFM nodes: find the median data item belonging to the union of the list (data list union contains all data items in the neighbourhood of the current node being update).

For computing the median data item, assume there are 3 data items (e.g. symbol strings S_1, S_2, S_3) and the following pair wise distances (Table 2):

	S_1	S_2	S_3
S_1	0	4	2
S_2	4	0	2
S_3	1	2	0

Table 2. Pair wise distances.

then compute the sum of the distances from each data item to others:

$$\begin{aligned} S_1 &: 0 + 4 + 1 = 5 \\ S_2 &: 4 + 0 + 2 = 6 \\ S_3 &: 1 + 2 + 0 = 3 \end{aligned} \tag{11}$$

The smallest sum of distances is for data item S_3, so that is the median of this data set. In the case of SOFM-NN, the distances could be weighted by the value of neighbourhood function (e.g. a Gaussian-shaped neighbourhood function).

7. Experimental results

7.1 Application of the resemblance and barycentre criteria in simple environment

The two criteria apply only on the certain landmarks and have been tested in two types of environments. In a first step, experiments in simple environments led us to point out the thresholds relevant ranges. In a second step, a complex environment has been used to validate these thresholds.

The problem is to find the right threshold for each criterion. A representative panel of situations is first established and systematic tests are made on each situation in which the frescoes are listed for different thresholds of the two criteria. Then a reference threshold for each criterion is fixed taking into account firstly the ratio of kept frescoes and secondly the position of these frescoes with respect to their situation along the robot's route in the considered environment. Finally, thresholds that have been defined are tested in a complex environment.

(a) Choice of different types of environment: indoor environments can be described using a limited number of situations (Al Alan, 1995): openings, walls, angles, room, corridor, dead-end and crossings. So far, tested situations are listed in Table 3. Fig. 8 shows the example of the "opening on the left situation". Numbers on the left of the figure show the different positions where frescoes have been constructed. In this example, frescoes are built from position 1 to position 31 (only one of five is drawn to make the figure readable).

Fig. 8. Example of situation: Opening on the left.

In the different situations, the initial numbers of frescoes are different (Table 3).

	Situation	Number of frescoes
Angle to the left	AL	31
Angle to the right	AR	31
Opening on the left	OL	31
Opening on the right	OR	31
X-crossing	CX	42

Table 3. Initial number of built frescoes.

(b) Number of pertinent frescoes vs. criterion: it is firstly interesting to observe the number of frescoes kept for different values of thresholds. For barycentre criterion, values between 0

and 2 with a step of 0.05 are tested. For resemblance criterion, values between 0 and 12 with a step of 0.5 are tested. Beyond these limits, only fresco number one is kept. As the initial number of frescoes is different in all situations, the ratio between the number of frescoes kept and the initial number of frescoes is analysed. Fig. 9 shows the results for resemblance criterion. Fig. 10 shows the results for barycentre criterion. It can be seen that curves in each figure are similar, meaning that criteria have the same response in all the environment situations. It seems then possible to find a common threshold.

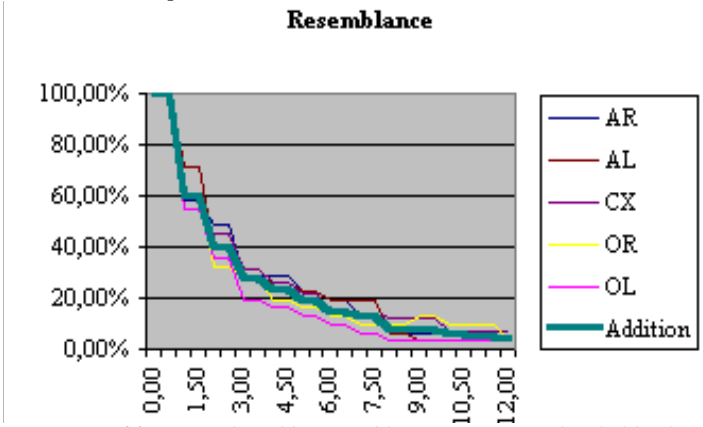


Fig. 9. Percentage of frescoes selected by resemblance criterion vs. threshold value (AR/AL: angle on the right/left, CX: X-crossing, LA: lab, OR/OL opening on the right/left, Sum: add up).

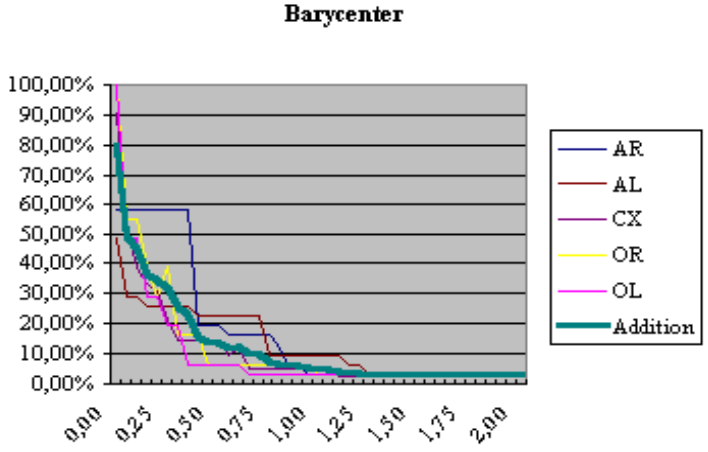


Fig. 10. Percentage of frescoes selected by barycentre criterion vs. threshold value (AR/AL: angle on the right/left,CX: X-crossing, LA: lab, OR/OL opening on the right/left, Sum: add up).

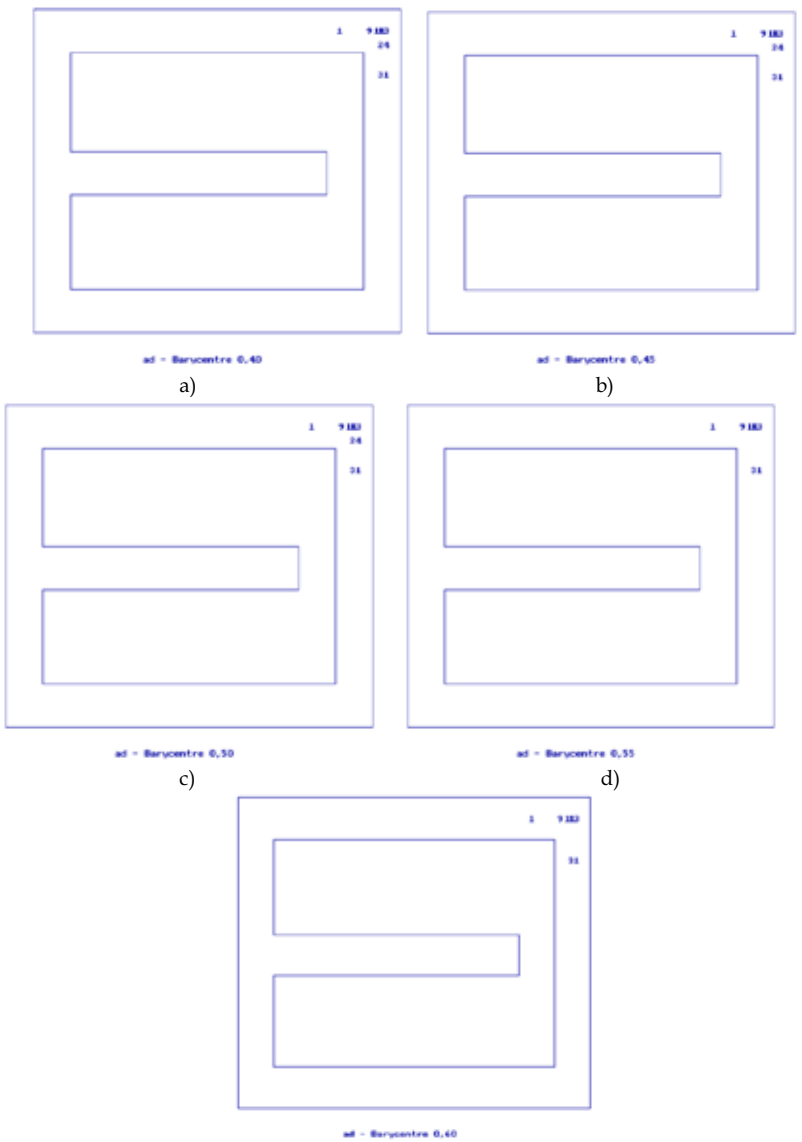


Fig. 11. Pertinent frescoes vs. barycentre criterion (AL situation); a) threshold=0.40; b) threshold=0.45; c) threshold=0.50; d) threshold=0.55; e) threshold=0.60.

It also can be noted that curves decrease quickly for low thresholds values. In fig. 8, frescoes between 1 and 10 represent the same part of the environment with very slight differences. The objective is to keep a reasonable part of frescoes between 10% and 20% in the first approximation. That means thresholds values comprise between 5 and 7 for resemblance criterion and between 0.4 and 0.6 for barycentre criterion.
(c) Positions of pertinent frescoes: for both criteria, it is interesting to visualise which frescoes are considered as pertinent (fig. 11). Frescos number 1 and 31 represent the beginning and the end of the trajectory: they appear for all thresholds. Frescoes 9, 11, 13 and 24 represent the heart of the turning. They are very close considering Euclidean distance but they differ in term of orientation. Fresco number 24 disappears for thresholds equal to 0.55 or 0.60. The value 0.50 is the central threshold value for barycentre criterion. A similar analysis has been conducted for all other situations. In the same way, the resemblance criterion leads to the same conclusion with 6.0 as central threshold.

7.2 Application of the resemblance and barycentre criteria in complex environments

A complete trajectory has been studied in a complex environment (fig. 12 a)). The two criteria have been applied. The variations of the thresholds have been limited to the range determined by the tests in simple environments: 5 to 7 for resemblance and 0.4 to 0.6 for barycentre. Fig. 13 shows the percentage of selected frescoes for both criteria. For barycentre criterion, there is no significant difference between the complex and the simple environments. For resemblance criterion, the ratio is greater in the complex environment than in the simple ones. Nevertheless, for a threshold equal to 7.0, the ratio becomes close to the ratio obtained in simple environments.

7.3 Application of the other criteria

Against the frescoes acquired (fig. 12 b)) from the lab environment (Hoppenot, 2003), the above mentioned possibilities of salient frescoes selection were implemented and compared.

a) b)

Fig. 12. a) Test environment: the lab; b) Frescoes acquired by the robot from the environment shown in the left.

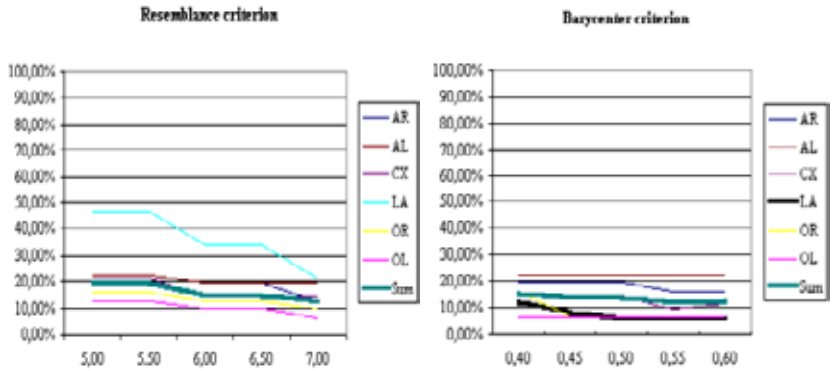
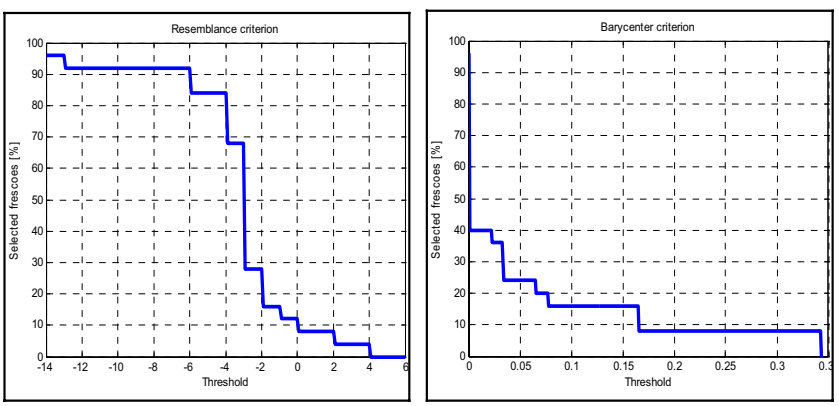


Fig. 13. Comparison of percentage of frescoes selected by resemblance/barycentre criterion in complex (LA) and simple environments vs. threshold.

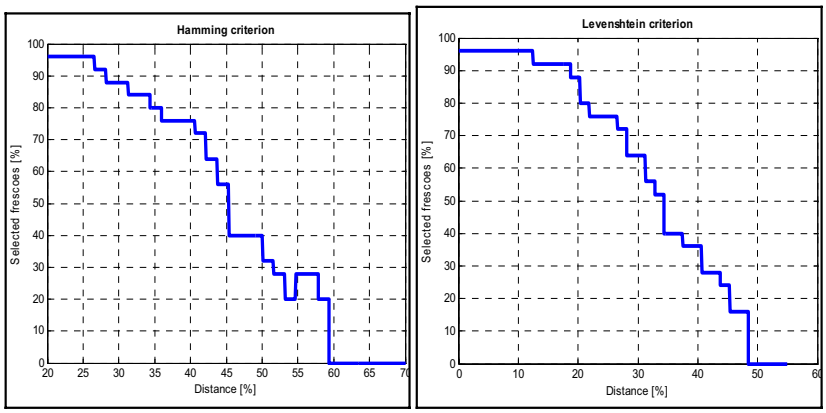
The current numbers of the selected frescoes are synthetically presented in Table 4.

Method	Index
R	2 3 8 22 23 24 25
B	4 9 11 13 21 23
H	9 10 13 15 17 18 20
L	9 11 15 17 19 22
C	9 11 15 17 18 19 25
N	1 3 7 8 13 15 17

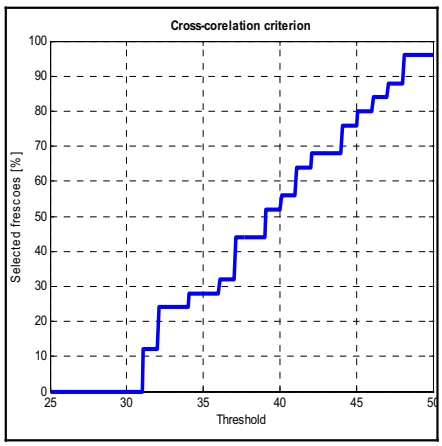
Table 4. Indexes of selected frescoes with R-Resemblance, B-Barycentre, H-Hamming, L-Levenshtein, C-Cross-correlation, N-Neural Network.



(a) (b)



(c) (d)



(e)

Fig. 14. The dependence percent of selected frescoes – threshold.
(a) Resemblance criterion;
(b) Barycentre criterion;
(c) Hamming criterion;
(d) Levenshtein criterion
(e) Cross-correlation criterion;

In fig. 14 the dependence percentage of selected frescoes vs. threshold is depicted. Fig. 15 show the salient frescoes selected by each method. An acceptable percent of the selected meaningful frescoes should be around 30% or less from the total amount of frescoes. In absolute values this mean around 7 frescoes selected.

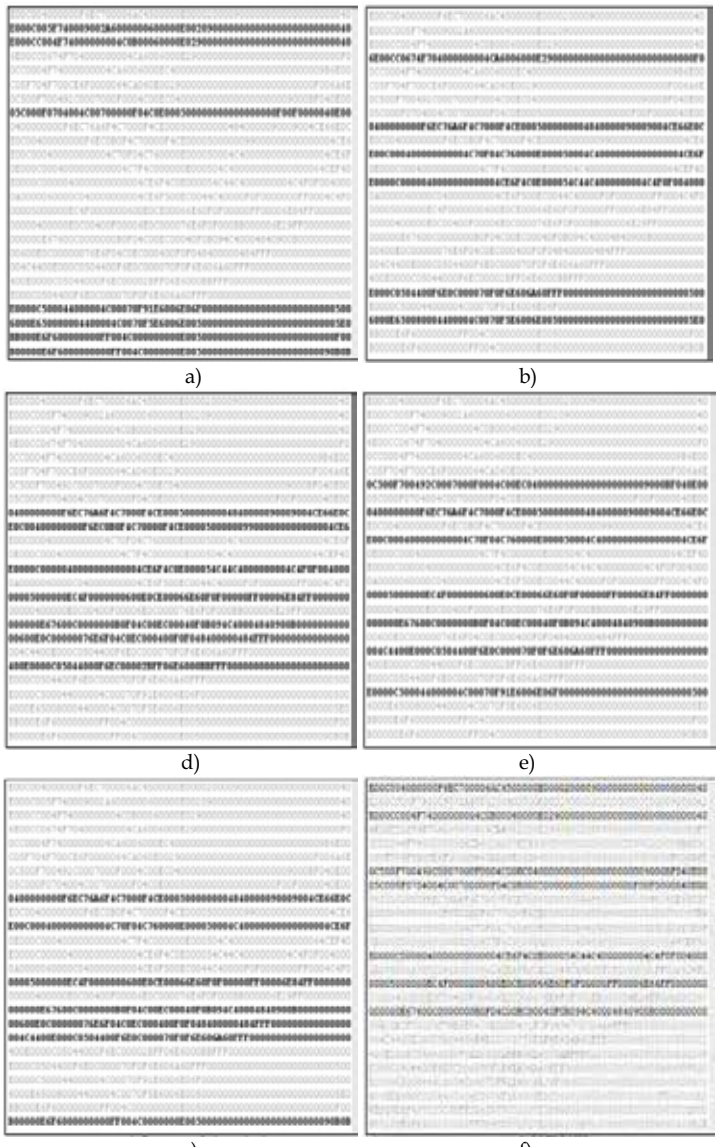


Fig. 15. Selected salient frescoes using the criteria of: a) resemblance; b) barycentre; c) Hamming; d) Levenshtein; e) cross-correlation; f) SOFM-NN.

8. Future works and conclusion

Human beings, as well as insects (Collet & al., 1992), use resemblance (or dissimilarity) to compare views of the environment rejecting those that do not bring up new elements without using metrics, only using the occurence of landmarks. In this work, we propose a qualitative method inspired of homing methods (Weber, 1999) to construct the environment surrounding an indoor mobile robot equipped with a 2D telemetry sensor. Every times distance measurements are made, landmarks are extracted and organised into series called frescoes. From this point, distance information are no longer used. In order to derive the pertinent frescoes that can describe the trajectory of the robot, we plan to use a pairing-like method. The first criterion that is primarily being investigated uses a resemblance between two frescoes. The landmarks are bounded and a correlation function measures the difference between consecutives frescoes. The second criterion is based on the difference between the barycentre positions of consecutive frescoes (Huttenlocher, 1993). Those frescoes separated by a difference higher than a threshold are considered as pertinent to describe the robot's route. In both cases the differences are compared with thresholds that are experimentally set up. Despite the criteria simplicity, the results in the very changing test environment (Fig. 12a) show that the thresholds experimentally trimmed in simple environments are well fitted to a complex environment. But the resemblance and barycentre methods have the disadvantage of not taking into account the qualitative aspect of landmarks but only the quantitative one. Lets consider an hypothetically example, in which two consecutive frescoes are completely different but has the same number of landmarks/quadrant. Both methods will give an inappropriate answer, resulting in meaningful frescoes losses, because both operate with number, not type, of landmarks. The value of the selection threshold for the resemblance method is also difficult to be anticipated because of rapid variation of the number of selected frescoes in the region of the optimal threshold. It could be easily observed the poor performance of this criterion: the marked frescoes are somehow similar and not representative. The barycentre method is similar with the previous one: in selecting the salient frescoes only number of landmarks from quadrants are counted. It differs in respect of computing the difference between strings and it seems to give slightly better results.

The Hamming distance compares two strings/frescoes character by character, the distance representing the number of different characters found. Here the selection threshold has been expressed in percentage form. The principle underlying Hamming distance is totally different from the previous two methods: it takes into account the qualitative aspect of strings and, as a consequence, is a better solution. In spite of this fact one might consider it giving unsatisfactory results. Let's take a fragment from two successive frescoes, for example: … 0004F74000 … and … 004F740000 … It is clear that these two consecutive frescoes contain basically the same landmark. The 1-character left shift is an environment perspective changing due robot movement along the trajectory. Although, HD score is very high, as the two consecutive frescoes were completely different, resulting in a possible selection of both strings.

This kind of problem is not present in the case of Levenshtein distance. LD computes the distance, in this case, as a simple insertion operation, the distance being kept at minimum. It appears that this method is the best solution for the problem of salient

frescoes selection. The computationally cost represents the main disadvantage of LD. One might observe that frescoes are padded with lots of zeros representing empty cells. In order to reduce the computation time, these empty spaces might be ignored. We called the result fast Levenshtein distance, fLD, which produce, in terms of selected frescoes, the same results as LD, but in a significantly shorter time. Almost the same results as LD are given by the cross-correlation principle. Due to alignment underling principle of these methods, the perspective modification of landmarks in a fresco is not seen as a fundamental change.

The SOFM-NN implemented has an input layer of 256 (64 symbols/fresco x 4 bits) neurons and 7 output neurons. Thus, the training is constituted of 25 binary vector having 256 elements. The network has been trained for 5000 epochs. After the learning phase, the seven weight vector corresponding to the output neurons should represent the essential frescoes selected from the input set.

Using a SOFM neural network for salient frescoes selection turns out to be improper. Among possible explanations are:

- The reduced size of training elements; The 25 considered set of frescoes are not enough to form appropriate prototype vectors. Thus, prototype vectors are not entire identically with some of the 25 training frescoes.
- There is no sufficiently redundancy in the 25 frescoes selected.
- The conversion process frescoes -> binary vectors -> real numbers and vice-versa generates errors.

Within the framework of mobile robots navigation, six methods for salient frescoes selection were described and tested. Of the six, the Levenshtein distance and cross-correlation defined for strings approaches produced the most accurate results and had some benefits in interpreting the score in meaningful ways (see Table 5). The good results given by these approaches could be explained based on theirs ability in dealing with frescoes perspective modification.

	R	**B**	**H**	**L**	**C**	**N**	**Score**
R	-	1(23)	0	1(22)	1(25)	2(3,8)	5
B	1(23)	-	2(9,13)	2(9,11)	2(9,11)	1(13)	8
H	0	2(9,13)	-	3(9,15,17)	4(9,15,17,18)	3(13,15,17)	12
L	1(22)	2(9,11)	3(9,15,17)	-	5(9,11,15,17,19)	3(7,15,17)	14
C	1(25)	2(9,11)	4(9,15,17,18)	5(9,11,15,17,19)	-	2(15,17)	14
N	2(3,8)	1(13)	3(13,15,17)	3(7,15,17)	2(15,17)	-	11

Table 5. Common selected frescoes. Based on these common frescoes a score for each method is computed.

Legend: R-Resemblance, B-Barycentre, H-Hamming, L-Levenshtein, C-Cross-correlation, N-Neural Network.

One application field is service robotics (e.g., supplying help to old or handicapped persons.). It can be easily foreseen that the robot will have to make return journeys in the user's flat. The problem is then fourfold: i) the journey must be described using a human-like language, ii) series of frescoes are inferred from the route description, iii) navigation uses these series to lead the robot to the target point, iv) the robot has to return to its starting point and must retrieve its route using only the pertinent frescoes recorded when on the way on?

Point 1 was studied in (Saïdi, 2006). From a high-level route description, the robot's journey is built. The problem was extended to a group of robots. To solve point 4, selected frescoes describing the way on are stored in robot's memory (LIFO). After having been processed its task, the robot has to return on a route that is not exactly the same than the way on. Therefore, the current fresco does not correspond exactly to the stored frescoes (the 180° rotation is, obviously, taken into account): the fresco and one situated on the top of the LIFO do not correspond. A first method consists in shifting left or right the current fresco to better fit to one of the stored frescoes (Pradel, 2000). Another method consisting in gathering landmarks into representative sets (alcove, cupboard ...) and using all possible transformations of the current fresco is too time consuming. On the contrary, a method grounded on the study of the evolution of very small groups of landmarks is more promising, simple and low resource consuming. On the other hand, with this method, the robot must anticipate the future environments. This anticipation, even if it needs a complete description of all transforms of a fresco, is simpler when the fresco is split into small groups of landmarks. Anticipating frescoes from the current one and comparing them with the stored frescoes seems to be a promising method that will allow the robot to choose the right return way. First results show that the robot is able to return to its starting point in various environments. Nevertheless, the method must be validated in complex and changing environments.
Present and future works focus on points 2, 3. Another perspective is to use a single vision sensor (CCD camera) instead of the laser range finder, extracting distances from images to build a structure similar to frescoes.

9. References

Aha, D.W.; Kibler, D. & Albert, M.K. (1991). Instance-based Learning Algorithms, *Machine Learning*, vol. 6(1), pp. 37-66, Springer Netherlands, ISSN: 0885-6125, 1991

Ahuactzin, J; E. Mazer, E. & Bessière, P, (1995). L'algorithme fil d'Ariane, *Revue d'intelligence artificielle*, vol. 9(1), pp. 7-34, 1995

Al Allan, S.; Pradel, G.; Barret, C. & Abou Kandil, H. (1995). Neural Architectures for Mobile Robot Navigation, *International Conference on Advanced Robotics and Intelligent Automation*, pp. 440-445, Athens. Greece, 1995

Altschul, S., (1991). Amino acid substitution matricesfrom an information theoretic perspective, *Journal of Molecular Biology*, vol. 219, pp. 555-565, 1991

Arkin, G., (1998). Nomad 200 – simulator MissionLab, *Behaviour-Based Robotics*, MIT Press, 1998

Bras, F.; Pradel, G. & Jin, Z., (1995). Cellular Automata Applied to the Path Generation and Environment Representation for a Mobile Robot, *IFAC Motion Control Conference*, pp. 395-402, Munich, Germany, 1995

Blanzieri, E. & Rizzi,F. (1999). Advanced metrics for class driven similarity search, *Proceedings of 10th International Workshop on Database and System Application*, pp. 223-227, Firenze, Italy, 1999

Choset, H. & Nagatani, K. (2001). Topological Simultaneous localization and Mapping (SLAM): Toward Exact Localization without Explicit Localization, *IEEE Transaction on Robotics and Automation*, vol. 2(17), pp. 125-137, 2001

Collet, T.; Dillmann, E.; Giger, A.; Wehner, R. (1992). Visual landmarks and route following in desert ants, *Journal of Comparative Physiology, Springler Verlag*, vol. 170, pp. 435-442, 1992

Crosnier, A. (1999). Modélisation géométrique des environnements en robotique mobile, *French Workshop on Robotic Resarch (Journées Nationales de la Recherche en Robotique)*, Montpellier, France, pp. 83-91, 1999

DeSouza, G.N. & Kak, A.C. (2002). Vision for mobile robot navigation: a survey, *IEEE Transaction on pattern analysis and machine intelligence, Vol. 24(2)*, pp. 237-267, 2002

Filliat, D. and Meyer, J.A. (2002a). Map-Based Navigation in Mobile Robots. I. A review of Localization strategies, *http://citeseer.ist.psu.edu/filliat03mapbased.html*

Filliat, D. & Meyer, J.A. (2002b). Global localization and topological map-learning for robot navigation, *http://citeseer.ist.psu.edu/filliat02global.html*

Fleisher, J.; Marshland S. & Shapiro, J. (2003). Sensory anticipation for autonomous selection of robot landmarks, *http://www.cs.man.ac.uk\~fleischj/research.html*

Franz, M.; Schölkopf, B. & Bülthoff, H. (1997). Image-based Homing, *Proceedings of the European Conference on Artificial Life*, pp. 236-245

Gaussier, P. ; Joulain, C.; S.~Zrehen, S. & Revel, A. (1997). Image-based Homing, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 545-550, 1997

Gusfield, D. (1997). Algorithms on strings, Trees and Sequences, *Computer Science and computational Biology*, Cambridge University Press, NY, USA, 1997

Haykin, S. (1999). Neural neworks, A Comprehensive Foundation, second Edition, Prentice Hall, ISBN: 0 13 273350 1, 1999

Hong, I. (1991). Image-based Homing, *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, USA, pp. 620-625, 1991

Hoppenot, P.; Pradel, G.; Căleanu, C.D.; Perrin, N. & Sommeilly, V. (2003). Towards a symbolic representation of an indoor environment, *Proc. IEEE-SEE-CESA2003 - Computing Engineering in Systems Applications,* CD ROM paper no. S1-R-00-0048, Lille, France, 2003

Huttenlocher, D.; Klanderman, G. & Rucklidge, W. (1993). Comparing images using Hausdorff distance, *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15(9), pp. 850-863, 1993

Judd, S. & Collett, T. (1998). A Mobile Robot That Learns Its Place, *Nature*, vol. 392, pp. 710-714, 1998.

Karlin, S. & Altschul, S. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes, *proceedings of the national Academy of Science*, vol. 87, pp. 2264-2268, 1990

Kim, D. & Neviata, R. (1994). A method for recognition and localization of generic objects for indoor navigation, *Proceedings of ARPA Image Understanding Workshop*, Monterey, USA, 1994

Kohonen, T. (1987). Content-Addressable Memories, *Springer Series in Information Sciences*, vol. 1, Springer Berlin Heidelberg, 1987

Kohonen, T. (1988). Self-Organization and Associative Memory, *Springer Series in Information Sciences*, vol. 8, Springer Berlin Heidelberg, 1988

Kohonen, T. & Somuervo, P. (1998). Self-organizing maps of symbols strings, *Neurocomputing*, vol. 21, pp. 19-30, Elsevier, ISSN: 0925-2312, 1998

Kuipers, B. & Byan, Y.T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representation, *International Journal of Autonomous Systems*, vol. 8, pp. 47-63.

Kulic, R. and Vukic, Z. (2003). Methodology of concept control synthesis to avoid. unmoving and moving obstacles, *The Knowledge Engineering Review*, Vol. 18, issue 3, pp. 281-291, Cambridge University Press

Lamon, P.; Nourbakhsh, I.; Jensen, B. & Siegwart, R. (2001). Deriving and Matching Image Fingerprint Sequences for Mobile Robot Localization, *http://citeseer.nj.nec.com/445679.html*, *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, 2001

Leonard, J. J. & Durrant-Whyte, H.F. (1991). Simultaneous map building and localization for an autonomous mobile robot, *Proceedings of the IEEE/RSJ Int. Workshop on Intelligent Robots and Systems IROS'91*, pp. 1442-1447, New York, NY, USA, 1991

Levenshtein, L.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals*, Soviet Physics–Doklady*, vol. 10, no.7, pp. 707–710, 1966.

MacKenzie, P. & Dudek, G. (1994). Precise Positioning using Model-Based Maps, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1615-1621, San Diego, USA, 1994

Meyer, J.A. & Filliat, D. (2002). Map-based navigation in mobile robots - II. A review of map-learning and path-planning strategies, *http://citeseer.ist.psu.edu/meyer03mapbased.html*

Oore, S.; Hinton, G. & Dudek, G. (1997). A Mobile Robot That Learns Its Place, *Neural Computation,* pp. 683-699, *MIT Press*, vol. 9(3)

Pomerleau, D.A. (1993). Neural Network Perception for Mobile Robot Guidance}, *Kluwer Academic Publishers*.

Pradel, G.; Bras, F. & Jin, Z. (1994). 2D laser telemetry-based path trend generation and real time environment symbolic representation for an autonomous mobile robot, *Proceedings of the IFAC-IEEE International Conference on Machine Automation*, pp. 122-134, Tampere, Finland, 1994

Pradel, G.; Avrillon, S. & Garbuio, L. (2000). Landmark interpretation by means of frescoes in mobile robotics, *Proceedings of the 6th Int. Conf. On Methods and Models in Automation and Robotics*}, pp. 585-592, Miedzyzdroye, Poland, 2000.

Pradel, G. & Bras, F. (2001). Qualitative environment description by means of frescoes in mobile robotics, *Journal européen des systèmes automatisés*, vol. 9(35), pp. 1105-1128, ISSN 1269-6935, 2001.

Saïdi, F. & Pradel, G. (2006). A multi-robot path planner for group navigation, *Journal of Intelligent and Robotics Systems*, Springer-Verlag eds, under press, 2006

Simhon, S. & Dudek, G. (1998a). A global Topological Map formed by Local Metric Maps, *Proceedings of IEEE/RSJ International Conference on Intelligent Robotic Systems*, Victoria, B.C., Canada, pp. 1708--1714, 1998

Smith, C.M. & Leonard, J.J. (1997). A multiple-hypothesis approach to concurrent mapping and localization for autonomous underwater vehicles, *Proceedings of International Conference on Field and Service Robotics,* pp.249-256, Canberra, Australia, 1997

Tedder, M. & Hall, L.E. (2001). Symbolic processing methods for 3D visual processing, *http://www.robotics.uc.edu/papers2001/Maurice2001d.pdf*

Thrun, S. (2002). Robotic mapping: A survey, *Exploring Artificial Intelligence in the New Millenium*, Morgan Kaufmann, *http://citeseer.ist.psu.edu/thrun02robotic.html*

Wagner, R.A. & M. J. Fischer, M.J. (1974). The string to string correction problem, *Journal of the ACM*, 21, pp. 168-173, 1974.

Weber, K.; Venkatesh, S. & Srinivasan, M.V. (1999). Insect Inspired Robotic Homing, *Adaptive Behavior*, vol. 1, pp. 65-98

Wichert, G. (1996). Selforganizing Visual Perception for Mobile Robot Navigation, *http://www.rt.e-technik.th-darmstadt.de/~georg/pubs/EUROBOTS96/paper.html*

26

Robot Mapping and Navigation by Fusing Sensory Information

Maki K. Habib
Graduate School of Science and Engineering, *Saga University*
Japan

1. Introduction

Intelligent and autonomous mobile robots are required to wander around and explore their environment without colliding with any obstacles (stationary or moving) for the purpose to fulfill its mission by executing successfully an assigned task, and to survive by affording the possibility of finding energy sources and avoid dangerous hazards. To efficiently carry out complex missions, autonomous robots need to learn and maintain a model of their environment. The acquired knowledge through learning process is used to build an internal representation. Knowledge differs from information in that it is structured in long-term memory and it is the outcome of learning. In order to enable an autonomous mobile robot to navigate in unknown or changing environment and to update in real-time the existing knowledge of robot's surroundings, it is important to have an adaptable representation of such knowledge and maintain a dynamic model of its environment. Navigation in unknown or partially unknown environments remains one of the biggest challenges in today's autonomous mobile robots. Mobile robot dynamic navigation, perception, modeling, localization, and mapping robot's environment have been central research topics in the field of developing robust and reliable navigation approach for autonomous mobile robots. To efficiently carry out complex missions in indoor environments, autonomous mobile robots must be able to acquire and maintain models of their environments. Robotic mapping addresses the problem of acquiring spatial models of physical environments through mobile robots and it is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots. Acquiring and mapping unstructured, dynamic, or large-scale environments remains largely an open research problem. (Habib & Yuta, 1988; Kuipers & Byun, 1991; Thrun & Bucken, 1996; Murphy, 2000; Thrun, 2002). There are many factors imposing practical limitations on a robot's ability to learn and use accurate models. The availability of efficient mapping systems to produce accurate representations of initially unknown environments is undoubtedly one of the main requirements for autonomous mobile robots.

A key component of this task is the robot's ability to ascertain its location in the partially explored map or to determine that it has entered new territory. Accurate localization is a prerequisite for building a good map, and having an accurate map is essential for good localization (Se et al., 2002; Choset, 2001). All robots, which do not use pre-placed landmarks or GPS must employ a localization algorithm while mapping an unknown space. Therefore, accurate simultaneous localization and mapping (SLAM) represents a critical factor for

successful mobile robot dynamic navigation in a large and complex environment because it enables the robot to function autonomously, intelligently, purposefully, and robustly. The term SLAM was first coined to describe a technique used by robots and autonomous vehicles to build up a map within unknown environment while at the same time keeping track of its current position. This technique has attracted immense attention in the mobile robotics literature and has been applied successfully by many researchers (Se et al., 2002; Choset & Nagatani, 2001). SLAM has not yet been fully perfected, but it is starting to be employed in unmanned aerial vehicles, autonomous underwater vehicles, planetary rovers, and newly emerging domestic robots. All the numerous methods proposed in literature are based on some sort of incremental integration: a newly acquired partial map is integrated with the old maps. To integrate the partial map obtained at each sensing step into the global map of the environment, the localization of the robot is fundamental. To perform localization, it needs to estimate both robot's pose and obstacles positions are needed.

Map building in an unknown dynamic environment has been under study for a long time and many different approaches have been developed and evaluated (Borenstien & Koren, 1991a; Thrun & Bucken, 1996; Singhal, 1997; Borenstien & Ulrich, 1998; Murphy, 2000; Ellore, 2002). Other important issues related to navigation of an autonomous mobile robot are, dealing with moving obstacles/objects, and fusing sensory information from multiple heterogeneous sensors. These issues usually cannot be resolved through the use of conventional navigation techniques.

During real time simultaneous map building and localization, the robot is incrementally conducting distance measurements. At any iteration of map building the measured distance and direction traveled will have a slight inaccuracy, and then any features being added to the map will contain corresponding errors. If unchecked, these positional errors build cumulatively, grossly distorting the map and therefore affect the robot's ability to know its precise location. One of the greatest difficulties of map building arises from the nature of the inaccuracies and uncertainties in terms of noise in sensor measurements, which often lead to inaccurate maps. If the noise in different measurements is statistically independent, a robot can simply take multiple measurements to cancel out the effects of the noise. But, the measurement errors are statistically dependent due to odometry errors that accumulate over time and affect the way that future sensor measurements are interpreted. Small odometry errors can have large effects on later position estimates. There are various techniques to compensate for this, such as recognizing features that the robot has come across previously and re-skewing recent parts of the map to make sure the two instances of that feature become one. For the last decade, the field of robot mapping has been dominated by probabilistic techniques for simultaneously solving the mapping problem and the induced problem of localizing the robot relative to its growing map and accordingly different approaches have been evolved. The first category includes approaches that employ Kalman filter to estimate the map and the robot location (Lu & Milios, 1997; Castellanos & Tardos, 1999; Thrun, 2002). Another approach is based on Dempster's expectation maximization algorithm (Thrun, 2001; Thrun, 2002). This category specifically addresses the correspondence problem in mapping, which is the problem of determining whether sensor measurement recorded at different points in time correspond to the same physical entity in the real world. The Extended Kalman Filter (EKF) has been the de facto approach to the SLAM problem. However, the EKF has two serious deficiencies that prevent it from being applied to large, real-world environments: quadratic complexity and sensitivity to failures

in data association. An alternative approach called Fast-SLAM is based on the Rao-Blackwellized Particle Filter, and can scale logarithmically with the number of landmarks in the map (Montemerlo & Thrun, 2003). The other category of approaches seeks to identify objects and landmarks in the environment, which may correspond to ceilings, walls, doors, furniture and other objects that move. For the last two decades, there has been made tremendous progress in the development of efficient and highly accurate map building techniques. Most of these techniques focus either on capturing the metric layout of an environment with high accuracy (Moravec & Elfes, 1985; Moravec, 1988; Elfes, 1989a; Elfes, 1989b; Borenstien & Koren, 1991a and b; Borenstien & Koren, 1998; Ribo & Pinz, 2001; Ellore, 2002), or on representing the topological structure of an environment (Habib & Yuta, 1988; Kuipers & Byun, 1991; Habib & Yuta, 1993; Choset & Nagatani, 2001).

To acquire a map and achieve efficient simultaneous localization, robots must possess sensors that enable them to perceive the outside world. There are different types of sensor modalities commonly brought to bear for this task such as ultrasonic, laser range finders, radar, compasses, vision, infrared, tactile sensors, etc. However, while most robot sensors are subjected to strict range limitations, all these sensors are subject to errors, often referred to as measurement noise. Laser scanning system is active, accurate but slow. Vision systems are passive and of high resolution but it demands high computation Ultrasonic range finders are common in mobile robot navigation due to their simplicity of operation, high working speed and cheap but usually they are very crude. These sensors provide relative distances between them and surrounding obstacles/objects located within their radiation cone. However, these devices are prone to several measuring errors due to various phenomena, such as, multiple reflections, wide radiation cone, and low angular resolution. Robot motion is also subject to errors, and the controls alone are therefore insufficient to determine a robot's pose relative to its environment. Hence, one of the main problems in SLAM is coming from the uncertainty in the estimated robot pose. This uncertainty creates correlation between the robot pose and the estimated map. Maintaining such a correlation increases computational complexity. This characteristic of SLAM makes the algorithm hard to apply to estimate very dense maps due to the computational burden.

The objectives of this chapter are to discuss and understand the importance, the complexity, and the challenges of mapping robot's unknown, unstructured and dynamic environment, besides the role of sensors and the problems inherited in map building. These issues remain largely an open research problem facing the development of dynamic navigation systems for mobile robots. In addition, the chapter aims to discuss the available techniques and approaches of mainly indoor map building supporting the navigation of mobile robots. Furthermore, it introduces an autonomous map building and maintenance method with focus on developing an incremental approach that is suitable for real-time obstacle detection and avoidance. The robot maps its environment incrementally while wandering in it and staying away from all obstacles. In this case, the navigation of mobile robots can be treated as a problem of tracking geometric features that occur naturally in the environment of the robot. The implementation of the developed technique uses the concept of occupancy grids and a modified Histogrammic In-Motion Mapping (HIMM) algorithm to build and maintain the environment of the robot by enabling the robot to recognize and track the elements of the occupancy grid in real-time. In parallel to this, the incrementally built and maintained map has been integrated directly to support navigation and obstacle avoidance. To ensure real-time operation with limited resources, as well as to promote extensibility, these

modules were deployed in parallel and in distributed framework. Simulation based experiments has been conducted and illustrated to show the validity of the developed mapping and obstacle avoidance approach.

2. Mapping Approaches and Autonomous Mobile Robots Navigation

Mapping is the process of generating models of a mobile robot's environment based on sensory information with aim to determine the location of various entities, such as landmarks or obstacles. Most successful navigation algorithms require the availability of dynamic and adaptable maps. An accurate model of the environment surrounding a robot enables it to complete complex tasks quickly, reliably and successfully. Without such a model, a robot neither can plan a path to a place not currently sensed by its sensors nor may effectively search for an object or place.
Map based navigation calls for three processes: map learning, localization, and path planning (Levitt & Lawton, 1990). These three processes may rely on three distinct sources of information available to a robot; the first source enables the robot to sense its own personal configurations and its relationship to the environment; the second is required to enable the robot to survive in the mission's environment by detecting and avoiding obstacles along the desired trajectory toward its target; and the third is to recognize the associated requirements and features of the targeted mission and assure its successful execution. There are many ways to integrate these sources of information in a representation that is useful for robot navigation. The drawbacks and advantages of these sources of information are complementary. Localization is a critical issue in mobile robotics. If a robot does not know where it is, it cannot effectively plan movements, locate objects, or reach goals. Localization and map learning are interdependence processes, because using a map to localize a robot requires a fact that the map exists and building a map requires the position to be estimated relative to the partial map learned so far. Map building and learning is a crucial issue in autonomous mobile robot navigation, where robots must acquire appropriate models of their environment from their actual sensory perceptions while interacting with the real world. Generally, there are three main approaches to generate purposeful maps. Metric (geometric), topological or hybrid navigation schemes have been proposed and studied. Each one of these methods is optimal concerning some characteristics but can be very disappointing with respect to other requirements. These approaches present complementary strengths and weaknesses.
The mapping problem is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots. Despite significant progress in this area, it still poses great challenges (Thrun, 2002). Robust methods are available for mapping environments that are static, structured, and of limited size, whereas mapping unstructured, dynamic, or large-scale environments remains largely as an open research issue. Since the 1990s, the field of robot mapping has been dominated by probabilistic techniques because of its ability to deal successfully with the inherent uncertainties associated with robot perception that would otherwise make map-building a very hard process. Some algorithms are building the map incrementally to suit real-time needs, whereas others require multiple passes through the data. Some algorithms require exact pose information to build a map, whereas others can do so by using odometry measurements. Dynamic environment with dynamic objects, and imperfect sensors and actuators can lead to serious errors in the resulting map. The problem of learning maps in unknown and dynamic environment has been studied by many researchers. Some of these efforts include tracking dynamic objects while filtering out the measurements reflected by those objects (Hahnel et al. 2002; Wang &

Thorpe 2002). The approach developed by Hahnel et al, interleaves mapping and localization with a probabilistic technique to identify spurious measurements. Such approaches have been demonstrated some robustness compared to traditional approaches, but they results in producing maps that describe the static aspects of the environment. Enhanced sensor models combined with the EM algorithm have been used to filter out arbitrary dynamic objects to improve the scan registration and lead to more accurate maps (Hahnel et al., 2003). Another approach deals with mapping dynamic environments by explicitly modelling the low-dynamic or quasi-static states, and this was achieved by dividing the entire map of the environment into several sub-maps and learns the typical configurations of the corresponding part of the environment by clustering local grid maps for each of the sub-maps (Stachniss & Burgard, 2005). Similar work was developed aims to learn models of non-stationary objects from proximity data by applying a hierarchical EM algorithm based on occupancy grids recorded at different points in time. A robotic mapping method based on locally consistent 3D laser range scans has been developed (Nüchter et al. 2006). In this method, scan matching is combined with a heuristic for closed loop detection and a global relaxation method, results in a 6D concurrent localization and mapping on natural surfaces. In recent years, a few researchers have discussed the possibility of decoupling the mapping and localization processes in SLAM in order to gain computational efficiency. Since the observations made by a robot are about the relative locations between the robot and features, a natural way to decouple mapping and localization is to extract information about the relative locations among the features and then construct a relative map using this part of information. Martinelli et al (2004) made use of relative maps where the map state only contains distances between features that are invariants under shift and rotation. But, this approach has redundant elements in the state vector of the relative map while a significant increase in computational complexity of the SLAM algorithm results due to constraints applied to avoid the generation of an inconsistent map.Furthermore, issues of how to extract the information about the relative map from the original observations has has not been been addressed yet properly. Some of these issues have been tackled by developing a decoupled SLAM algorithm (D-SLAM) based on a new formulation of 2-D relative map with no redundant elements (Wang et al., 2005). The Extended Kalman Filter (EKF) has served as the primary approach to SLAM for the last fifteen years. However, EKF-based SLAM algorithms suffer from two well known shortcomings that complicate their application to large, real world environments. These two problems are the quadratic complexity, and the sensitivity to failures in data association. The EKF has become widely known in terms of the growth of complexity due to the update step that requires computation time proportional to the square of the number of landmarks. This obviously becomes difficult to deal with in large environments. To address these issues, FastSLAM family of algorithms that applies particle filters to the SLAM problem have been developed to provide new insights into the data association problem. In the original form of FastSLAM, each particle may have different number of features and maintains these features in its local map. Multiple readings can be incorporated per time step by processing each observation sequentially to increase the accuracy of data association. The weight for each particle is equal to the product of the weights due to each observation considered alone (Montemerlo & Thrun, 2003). Modification, extensions and analysis has been made to the original FastSLAM approach, and the recent analysis shows that it degenerates with time, regardless of the number of particles used or the density of landmarks within the environment, and

will always produce optimistic estimates of uncertainty in the long-term. FastSLAM behaves like a non-optimal local search algorithm; in the short-term it may produce consistent uncertainty estimates but, in the long-term, it is unable to adequately explore the state-space to be a reasonable Bayesian estimator. However, the number of particles and landmarks does affect the accuracy of the estimated mean. Given sufficient particles FastSLAM can produce good non-stochastic estimates in practice (Bailey et al., 2006). FastSLAM has several practical advantages, particularly with regard to data association, and may work well in combination with other versions of stochastic SLAM, such as EKF-based SLAM. FastSLAM-type algorithms have enabled robots to acquire maps of unprecedented size and accuracy, in a number of robot application domains and have been successfully applied in different dynamic environments, including the solution to the problem of people tracking (Nieto et al, 2003; Montemerlo & Thrun, 2007). Since the complexity of the global map cannot be avoided, some researchers have proposed hierarchical division of the global map into sub-maps, within which the complexity can be bounded through the use of the generalized Voronoi graph (GVG). The GVG serves as a high-level topological map organizing a collection of feature-based maps at the lower level, and this leads to create a hierarchical approach to the simultaneous localization and mapping (Lisien et al, 2003).

2.1 Types of map representation

2.1.1 Metric maps

The geometric approach to world representation attempts to capture the geometric properties, and build a detailed metrical description of robot's environment from sensor data (Preciado et al., 1991; Durrant-Whyte, 1998; Thurn, 2001). This kind of representations has a reasonably well defined relation to the real world, but it is highly vulnerable to metrical inaccuracy in sensory devices and movement of actuators.

The map quantitatively reproduces the metric spatial features of the environment in an accurate way. The occupancy grid framework represents a fundamental departure from conventional traditional geometric approaches and it uses a stochastic tessellated representation of spatial information maintaining probabilistic estimates of the occupancy state of each cell in a spatial lattice. An early representative of the grid based metric approach was Elfes and Moravec's important occupancy grid mapping algorithm (Elfes, 1987; Moravec, 1988; Elfes, 1989a; Elfes, 1989b). In this approach, the metric map represents the environment with a grid of fixed resolution. At a low level, a local map, such as a fine array of pixels can model the environment of the robot. The grids hold information of the observed environment and each cell of the grid represents some amount of space in the real world. Sensors are used to detect obstacles and objects, and the acquired sensory data are used to determine the occupancy value of each grid space so that a map can be generated. Each grid cell estimates the occupancy probability of the corresponding region of the environment. Metric maps are easy to construct because grids reproduce explicitly the metrical structure of the environment. Besides, the geometry of the grid corresponds directly to that of the real environment, so that the robot position and orientation within its model can be determined by its position and orientation in the real world. However, any approaches using purely metric maps are vulnerable to inaccuracies in both map-making and dead-reckoning abilities of the robot. Even by taking into account all relationships between features and the robot itself, in large environments the drift in the odometry causes big troubles

for map maintenance and makes the global consistency of the map difficult to maintain. Landmark-based approaches, which rely on the topology of the environment, can better handle this problem, because they only have to maintain topological global consistency, not the metric one. While grid-based methods produce accurate metric maps, their complexity often prohibits efficient planning and problem solving in large-scale indoor environments, and it can be very memory intensive as the world map grows. This is because the resolution of a grid must be fine enough to capture every important detail of the world (Kuipers & Byun, 1991; Engelson & McDermott, 1992; Borghi & Brugali, 1995; Thrun & Bucken, 1996). On the downside, it is slow, not expressive and also it is associated with local minima problem. The grid based metric approach has been used in a great number of robotic systems, such as (Borenstien & Koren, 1991a and b; Burgard et al., 1999; Yamaguchi & Langley, 2000). Alternative metric mapping algorithms were proposed by many researchers aiming to describe the free space geometrically using natural landmarks, polyhedral shapes, prime rectangular area, and cones to formulate the geometrical features of the environments (Habib & Yuta, 1993; Chatila & Laumond, 1985; Habib & Yuta, 1990; Brooks, 1983). While metric maps have the advantage of its high resolution and of being well suited for robot navigation tasks, they are typically unstructured and contain no information about the different types of places or objects in the environment. Also, it suffers from their enormous space and time complexity.

2.1.2 Topological maps

Topology maps describe the connectivity of different places and it can be automatically extracted while building metric maps. They can also be enriched by path shape information, i.e. the geometrical relations between places to help planning and navigation (Fabrizi & Saffiotti, 2000; Engelson & McDermott, 1992).

A topological map is a feature-based map that uses symbolic description. Such maps can be used to reduce the SLAM problem to a graph-matching problem at the topological scale. Hence, in office buildings with corridors and rooms, or roads, the topology of important locations and their connections can highly support the navigational requirements. The topological approach is a qualitative one, less sensitive to sensor errors, less complex and permits more efficient planning than metric maps. Topological maps can capture the structure of the free space in the environment in terms of basic topological notions of connectivity and adjacency as a list of significant places connected via arcs that are needed to plan a navigational strategy. At high level, a topological map serves as an example of symbols and connects between them. The symbols that are local maxima of the distance to nearby obstacles are nodes in a graph and correspond to perceptually distinct regions, places, landmarks or situations. While the connections between the symbols represent the graph edges/arcs that link the distinct places to indicate the spatial relations between them, i.e. a direct path exists between them. Arcs are usually annotated with information on how to navigate from one place to another. For an indoor building environment, junctions and termination points of corridors represent symbols while the corridors themselves are the connections (Habib & Yuta, 1988 ; Habib & Yuta, 1993).

Distances are usually not considered and instead only relative positions are considered. The graph describing a topological map is good to be stored and communicated because it has the advantage of using a small amount of memory and the ability to record only parts of the environment that are of interest and necessary for navigation. Such models involve more

compact representations. However, it relies heavily on the existence of recognizable landmarks (Kuipers & Byun, 1991; Engelson & McDermott, 1992; Borghi & Brugali, 1995). Since sensory input depends strongly on the viewpoint of the robot, topological approaches may fail to recognize geometrically nearby places. The resolution of topological maps corresponds directly to the complexity of the environment. The compactness of topological representations gives them three key advantages over grid-based approaches: they permit fast planning; they facilitate interfacing to symbolic planners and problem-solvers; and they provide natural interfaces for human instructions. Moreover, it is easier to generate and maintain global consistency for topological maps than for metric maps. Since topological approaches usually do not require the exact determination of the geometric position of the robot, they often recover better from drift and slippage that must constantly be monitored and compensated in grid-based approaches. However, these techniques often ignore valuable metric information and they still lack the ability to describe individual objects in an environment. In addition, coherent topological maps are often difficult to learn and maintain in large-scale environments. A fundamental problem in topological mapping is recognizing an already-visited place, i.e., closing the loop. With sufficiently rich sensory information and features abstraction, detecting unique sensing signatures that describe such places will easily solve this problem. Topology-based maps are fairly robust against sensor noise and small environmental changes, and have nice computational properties. Some examples of topological approaches are available in the following references (Habib & Yuta, 1988; Mataric, 1990; Kuipers & Byun, 1991; Engelson & McDermott, 1992; Habib & Yuta, 1993; Choset, 1996; Kuipers et al, 2004).

2.1.3 Hybrid maps

The hybrid integration of both metric and topological paradigms is gaining popularity among researchers because it leads to maximizing their effectiveness by combining the best characteristics of both universes and can support to cover a large scale environment. For this, the overall environmental model embodies both a metric and a topological representation (Habib & Yuta, 1993). In a hybrid map, the environment is described by a global topological map, so that it facilitates planning and re-planning within the whole environment to enable the robot finding a suitable route to its currently stated target; on the contrary, local metric maps can be used by the robot for safe navigation toward a given target, and the robot needs further localization precision. The world can be represented as a graph, like a pure topological map. The nodes in this graph represent topological locations and each node can be a metric map. The traveling between nodes along the edges causes a switch from the topological to the metric paradigm. A global topological map connects the disconnected local metric maps, and allows the representation of a compact environment model, which does not require global metric consistency and permits both precision and robustness. The environment models allow the use of two different navigation methods with complementary characteristics. This gives the map precise detail where it is needed, while leaving out details that are not needed, and this can reduce memory size dramatically. Detectable metric features are needed to determine the transition point and to initialize the metric localization. The metric localization permits a very precise positioning at the goal point whereas the topological one guarantees robustness against getting lost due to the multimodal representation of robot's location. The effectiveness of such integration for

navigation and localization has already been demonstrated in different work (Elfes, 1987; Habib & Yuta, 1993; Tomatis et al., 2001).
In (Habib & Yuta 1988; Habib & Yuta 1993) the authors have developed and implemented a 4-levels hierarchical topological map integrated with local metric maps. The 4-levels hierarchical topological map used to describe large indoor environment consists of a network of buildings connected through common corridors like passage ways. The four levels are

a) Network of buildings with nodes at the center of each passage way between adjacent buildings,
b) Network of corridors within each building. Each corridor has two nodes, one at each of its terminals.
c) Rooms within each corridor. Each room is represented by a node at each entry point of the associated doors,
d) The free space within a room is structured into prime rectangular areas (PRAs). The PRAs are intersecting with each other, and the header of each PRA is treated as a node.

The geometric maps are generated and required for each corridor, and each PRA that are part of a generated path connecting the current robot location to a given target.

2.1.4 Other classification of maps

Another way to classify the available mapping algorithms is world-centric versus robot-centric. World-centric maps are represented in a global coordinate space. The entities in the map do not carry information about the sensor measurements that led to their discovery. Robot-centric maps, in contrast, are described in measurement space. They describe the sensor measurements that a robot would receive at different locations (Thrun, 2002). Robot-centric maps suffer two disadvantages. First, it is often difficult to infer or estimate by extending or projecting known information (extrapolate) of individual measurement to measurement at nearby, unexplored places, i.e., there is usually no obvious geometry in measurement space that would allow for such extrapolation. Such extrapolation is typically straightforward in world-centric approaches. Second, if different places look alike, robot-centric approaches often face difficulties to disambiguate them, again due to the lack of an obvious geometry in measurement space. For these reasons, the dominant approaches to date generate world-centric maps.

2.2 Sensors and problems inherited in map building

Map building potentially provides a good framework for integrating information from many different sensors, of the same types or heterogeneous types that can perceive the surroundings from many different poses (position and orientation) into a single knowledge source. The use of heterogeneous sensors is essential because it compensates for many weaknesses inherent in various sensors used in map building. If the information from different sets of sensors is integrated into the same map, then they can be used to either confirm or dispute each other, resulting in a more robust and less error-prone model of the environment. One of the most prominent difficulties in mapping an environment is in knowing the exact position and orientation the robot was in when it received the sensor readings. If a wrong position is estimated, then the incorrect part of the map will be updated and this leads to large errors. Information regarding the distance the robot has traveled, and in which direction, is usually calculated by

measuring the number of times each wheel has turned, i.e., from odometry. Unfortunately, occurrences of wheel slippage and angular drift can lead to the estimation of wrong robot position and orientation. It is therefore necessary to integrate other sensors with the robot, or recognizing certain situations with environmental features to help correct errors in pose estimation. This is called localization. Global localization is necessary when the robot has a previously generated map of its environment. When it is first turned on, it must be able to know where it is within that map in order to incorporate its new sensor readings into the correct area of that map. Position tracking is used to compensate for wheel slippage and angular drift. When performing map building with mobile robots, simultaneous localization and map building is usually performed in parallel with continuous corrected update of robot's estimated pose.

In general, the real world is: a harsh place that is not complete and with no consistent environmental knowledge, non-deterministic (interactions in dynamic and open environments can neither be formulated, nor designed, or analyzed fully using stationary or static models), dynamic (changes happen at the time of decisions are made, such as closing/opening motion of doors, movement of chairs and people, etc.), and it is continuous and not discrete. Accordingly, the robot should not only be capable of controlling their motion in response to sensor inputs, but it should capable of,

a) reacting to unexpected events and accordingly change course if necessary by efficiently deciding the next action to be performed,
b) learning from their experiences in different ways to improve their behavior and performance with time and when facing similar situation,
c) considering multiple conflicting objectives simultaneously, and overcome errors in perceptions and actions.

Primary difficulties arise when building maps of an environment, and such difficulties may include:

a) Sensors are inaccurate, noisy, faulty, and with limited field of view, which means that decisions might be based on wrong information. Limitation is related to range, resolution, occlusion, etc. Accordingly, there is a need to efficiently interpret sensor readings into knowledge about the environment, while compensating for the considerable amount of inaccuracy in sensor readings.
b) The more information is stored about an environment by increasing its dimensionality, the more use can be made of it, but also the more memory and computation time is required.
c) The time available to decide what to do is limited, because the robot needs to respond quickly to environmental demand and has to operate at a pace dictated by its surroundings. Such limit to respond may be due to computation time or control paradigm
d) A robot cannot assume correct and perfect execution of its actions due to imperfections of actuators and uncertainties in the environment, such as the accumulation of odometry errors over time. Accordingly, there is a need of a reliable approach to estimate the position and orientation of the robot.
e) The dynamic and complex features of robot environments make it principally impossible to maintain exact models and to predict accurately.

The problem is how to extract reliable evidence from unreliable sources of information.

3. Representation Requirements and Incremental Grid Map Building

Any representation used for modeling physical attributes of an environment and any of the entities present in it, must possess certain qualities. It must be powerful enough to express all the entities that need to be represented. At the same time, it must be adaptive in those entities that affect the navigation strategy at any given state of the environment. Also, the characteristics of any representation must be able to quickly update its knowledge about the current state of the environment without heavy computation effort (Singhal, 1997).
In order to effectively support the goal of acquiring autonomously a world model, the robot must exhibit several characteristics: able to manage a collection of different sensors; provide mechanisms to manage the uncertainty associated with sensory information; able to recover the uncertainty on its position, and able to provide exploration strategies in order to plan autonomously the sequence of actions that allow to execute an assigned task.

3.1 Spatial information and occupancy grid

The spatial information is represented by a metric map and it is based on occupancy grids. The occupancy grids, also known as evidence grids or certainty grids were pioneered by Moravec and Elfes (Moravec & Elfes, 1985; Elfes, 1987, Moravec, 1988; Elfes, 1989a; Elfes, 1989b) and formulated in CMU (Martin & Moravec, 1996) as a way to construct an internal representation of static environments by evenly spaced grids based on ultrasonic range measurements. Occupancy grids provide a data structure that allows for fusion of sensor data. It provides a representation of the world which is created with inputs from the sensors. Apart from being used directly for sensor fusion, there also exist interesting variations of evidence grids, such as place-centric grids (Youngblood et al., 2000), histogram grids (Koren & Borenstein, 1991) and response grids (Howard & Kitchen, 1996). The variations will not be studied in this chapter. Occupancy Grids is certainly the state of the art method in the field of grid based mapping. It is the most widely used robot mapping technique due to its simplicity and robustness and also because it is flexible enough to accommodate many kinds of spatial sensors with different modalities and combining different sensor scans. It also adapts well to dynamic environments. Occupancy grids have been implemented with laser range finders, stereo vision sensors (Moravec, 1996) and even with a combination of sonar, infrared sensors and sensory data obtained from stereo vision (Lanthier et al., 2004).
In general, the occupancy grid technique divides the environment into two dimensional discrete grid cells as shown in Fig. 1.a. The occupancy grids map is considered as a discrete state stochastic process defined over a set of continuous spatial coordinates. Each grid cell is an element and represents an area of the environment. The state variable associated with any grid cell C_{ij} in the grid map yields the occupancy probability value of the corresponding region. Since the probabilities are identified based on the sensor data, they are purely conditional. Given a sensor data, each cell in the occupancy grid can be in two states $s(C_{ij})$= Occupied or $s(C_{ij})$= Empty, and to each cell there is probability $P[s(C_{ij})$ = Occupied$]$ attached, which reflects the belief of the cell C_{ij} being occupied by an object. Since

$$P[s(C_{ij}) = \text{Empty}] = 1 - P[s(C_{ij}) = \text{Occupied}] \quad (1)$$

Sensor readings supply uncertainty regions within which an obstacle is expected to be. The grid locations that fall within these regions of uncertainty have their values

increased while locations in the sensing path between the robot and the obstacle have their probabilities decreased.
For the purpose of this work, the histogram grid approach has been selected due to its simplicity, robustness and adaptability to dynamic environments. Like in certainty grid concept, each cell Cij in the histogram grid holds a certainty value, CVij, that represents the confidence of the algorithm in the existence Cij of an obstacle at the location pointed pout by i and j.

3.2 Ultrasonic sensor and map building

Map building methods depend strongly on the characteristics of the sensors that provide the raw data. In order to create a map using sensors, such as, ultrasonic range measurements, it is necessary to consider the following important steps (Thrun, 1998): sensor interpretations, integration over time, pose estimation, global grid building, and exploration. The probabilistic representation approach yields a world model, called a certainty grid that is especially suited to the unified representation of data from different sensors such as ultrasonic, vision, and proximity sensors as well as the accommodation of inaccurate sensor data (Moravec, 1988).
Ultrasonic sensors provide good range data but offer only poor directionality and associated with inaccuracy. A typical ultrasonic sensor returns a radial measure of the distance to the nearest object within its conical field of view (Polaroid, 1989), yet it does not specify the angular location of the object. The angle of the cone depends on the noise absorption of the material which causes the reflection. Typically, it lies between 10°to 30°.
The robot used in this research is a cylindrical in shape (the robot is 20 cm in radius), equipped with multiple simulated ultrasonic sensors (24 Polaroid ultrasonic ranging modules that are equally spaced (15 degree apart), arranged and mounted on a horizontal ring around the robot circumferences at a height of 30 cm above the ground), and has been designed, modeled and simulated by the author as a test bed for this work (see Fig. 1.b). Scanning with a horizontal ultrasonic ring does not require rotating parts and motors, and a view of 360^{o} coverage can be acquired rapidly. However, due to the possibility of a significant crosstalk, all sensors cannot be fired at the same time. Hence, there is a need to design a firing sequence that can reduce crosstalk (6 groups, each group with 4 sensors, and the sensors of each group are triggered simultaneously), but this will inevitably increase the overall time needed to obtain all 24 sensors readings for full 360^{o} coverage. Typical scan times range from 60 to 500 ms, for a full scan cycle (Crowley, 1989), and it is important to keep it as small as possible. Because of the wide beam angle, any individual range provides only indirect information about the location of the detected objects. Hence, the constraints from individual readings were combined to reduce the uncertainty.
The ultrasonic sensors were set up to detect an object within a range between 15 cm to 180 cm with accuracy of ±1%, and the beam of acoustic energy spreads in a cone of approximately 30°. Therefore, the longest time for each sensor, i.e. each sensors group, to wait for its echo is bounded by the maximum measurable distance, i.e., R =180 cm, and this time equal to $t = 2*1.8/340 = 10.58$ ms. A complete 360 scan should therefore take $6*10.58 = 63.48$ ms. Accordingly, If the sensor returns a distance value between the specified minimum and maximum range ($12 \leq r < 180$), then the returned distance measurement is proportional to the distance of nearest object located within the field of view of the sensor with small error along the arc perpendicular to the sensor acoustic axis. In this case, we have the range to the nearest point on the object but the sensor can't recognize the angle between

the sensor axis and the nearest point on the object. An object may be detected at a certain instance by one or more sensors simultaneously. An object located near the acoustic axis is more likely to produce an echo than an object further away from the acoustic axis. Whether or not an echo is received as an indication to the presence of an object depends on the relative angles, the surface structure, object reflect-ability, and the distance to the object.

Let's now consider the regions in the general sonar model. We can recognize four zones within the operational view in front of the sensor. These zones are named as Z1, Z2, Z3 and Z4. The areas of the four areas are superimposed over a grid of cells as shown in Fig. 1.c. The zone, Z1, no reading returned from anything in this region and this indicates that cells within this zone must be free from objects. The zone, Z2 (the area occupied by the arc at the radial distance equal to the measured distance (r)), indicates, for a given sensor range reading, the cells in which the nearest point on the nearest object that cause the measurement distance (r) is likely to be found, i.e., a reflection from an object laying in this region. In zones Z3 and Z4, the area of these zones is undetected by the current sonar reading. Our interest in this work is focusing on the area of Z1 and mainly the cells along the acoustic axis for each ultrasonic sensor. Given that the beam is superimposed on an occupancy grid, the sonar model looks similar to Fig. 1.d. The elevation indicates a high probability of the grid element being occupied and the depression signifies a high probability of being empty.

The sensor interpretation is the first phase that interprets sonar data for the purpose to perform the higher level mapping and navigation functions to support occupancy grid based navigation. The grids allow the efficient accumulation of small amounts of information from individual sensor readings for increasingly accurate maps of the robot's surroundings. During this process, the incoming 24 sonar range scalar values are interpreted and converted to local occupancy values around the robot. It should be noted that the range readings of each sensor group are asynchronously communicated to the map building algorithm. Since different sonar measurements give different values for a grid cell because of noise/error and the change in sensor's viewpoint, it is important to integrate the conditional probabilities of distinct moments. After the local grid is created around robot's current location, its values have to be merged into the global grid. Beyond the coordinate transformation between the grids, there is a need for a global position where the local grid can be integrated. Robot position can be estimated using position estimation method like "odometry", a continuous calculation of changes of the robot pose combined with correction of errors cumulated by sensor and motor noise (Moravec, 1988). This method is basically a data structure to store data acquired from different sources and takes into account the uncertainty of sensory data by working with probabilities or certainty values. The resulting probabilistic spatial models obtained from these sensor models serve as maps of the robot's environment, and can be used directly for robot navigation tasks such as path planning and obstacle avoidance, position estimation, and also to incorporate the motion uncertainty into the mapping process. It provides a way of representing data to be later extracted as useful and informative maps that can be used directly in robotic planning and navigation (Moravec, 1988). The data structure for each point in the environment represents the probability that some object occupies the point. In principle, uncertainty grid store qualitative information about which areas of the robot's surroundings are empty, and which areas are occupied by obstacles. Besides, no other characterization of the environment is of interest.

3.3 Building the grid map incrementally with robot in motion

There are different uncertainty calculi techniques to build occupancy grids of an unknown environment using sensory information provided by a ring of ultrasonic range finders. These techniques are based on Bayesian theory, Dempster–Shafer theory of evidence, fuzzy set theory, and the histogrammic in-motion mapping (HIMM) method (Murphy, 2000; Ribo & Pinz, 2001; Borenstein & Koren, 1991). Each of these methods has its own approach in transforming sensor readings into the representation of uncertainty and the update rule for combining the uncertainty for a pair of observation. It has been recognized that Dempster-Shafer method provides an accurate map compared to HIMM. However, the computational time required by HIMM method is significantly less than both Bayesian and Depmpster-Shafer methods. The method introduced by (Moravec & Elfes, 1985; Elfes, 1989a; Elfes, 1989b) projects a heuristic probability function that assign CVs to all cells in zone 1 (see Fig. 1.c), and this makes it computationally intensive and would impose a heavy time-penalty in relation to real-time execution, and due to this the mobile robot should remain stationary while taking a full scan with its 24 ultrasonic sensors. The histogram grid differs from the certainty grid in the way it is built and updated.

Let's focus on the incremental grid based map building approach using ultrasonic range measurements. The map of robot's environment has been constructed incrementally based on the histogrammic in-motion mapping approach as it is a good method for real-time map building with a mobile robot in motion, that is well suited to model inaccurate and noisy range sensor data and require minimum computational time. What makes it attractive is its capability to provide instantaneous environmental information that is ready to be used with integrated real-time obstacle avoidance. Each cell in the histogram grid holds a certainty value (CV) that represents the confidence in the existence of an obstacle at that location. Upon receiving a sonar range reading from each sensor within the available six sensor groups, the algorithm updates only the CVs of the cells that are directly a long the acoustic axis and within the sensor range reading. This approach results in a histogram probability distribution by continuously and rapidly sampling each sensor while the mobile robot is in motion. High certainty values are obtained in cells close to the actual location of the obstacle. For illustration the size of the environment has been considered as 10 meter x 10 meter, and the size of each cell has been decided to be as 10 x 10 cm. Each cell has its own coordinate and contains a CV. The CVs are updated by a heuristic probabilistic function that is applied to each one of the 24 range readings and takes into account the characteristics of a given sensor. The range of a CV has chosen arbitrary from 0 (the minimum CV of a cell) to 16 (the maximum CV of a cell) to represent the confidence in the existence of an obstacle. The CV=16 shows, any cell with this value represents high probability to be occupied by obstacle or object at that instance. While the CV=0 indicates that any cell with this value represents high probability to be empty. Sensor position is determined relative to robot reference frame. Sensor readings are mapped into CVs according to specific rules that produce high CVs for cells that correspond to obstacles, while it keeps low CVs for empty cells or for cells that were incremented due to misreading or moving objects.

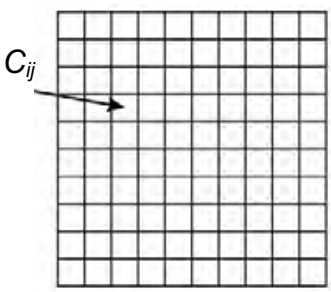
The update rule for any cell's CV consists of the following steps:

1. At the beginning, it starts by having the certainty values of all cells representing the environment assigned to zero, i.e., all cells are assigned to be free from obstacles.
2. The robot starts at an unknown position and obtain measurements of the

environment relative to its momentary position.

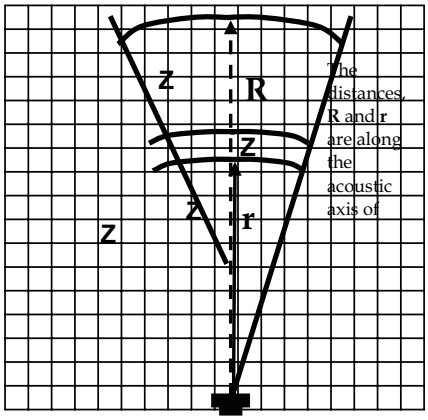
3. For each sensor group, sensor readings are communicated to the map building module. This information is used to build and maintain incrementally a map that can support navigation and localization.
4. For each sensor range reading, the algorithm updates the CVs of the cells that are only located along the acoustic axis of the sensor and at the radial distance within the view of the sensor. The update rules are as follow:
   a) The certainty value (CV) of the cell on the acoustic axis and corresponds to the measured distance (r) is updated by adding 4 to it and the CV is bounded by its maximum value, i.e., only one cell is incremented for each range reading. A large increment makes the robot reacts to a single and possibly false readings, while a small increment would not build up timely CVs to support real-time avoidance maneuver.
   b) The CVs of other cells that lie along the acoustic axis and at a range less than the sensor range reading are decremented. The CVs of the cells are decremented according to the following:
      I. If the range reading is, $r < 60$, then decrement the CVs of the cells that are at a distance less than the range reading by 1, i.e., CV -= 1.
      II. If the range reading is, $60 \le r < 120$, then decrement the CVs of the cells that are within 60 cm distance along the acoustic axis by 2 and decrement the other cells that are with the distance between 60 cm and less than 120 cm by 1.
      III. If the range reading is, $120 \le r < 180$, then decrement the CVs of the cells that are within 60 cm distance along the acoustic axis by 3, decrement the cells within the distance $60 \le r < 120$, by 2, and the remaining cells within the distance between 120 cm and less than 180 cm by 1.
   c) When there is no echo reflecting back, i.e., no obstacle within the view of the sensor, i.e., the sensor reading is $r = R$. In this case, there is no need for any increment to the CV of the cell corresponds to the sensor range reading, and the decrement will follow the rule mentioned in 4.b.III.

A histogram type probability distribution is obtained by continuous and rapid sampling of the sensors whiles the robot in motion (see Fig. 2.a and b for illustration). A global path-planning and obstacle avoidance strategies can be employed iteratively as the map is incrementally updated.

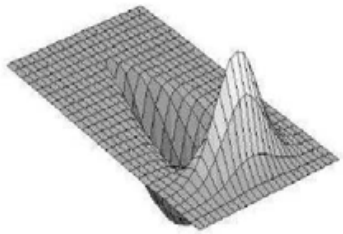


(a)An example of occupancy grid structure.

(b) The designed mobile robot as a test-bed for simulation



(c) The classification of regions in the sonar model



(d) Visualization of the probabilities distribution

Fig. 1. The model of the used mobile robot and the regions in the sonar model

(a) Example of a robot environment with a robot inside it.

(b) Incrementally constructed the histogram grid-based map.

Fig. 2 Example of a robot environment and a histogram grid-based map building.

4. Dynamic Navigation Supporting Obstacle Avoidance

Different obstacle avoidance techniques have been developed and available in the literature. Some of the relevant obstacle avoidance techniques are based on edge detection, potential field and certainty grids. The edge detection algorithm tries to determine the position of the vertical edges of a detected obstacle using ultrasonic sensors, in order to enable the robot to steer around the obstacle (Kuc & Barshan, 1989). A disadvantage of this method is the need for the robot to remain stationary while gathering sensor information about the obstacle. In addition, this method is very sensitive to sensor accuracy. As for the certainty grid for obstacle detection and avoidance, a method for probabilistic representation of obstacles in a certainty grid-type world model has been developed (Moravec & Elfes, 1985, Elfes, 1987; Moravec, 1988). This approach is especially suited to accommodate inaccurate sensor data

such as range measurements from ultrasonic sensors. With this approach too, the mobile robot has to remain stationary while it takes a panoramic scan with its 24 ultrasonic sensors. After updating the certainty value of the relevant grid cells, the robot moves to a new location, stops, and repeats the procedure. After having robot traverses its environment and having its map, a global path-planning method is then employed for off-line calculations of subsequent robot paths. The critical issue with this method is computational cost. In addition, there are widely used methods for obstacle avoidance that are based on the concept of potential field. The idea of imaginary forces acting on a robot has been suggested by (Khatib, 1983) in which obstacles exert repulsive forces while the target applies an attractive force to the robot, and accordingly a resultant force vector comprising the sum of a target-directed attractive force and repulsive forces from obstacles, is calculated for a given robot position. With the resultant force vector acting as accelerating force on the robot, the robot's new position for a given time interval is calculated, and the algorithm is repeated. Krogh and Thorpe (Krogh & Thorpe, 1986) suggest a combined method for global and local path planning, which uses a generalized potential field approach. This method assumes a known and prescribed world model, in which simple, predefined geometric shapes represent obstacles and the robot's path is generated off-line.

But, the need is to have a method that deals with real-time sensory information and allows for fast, continuous, and smooth motion of the controlled vehicle among unexpected obstacles, and does not require the vehicle to stop in front of obstacles. Obstacle avoidance methods must account for the sensors' shortcomings, such as (for the case of ultrasonic sensors) inaccuracies, crosstalk, and spurious readings. The adopted approach in this work used the extended vector field histogram with obstacle avoidance technique (Borenstien & Ulrich, 1998) that uses in real-time as input an intermediate data structure as a 2D histogram grid representing robot's local environment that is based on occupancy grid concept. This approach was further developed for real time mobile robot obstacle avoidance to offer smoother robot trajectories and better reliability.

This approach employs the following steps:

1. The two-dimensional map grid is reduced to one-dimensional polar histogram that is constructed around the robot's momentary location. This includes:

   (a) Select a circular masking window of a certain diameter and attach it to the robot reference frame at the robot centre point. This window moves together with the robot. The area covered by this window at any time is called the active region.

   (b) Based on the current map information, the content of every cell covered by the circular masking window is represented by an obstacle vector with a direction described by the robot's current position and the relevant cell position. The magnitude is proportional to the squared value of CV of that cell, and it is a function of the squared distance between that cell and the current position of the robot. Occupied cells produce larger vector magnitude when they are close to the robot. According to the adopted axes and angle convention (Fig. 3.b (upper)), the direction of the obstacle vector is $\tan^{-1}[(x_i - x_0)/(y_j - y_0)]$ and the final value is calculated using the signs of both arguments to determine the quadrant of the return value. x_0, y_0, represent the coordinates of the robot's current position at its center point, and x_i, y_j represents the coordinates of a cell C_{ij} within the circular mask window. Global coordinates is used to describe robot position.

(c) The circumference of the circular mask window is divided into *m* angular sectors. The resolution for each sector has been selected to be 5°, i.e., the total number of sectors is, $n = 360^{\circ}/5^{\circ} = 72$, $S_n, n = 1, 2, .., 72$ (See Fig. 3.c). It is possible to have a finer resolution for the sectors, but this will result in extra computation cost. Each sector is represented by a value that reflects the confidence in the existence of object/obstacle within it, and an obstacle density describes it. This value is calculated by summing up the obstacle vector magnitudes for all cells within each sector to yield the confidence value of the corresponding sector.

(d) Map the confidence value of each sector by using its angular position ($5 \times n$) onto the primary polar histogram.

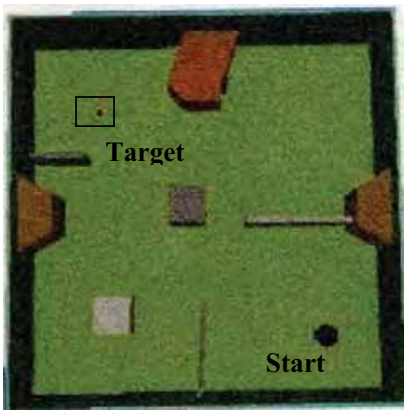
(e) Build the polar histogram around the current position of the robot using the adopted axes and angle convention, and it is independent of robot's orientation.

A polar histogram with robot's width consideration is the output from this step.

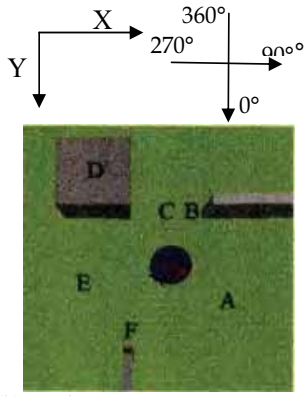
2. Using the results of the previous step to generate a binary polar histogram. This is performed by using a hysteresis technique based on two thresholds values, high and low, (these values are decided experimentally). The two threshold values help to build a binary polar histogram and help to identify sectors within the binary polar histogram that are free from objects or blocked, and if it is free, whether it has a wide or a narrow opening that allow the robot to pass through. In addition, the two threshold values help to smooth the desired trajectory and avoid oscillations in the steering command. The binary polar histogram is updated in real-time.
3. Select the most appropriate candidate as new direction for the robot based on the generated binary polar histogram, the give target, and a cost function. This selection aims to enable the robot to achieve the desired target through simultaneous translation and rotation.

The navigation system continuously reads sensory input and writing motor commands at regular intervals. The sensory input comprises the readings of the ultrasonic and odometry sensors. The motor commands govern information about the environment that is stored in the internal map. To navigate reliably, a mobile robot must know its pose (position and orientation) and this can be done by estimating robot's pose relative to its environment. A kinematic model of the designed differential drive robot is used for simulation based experiments to track robot position and orientation. Other parameters can be considered to increase accuracy, such as wheel radius and slip angle.

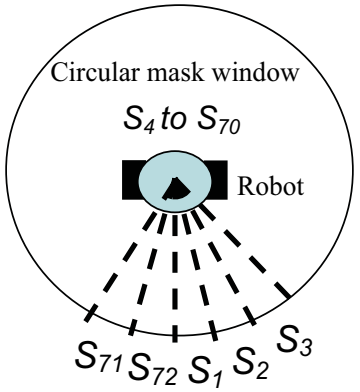
This approach has been combined with the mapping algorithm discussed in section 3, implemented and demonstrated. Figure 3.a shows an example of a selected robot environment with a target location framed by dark lines and the robot is shown in a dark spot, and figure 3.b shows a momentary situation faced by the robot within the selected environment. The primary polar histogram for this situation is shown in Figure 3.d. For the sectors represented by 'A' and 'E', the obstacle density is very small. For the sectors represented by 'B' and 'F', the obstacle density is high. The sectors covered by 'D' posse medium value of the obstacle density as the obstacle at 'D' has just been detected, and hence its CV is still small. Although sectors represented by 'C' is free of obstacles, it doesn't have a small obstacle density value, and this is due to the poor angular resolution of the sonar which restricts the detection of small opening. Figure 3.e shows the binary polar histogram for the considered momentary situation.



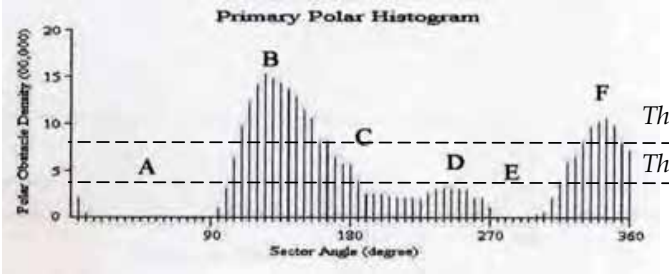
(a) A selected robot environment populated with objects



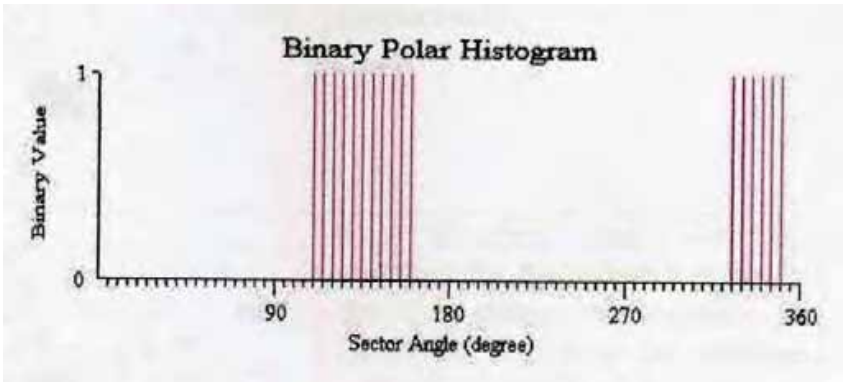
(b) A robot's momentary situation



(c) Circular mask window and the divided sectors.



(d) Polar histogram and the two threshold parameters



(e) Binary polar histogram.

Fig. 3. Obstacle avoidance example with highlight for a robot's momentary location.

5. Conclusion and Future Work

This chapter presented the state of the art, the complexity, and the challenges facing simultaneous and real-time mapping, obstacle avoidance, and dynamic navigation for mobile robots. The development of an autonomous map building and maintenance method with focus on incremental map building technique that is suitable for real-time obstacle detection and avoidance has been presented. The chapter introduced a modified histogram grid based navigation techniques that is integrated with the concept of vector field histogram to support real-time obstacle avoidance in parallel with the incremental map building. Through the integrated implementation, the mobile robot was able to map the environment, avoid obstacles and move towards its target successfully. The results are important for simultaneous localization and map building applications, and can facilitate the use of additional natural landmarks to improve the accuracy of the localization algorithm.

5.1 Future work

The future expansion can include the following:

1. As the grid based mapping in general and the histogram grids require fixed-size environment, it is important to consider a technique that helps to deal with dynamic size of robot's environment, i.e. building variable-sized maps without imposing size or shape constraints on the grid map. Dynamically expanding occupancy grids seek to remove dependency on an array for representing a histogram or occupancy grid. Dynamically expanding occupancy grids tackle the issues related to the efficient use of available memory and the requirements for real time searching and actions. Such techniques help to increase the size of map dynamically when new area is detected.
2. Integrate heterogeneous sensor modules, such as ultrasonic range data and an

active stereo-vision system or other type of sensors. Due to the complementary error characteristics with respect to range and angular resolution, fusion of stereo-vision data with ultrasonic range information improves mapping precision significantly. In addition, the vision can support the classification of dynamic and modeling of obstacles in 3D,

3. Consider building 3D based grid map,
4. It is essential to develop a powerful incremental integration between the geometric and topological mapping approaches supported by belief values. This should have simultaneous support for localization, path planning and navigation,
5. Multi-robot sharing map building. Merging accurately topological maps, or metric maps or hybrid maps created by different mobile robots. In addition, the key challenge here is, how representation (i.e., any form of world model and mapping) can be effectively distributed over the behavior structure?
6. Due to the limitations associated with grid and topological based mapping, it is necessary to find new techniques to efficiently integrate both paradigms.

6. References

Bailey, T.; Nieto, J. & Nebot, E. (2006). Consistency of the FastSLAM algorithm. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'06), May 2006, pp. 424-429.

Borenstien, J. & Koren, Y. (1991a). Histogrammic In-Motion Mapping for Mobile Robot Obstacle Avoidance, *IEEE Transaction on Robotics and Automation*, Vol. 7, No. 4, pp. 535-539, 1991.

Borenstein, J. & Koren, Y. (1991b). The Vector Field Histogram: Fast Obstacle Avoidance for Mobile Robots, *IEEE Journal of Robotics and Automation*, Vol. 7, No. 3, June 1991, pp. 278-288.

Borenstien, J & Ulrich, I. (1998). VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots, *IEEE International Conference on Robotics and Automation (ICRA'98)*, Leuven-Belgium, pp. 1572-1577, 1998.

Borghi, G. & Brugali, D. (1995). Autonomous Map Learning For A Multisensor Mobile Robot Using Diktiometric Representation And Negotiation Mechanism, Proceedings International Conference on Advanced Robotics, Spain, September 1995, pp. 1-8.

Brooks, A. (1983). Solving the Find Path Problem by Good representation of free space, *IEEE Transaction on Systems, Man and Cybernatics*, Vol.13, No. 2, March 1983. pp. 190-197.

Burgard, W.; Cremers, A.B.; Fox, D.; Hahnel, D.H.; Lakemeyer, G.; Schulz, D.; Steiner, W. & Thrun, S. (1999). Experiences with an Interactive Museum Tour-Guide Robot. *Artificial Intelligenc*e, Vol. 114, No. 1-2, pp. 3–55, 1999.

Castellanos, A. & Tardos, J.D. (1999). Mobile Robot Localization and Map Building: A Multisensor Fusion Approach, Kluwer cademic Publishers, Boston, MA, 1999.

Chatila, R. & Laumond. J.-P. (1985). Position Referencing and Consistent World Modeling for Mobile Robots. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automatio*n ICRA'1985, March 1985, pp. 138 – 145 vol. 2.

Choset, H. (1996). Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph. PhD thesis, California Institute of Technology, 1996.

Choset, H. & Nagatani, K. (2001). Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without, *IEEE Transactions on Robotics and*

Automation, Vol. 17, No. 2, April 2001, pp. 125-137.

Crowley, J.L. (1989). World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'89)*, Scottsdale, Arizona, May 1989, pp. 674-680.

Durrant-Whyte H.F. (1998). Integration, Coordination and Control of Multi-Sensor Robot Systems, *Kluwer Academic Publishers, 1988.*

Elfes, A. E. (1987). Sonar-based Real-World Mapping and Navigation, *IEEE Journal of Robotics and Automation*, Vol. RA-3, No 3, June 1987, pp. 249-265.

Elfes, A. E. (1989a). Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

Elfes, A. E. (1989b). Using Occupancy Grids for Mobile Robot Perception and Navigation, *Computer Magazine*, , Vol. 22, No. 6, June 1989, pp. 46-57.

Ellore, B.K. (2002). Dynamically Expanding Occupancy Grids, M. Sc. Thesis, Computer Science, Texas Technical University, 2002.

Engelson, S.P. & McDermott, D.V. (1992). Error Correction in Mobile Robot Map Learning, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'92)*, Nice-France, 1992, pp. 2555-2560.

Fabrizi, E. & Saffiotti, A. (2000). Extracting Topology-Based Maps from Grid maps, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2000)*, pp. 2972-2978, 2000.

Habib, M.K. & Yuta, S. (1988). Map Representation and Path Planning for Mobile Robots in a Building Environment, *IEEE International Conference on Intelligent Robots and Systems (IROS'88)*, Tokyo, Oct. 1988, pp.173-178.

Habib, M.K. & Yuta, S. (1990). Efficient On-line Path Planning Algorithm and Navigation for a mobile robot, *International Journal of Electronics*, Vol. 69, No. 2, 1990, pp. 187-210.

Habib, M.K. & Yuta, S. (1993). Map Representation for Large In-door Environment, Path Planning and Navigation Techniques for an Autonomous mobile robot with its Implementation, *International Journal of Automation in Construction*, Vol.1, No.2, 1993, pp.155-179.

Hahnel, D.; Triebel, R.; Burgard, W. &Thrun, S. (2003). Map building with mobile robots in dynamic environments. Proceedings of the IEEE International Conference on Robotics and Automation(ICRA'03), Sept. 2003, pp. 1557- 563 vol.2.

Howard, A. & L. Kitchen (1997, March). Sonar mapping for mobile robots. Technical Report 96/34, Department of Computer Science, University of Melbourne.

Khatib, O. (1985). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, *Proceedings of the IEEE International Conference on Robotics and Automation(ICRA'85)* , St. Louis, Missouri, March 1985, pp. 500-505.

Koren, Y. & J. Borenstein (1991). Potential eld methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA'91)*, pp. 1398–1404.

Krogh, B. H. & Thorpe, C. E. (1986). Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'86)*, San Francisco, California, April, 1986, pp. 1664-1669.

Kuc, R. & Barshan, B. (1989). Navigating Vehicles Through an Unstructured Environment With Sonar, *Proceedings of the IEEE International Conference on Robotics and*

Automation (ICRA'89), Scottsdale, Arizona, May 14-19, 1989, pp. 1422-1426.

Kuipers, B. & Byun, Y.T. (1991). A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations, *Robotics and Autonomous Systems*, 8, pp. 47-63, 1991.

Kuipers, B.; Modayil, J.; Beeson, P.; MacMahon, M. & Savelli F. (2004). Local metrical and global topological maps in the Hybrid Spatial Semantic Hierarchy. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04), April-May 2004, pp. 4845-4851 Vol.5.

Lanthier, M., D.Nussbaum, and A.Sheng (2004, August). Improving Vision-Based Maps by using sonar and infrared data. submitted to Robotics and Applications, IASTED 2004.

Levitt, T.S. & Lawton, D.T. (1990). Qualitative navigation for Mobile Robots, *Artificial Intelligence*, Vol. 44, No. 3, pp. 305-360.

Lisien, B.; Morales, D.; Silver, D.; Kantor, G.; Rekleitis, I. & Choset, H. (2003). Hierarchical Simultaneous Localization and Mapping. Proceedings of IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS'03), Las Vegas, Nevada, October 2003, pp. 448-453.

Lu, F. & Milios, E. (1997). Globally Consistent Range Scan Alignment for Environment Mapping, *Autonomous Robots*, Vol.4, N. 4, pp. 333–349, October 1997.

Martin, M. & H. Moravec (1996). Robot Evidence Grids, *Technical report CMU-RI-TR-96-06*, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

Martinelli, A.; N. Tomatics, N. & Siegwart, R. (2004). Open challenges in SLAM: An optimal solution based on shift and rotation invariants. In Proceedings IEEE International Conference on Robotics and Automation {ICRA'04), 2004, pp. 1327-1332.

Mataric, M. J. (1990). A distributed model for mobile robot environment-learning and navigation, Master thesis, MIT, Cambridge, MA, January 1990. Also available as MIT AI Lab Tech Report AITR-1228.

Montemerlo, M. & Thrun, S. (2003). Simultaneous Localization and Mapping with Unknown Data Association Using FastSLAM, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2003)*, September 2003, pp. 1985 - 1991 vol.2.

Montemerlo, M. & Thrun, S. (2006). FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics. Springer Tracts in Advanced Robotics, 2007.

Moravec, H. P. & Elfes, A. E. (1985), High Resolution Maps from Wide angle Sonar, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'85)*, St. Louis, Missouri, March 1985, pp. 116-121.

Moravec, H. P. (1988). Sensor Fusion in Certainty Grids for Mobile Robots, *AI Magazine*, Vol. 9, No. 2, Summer 1988, pp. 61-74.

Moravec, H. P. (1996, September). Robot spatial perception by stereoscopic vision and 3d evidence grids. *Technical report,* Carnegie Mellon University.

Murphy, R.R. (2000). Introduction to AI Robotics", The MIT Press, *Massachusetts Institute of Technology*, MA-USA, 2000.

Preciado A., Meizel D.; Segovia, A. & Rombaut, M (1991). Fusion of Multi-Sensor Data: a Geometric Approach, *Prorоceedings of the IEEE International Conference on Robotics and Automation (ICRA'1991).* Sacramento, California, April 1991, pp. 2806-2811.

Ribo, M. & Pinz, A. (2001). A Comparison of three Uncertainty Calculi for Building Sonar-Based Occupancy Grids, *The Journal of Robotics and Autonomous Systems,* 35, pp. 201-209, 2001.

Polaroid Corporation, Ultrasonic Components Group, 119 Windsor Street, Cambridge, Massachusetts 02139, 1989.

Se, S.; & D. Lowe, D. & Little, J. (2002). Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks, *The International Journal of Robotics Research,* Vol. 21, No. 8, August 2002, pp. 735-758.

Singhal, A. (1997). Issues in Autonomous Mobile Robot Navigation, Report, Computer Science Department, University of Rochester, 1997.

Stachniss, C. & Burgard, W. (2005). Mobile Robot Mapping and Localization in Non-Static Environments. In Proc. of the National Conference on Artificial Intelligence (AAAI), Pittsburgh, PA, USA, 2005, pp. 1324-1329.

Szab, R. (2004). Topological Navigation of Simulated Robots Using Occupancy Grid, *International Journal of Advanced Robotic Systems*, Vol. 1, No. 3, 2004, pp. 201-206, ISSN 1729-8806

Thrun, S. & Bucken, A. (1996). Learning Maps for Indoor Mobile Robot Navigation, Technical Report CMU-CS-96-121, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213, 1996.

Thrun, S. (1998). Learning Metric-Topological Maps for Indoor Mobile Robot Navigation, *Artificial Intelligence*, Vol. 99, No. 1, pp. 21-71, 1998.

Thrun, S. (2001). A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots", *International Journal of Robotics Research*, Vol.20. No.5, pp. 335–363, 2001.

Thrun, S. (2002). Robotic Mapping: A Survey, CMU-CS-02-111 report, Feb. 2002.

Tomatis, N.; Nourbakhsh, I.; Arras, K. & Siegwart, R. (2001). A Hybrid Approach for Robust and Precise Mobile Robot Navigation with Compact Environment Modeling, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2001)*, Seoul-Korea, May 2001, pp. 1111-1116.

Wang, C.-C., & Thorpe, C. (2002). Simultaneous localization and mapping with detection and tracking of moving objects. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2002), Sep. 2002, pp. 842-849 vol.1.

Wang, Z.; Huang, S. & Dissanayake, G. (2005). Decoupling localization and mapping in SLAM using compact relative maps. In Proceedings of IEEE/JRS Intl. Conf. on Intelligent Robotics and Systems (IROS'5), Edmundton, Canada, Aug 2005, pp. 1041-1046.

Yamauchi, B. & Langley, P. (2000). Place Recognition in Dynamic Environments. *Journal of Robotic Systems*, Vol. 14, No. 2, pp. 107–120.

Youngblood, G. M.; Holder, L. B. & Cook, D. J. (2000). A Framework for Autonomous Mobile Robot Exploration and Map Learning Through the Use of Place-Centric Occupancy, *ICML Workshop on Machine Learning of Spatial Knowledge.*

27

Intelligent Control of AC Induction Motors

Hosein Marzi
Department of Information Systems, St. Francis Xavier University
Canada

1. Introduction

It has been proven that fuzzy controllers are capable of controlling non-linear systems where it is cumbersome to develop conventional controllers based on mathematical modeling. This chapter describes designing fuzzy controllers for an AC motor run mechanism. It also compares performance of two controllers designed based on Mamdani and Takagi-Sugeno with the conventional control scheme in a short track length, following a high disturbance. Fine and rapid control of AC motors have been a challenge and the main obstacle in gaining popularity in use of AC motors in robots actuators. This chapter reviews how use of intelligent control scheme can help to solve this problem.

2. Cart and Pendulum Problem

Design and implementation of a system is followed by vigorous testing to examine the quality of the design. This is true in the case of designing control systems. One the classical systems to test quality and robustness of control scheme is inverted pendulum. In recent years, the mechanism of an inverted pendulum on a moving cart has been used extensively and in many different types. The cart and pendulum mechanism has become even more popular since the advent of intelligent control techniques. This mechanism is simple, understandable in operation, and stimulating. It has a non-linear model that can be transformed into linear by including certain condition and assumption in its operation. For the above reasons, inverted pendulum's performance has become a bench mark for testing novel control schemes. In this chapter the focus is on the driving power in balancing the inverted pendulum which is an electrical motor. Traditionally, DC motors are used for this type of tasks. However, in this chapter the focus is on AC electrical motors for producing the torque required for the horizontal movements of the inverted pendulum. A simplified control model for the AC motor is used which includes the motor's equivalent time constant as the crucial parameter in producing rapid responses to the disturbances. In the modeling of fuzzy controllers for the inverted pendulum, the input to the pendulum block is considered to be a torque. This torque is produced by an electrical motor which is not included in the model. That is, the torque is output of the motor. A disadvantage in this modeling is that the electrical motor dynamics is not built-in in the control system independently. On the other hand, not including the electrical motor in the control scheme of the pendulum mechanism provides the freedom to alter the electrical motor and examine the performance of the pendulum with different types of the drive. Here, a simplified model of an AC electrical motor is incorporated into the system. The electrical motor receives its

inputs as current or voltage and produces a torque as output to control the balance of the mechanism.
The new approach in modeling a fuzzy control system assists in achieving a number of goals such as: examining use of AC motors in producing rapid response, selecting sensitive parameters for an optimum high performance electrical motor capable to stabilize the inverted pendulum system, designing a Takagi-Sugeno type fuzzy controller, and comparing the effectiveness of inclusion of fuzzy controller along with the conventional control scheme.

3. Conventional Controllers & Fuzzy Controllers

The conventional approach in controlling the inverted pendulum system is to use a PID (Proportional, Integral, and Derivative) controller. In order to model the system the developer would have to know every technical detail about the system and be able to model it mathematically. Fuzzy Logic control (FLC) challenges this traditional approach by using educated guesses about the system to control it (Layne & Passino 2001). Passino states that differential equations are the language of conventional control (PID), while "rules" about how the system works is the language of fuzzy control (Passino and Yurkovich, 1998).
Fuzzy logic has found its way into the everyday life of people, since Lotfi Zedah first introduced fuzzy logic in 1962. In Japan, the use of fuzzy logic in household appliances is common. Fuzzy logic can be found in such common household products as video cameras, rice cookers and washing machines (Jenson 2005). From the weight of the clothes, fuzzy logic would be able to determine how much water as well as the time needed to effectively wash the clothes. Japan developed one of the largest fuzzy logic projects, when they opened the Sendai Subway in 1987 (Kahaner 1993). In this subway, trains are controlled by fuzzy logic. Fuzzy Logic is a subset of traditional Boolean logic. Boolean logic states that something is either true or false, on or off, 0 or 1. Fuzzy logic extends this into saying that something is somewhat true, or not completely false. In fuzzy logic there is no clear definition as to what is exactly true or false. Fuzzy logic uses a degree of membership (DOM) to generalize the inputs and outputs of the system (Lin and Lee 1996). The DOM ranges from [0 1], where the degree of membership can lie anywhere in between.
The majority of Inverted pendulum systems developed using fuzzy logic, are developed using a two dimensional approach, where only the angle and angular velocity of the pendulum's arm are measured. The following research will show why this method is insufficient for the development of an inverted pendulum on a limited size track. To have an efficient fuzzy controller for an inverted pendulum, the system must also include inputs for the position of the cart that the pendulum is balanced upon and the velocity of the cart. Two-dimensional fuzzy controllers are very simple examples of fuzzy control research. Many of them will balance the inverted pendulum, but are not in control of the cart's position on the track. Adeel Nafis proposed a two-dimensional fuzzy controller to balance the Inverted pendulum on a track (Nafis 2005). Tests showed that the controller would balance the pendulum but neglected to control the position of the cart and eventually the cart's position would exceed the length of the track. Another FLC was proposed by Passino; again this cart had the same result as the previous FLC (Passino and Yurkovich, 1998).
Control of the system requires that the cart holding the pendulum be moved by some mechanism. For simulation purposes, in this experiment a field oriented AC motor was used (Bose 1997).

4. Effect of Number of Inputs on Designing Fuzzy Logic Controllers

In a simple control mechanism there is one input and one output. Fuzzy Logic Controllers can have more than one input. Two-input FLC's are easy to implement and receive great performance responses from simulations. Layne (Layne & Passino 2001) modeled a fuzzy controller that had great performance balancing the pendulum but the cart's positioning was unstable, making it an impractical rule set for real life implementation. Two-input FLC's are the most commonly researched inverted pendulum systems. One of the most commonly researched types fuzzy controllers is two-input inverted pendulum systems. The 2-input system receives angle θ and angular velocity ω as its inputs. The system uses 5 membership functions for each input, and another 5 for the outputs which is the Force. The system consists of 25 (that is 5 to power 2; 52) rules. Table 1 shows the rule base for the inverted pendulum system. According to Table 1 a value of NL represents a negative large angle or angular velocity, and PL represents a positive large angle/angular velocity. As Table 1 indicates, if there is a situation where the angle is Zero (ZE) and the angular velocity is PS then the rule NS will be fired. Where, NL, NS, ZE, PS, PL are linguistic values of negative large, negative small, zero, Positive small, and positive large.

θ/ω	NL	NS	ZE	PS	PL
NL	PL	PL	PL	PS	ZE
NS	PL	PL	PS	ZE	NS
ZE	PL	PS	ZE	NS	NL
PS	PS	ZE	NS	NL	NL
PL	ZE	NS	NL	NL	NL

Table 1. Rule-base Matrix for the Inverted Pendulum.

A simulation that runs for 2 seconds is shown in Figure 1. The pendulum has an initial angle of 0.2 radians (dashed line). When the simulation is run, the angle of pendulum balances quickly, in about 1 second, but the position of the cart is not controlled (continuous line) so the cart's position will eventually drift off into the end of the track, even though the pendulum's arm is balanced.

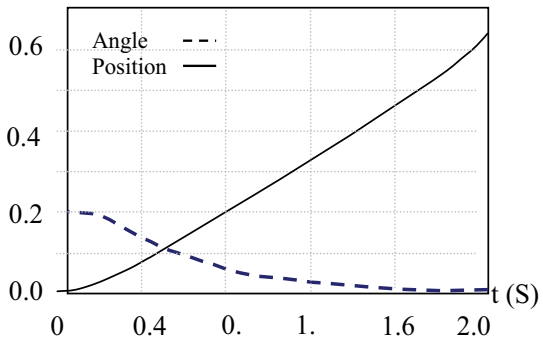


Figure 1. Variation of angle θ (rad) and position X (m) of pendulum vs. time t (s).

The benefit of adding two more inputs to the system to control the X-position of the cart and the velocity of the cart will greatly benefit the stability of the system. There is a cost for better stability; this is a greater computation time, and greater complexity in the model. The cost of adding more inputs increases exponentially with the number of inputs added. The above two-input system used five membership function for each input used; this resulted in a 25 (i.e. 52) rule base. By adding two more inputs to the system, the systems rule base would grow to 625 (i.e. 54) rules. Development time for a rule base this size can be very time consuming, both in development and in computational time. Bush proposed using an equation to calculate the rules, rather than taking the time to develop the rules individually (Bush 2001). The system was a 54 system with 17 output membership functions (OMF). The equation used was:

$$\text{Output Membership Function} = I + (J - 1) + (-K + 5) + (L+5) \quad (1)$$

This equation results in values ranging between 1 and 17. This corresponds to the OMF that is to be used in the calculation of the output. The performance of the system using this approach is not consistent with that of the original simulation, given by the author of the above Equation 1 (Bush 2001). The force given to the cart holding the pendulum was found not to be enough to balance the pendulum and the system failed within a small amount of time. It can be concluded that this system would be a good starting point for one to base a large rule set on, but the system would need some tweaking of the rules and membership functions to get to balance the system effectively. The final FLC controller that was modeled for simulation was a Takagi-Sugeno type fuzzy controller. All the previous FLC's modeled were of Mamdani type. A Takagi-Sugeno type fuzzy controller (Mathswork, 2002), (Liang & Langari, 1995), (Johansen et al. 2000), (Tanaka et al. 2003) varies from the traditional Mamdani type controller by using linear or constant OMF's instead of triangular, trapezoidal, Gaussian or any other method the developer decided to use. The system uses 4-inputs with only 2 input membership functions for each. This resulted in a 24, 16 rule system. The linear output membership functions are calculate using the equation

$$\text{Output Membership Function} = c_0 + (c_1 * x_1) + (c_2 * x_2) + (c_3 * x_3) + (c_4 * x_4) \quad (2)$$

Where cn is the parameters of the OMF, and xn is the values of θ, ω, X and linear velocity V respectively. The system modeled here uses fuzzy logic toolbox of Matlab (Sugeno 2002).

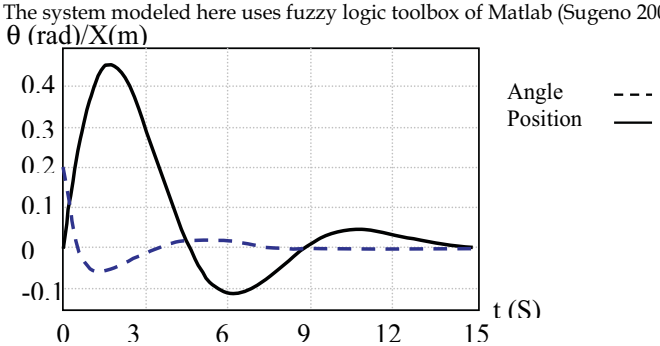


Figure 2. In adjusting the balance of pendulum angle θ (rad), and position X (m) changes with time t (s).

The control of all 4 parameters with only 2 membership functions causes the system to run very quickly. The down side to this quick response is that it takes more time for the system to stabilize when there are so few membership functions. The system will overshoot the targeted position and eventually come to rest. The settling time of this system takes more time than any other system.
Figure 2 is the result of the simulation. The pendulum is started with an initial disturbance of 0.2 radians. As shown, the fuzzy controller overcompensates for this initial disturbance and sends the pendulum's angle (dashed line) in an opposite direction in an attempt to balance it, this is the overshoot. It takes approximately 5 seconds for the pendulums arm to balance.

5. Mathematical Modeling of Field Oriented AC Induction Motors

The motor chosen for the simulation is an AC motor. The motor is modeled, Figure 3, using field oriented control scheme (Bose 1997).

$$V_{qs} = R_s i_{qs} + \frac{d}{dt}\varphi_{qs} + \omega_e \varphi_{ds} \tag{3}$$

$$V_{ds} = R_s i_{ds} + \frac{d}{dt}\varphi_{ds} - \omega_e \varphi_{qs} \tag{4}$$

$$0 = R_s i_{qr} + \frac{d}{dt}\varphi_{qr} + (\omega_e - \omega_r)\varphi_{dr} \tag{5}$$

$$0 = R_r i_{dr} + \frac{d}{dt}\varphi_{dr} + (\omega_e - \omega_r)\varphi_{qr} \tag{6}$$

$$T_e = 1.5p\frac{L_m}{L_r}(\varphi_{dr} i_{qs} - \varphi_{qr} i_{ds}) \tag{7}$$

$$\begin{cases} \varphi_{qs} = L_s i_{qs} + L_m i_{qr} \\ \varphi_{ds} = L_s i_{ds} + L_m i_{dr} \end{cases} \tag{8}$$

$$\begin{cases} \varphi_{qr} = L_r i_{qr} + L_m i_{qs} \\ \varphi_{dr} = L_r i_{dr} + L_m i_{ds} \end{cases} \tag{9}$$

$$\varphi_{qr} = 0 \Rightarrow \frac{d}{dt}\varphi_{qr} = 0 \Rightarrow \varphi_{dr} = \varphi_r \tag{10}$$

$$\omega_{sl} = (\omega_e - \omega_r) = \left(\frac{L_m R_r}{\varphi_r L_r}\right) i_{qs} \tag{11}$$

$$T_e = 1.5p\frac{L_m}{L_r}(\varphi_r i_{qs}) \tag{12}$$

$$\varphi_r + \tau_r \frac{d}{dt}\varphi_r - L_m i_{ds} = 0 \tag{13}$$

Where: $\tau_r = L_r / R_r$ is the rotor time constant.

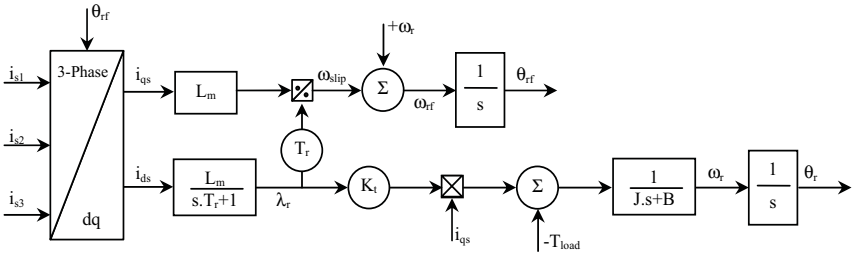


Figure 3. Magnetic Flux Control Scheme in Induction Motors

The field-oriented scheme makes control of AC machine analogous to that of DC machine. This is achieved by considering the d-q model of the AC machine in the reference frame rotating at synchronous speed ωe. In this model ids and iqs are current components of the stator current on d-q axis, where ids component is aligned with the rotor field. The rotor flux and torque can be controlled independently by ids and iqs, shown in Figure 4. The electric torque Te is proportional to the quadrature-axis current iqs, component of the stator current Is, and the rotor flux ψr can be controlled by the direct-axis current ids, of Is, where: Is = ids + J iqs.

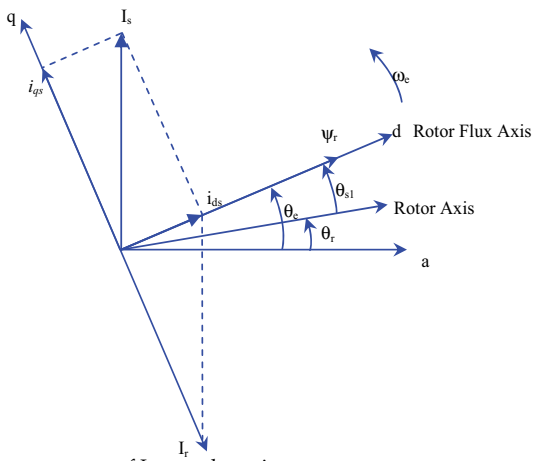


Fig 4. iqs and ids components of Is on a d-q axis

The transfer function of this AC motor yields angular velocity (ω) as the motor shaft output. In the simulation, ω was easily converted into the force on the cart. The motor responded well, reaching its maximum force exerted on the cart in less than 2.5 seconds.

6. Discussion and Results

The simulation consists of four main components, the fuzzy controller, AC motor, the cart and the inverted pendulum, Figure 5. The cart passes the fuzzy controller four parameters θ, ω, X, V. Based on these four parameters the fuzzy controller outputs a voltage to the motor. The motor in turn calculates the force that will be exerted on the cart. The system then calculates the new values for parameters θ, ω, X, V and the cycle will be repeated.

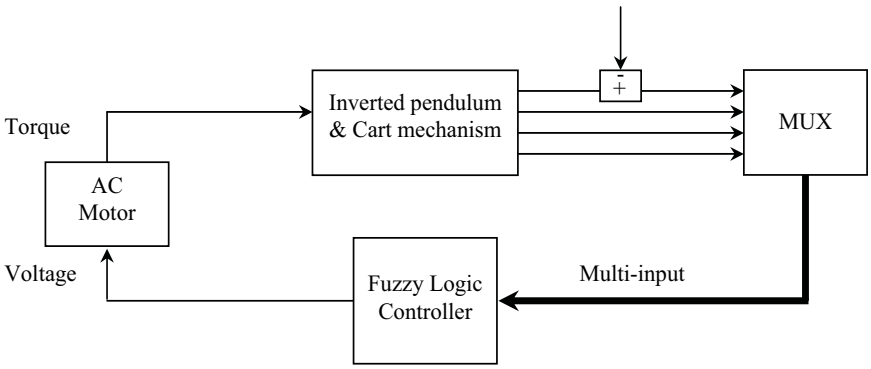


Figure 5. Schematic diagram of fuzzy controller for the inverted pendulum.

The fuzzy controller used in the simulation, with the AC motor included, is a 24 FLC as described above. The system runs identical to the 24 system only the settling time for the simulation, with the motor included, is larger. Figure 6 shows the results of the simulation using the same fuzzy controller as (Sugeno 2002) with the AC motor included in the simulation.
The AC motor has a delay, where it takes the motor a given time to reach a maximum force. This in turn causes the simulation take longer to reach steady state. Parameters used in the simulation of the motor are listed in Table 2.

T_r	=	0.06 S	Rotor time constant
L_m	=	75 mH	Magnetizing inductance
K_t	=	1.00	Motor torque constant
Rating	=	1.0HP	Motor rating
J	=	0.06kgm^2	Moment of Inertial of the motor and the load
b	=	0.002 kgm^2/S	Coefficient of friction

Table 2. Parameters of the Model Motor.

Figure 6 shows that it takes approximately 12 seconds for the pendulum's angle to become steady, and even longer for the cart's position to stabilize. The difference in the response time of this system can be found in the motor. The motor has a time constant which delays the motor's response time to an inputted voltage. A typical AC motor has a time constant larger than that of a DC motor. The shorter the time constant of the motor, the quicker the system will respond. Therefore, it can be expected that it takes longer for AC motor to balance the pendulum.

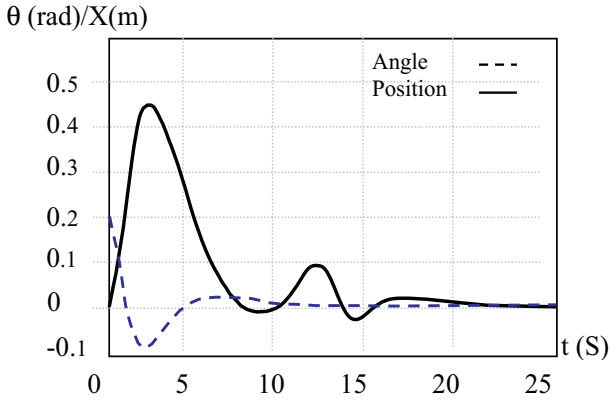


Figure 6. Variation of angle θ (rad) and position X(m) of the pendulum with time t(s).

The simulation shows that the system responds well even with a motor attached to the system. The cost of implementing a motor into the simulation is response time for the pendulum to stabilize. Simulations done without the addition of the AC motor can not be considered for real life implementation because the motor is needed to investigate the response time that the system will observe in real life.
In a series of tests carried on without the use of fuzzy controller, it was revealed that the pendulum can hardly overcome any disturbances. If the disturbance is very small, it takes twice longer for the pendulum to balance again in an upright position.
Performances of vector control AC induction motors are comparable to that of DC motors; however, AC motors are rugged and low cost. Therefore, whenever possible, usage of AC motors will greatly reduce the capital cost of equipment and devices.

7. Conclusion

In this chapter design of a Fuzzy Logic Controller for a multi-input output system is described. It demonstrates a trade-off between precision which requires complex design and simplification which achieves less precise system. There is no absolute solution in developing fuzzy logic controllers. Designer of a FLC system must consider whether precision will be sacrificed for performance and simplicity. The 52 system developed in this work was very simple and computed quickly. The drawback of this initial design was that

precision was compromised. The 24 system was also very simple and ran quickly but the performance of the system was not satisfactory. The settling time for the cart pendulum was required to be quicker. The 54 system was very complex and performance was slow, but if tuned correctly, a system of this size would be very precise.
Implementation of the system requires a high performance AC motor. Simulation results showed that the system would work for this type of motor. Having a smaller time constant in the AC motor would result in a shorter response time of the system. The FLC would need to be fine tuned for other types of motors.
With the AC motor implemented in the simulation model, the system did not react as well to high disturbances as it did when the motor was neglected in the simulation, or a DC motor was used. This indicates that the system will react well to small disturbances and be able to recover from them quickly. As the results indicates, in order for this system to handle large disturbances a motor with high performance dynamics need to be used that has a very small time constant. Use FLC made significant improvement to the controllability of the inverted pendulum by improving the response time.

8. References

Bose, K. B. (1997) Power Electronics and Variable Frequency Drives; Technology and Applications, Chapter 8: Simulation of Power Electronic and Motion Control Systems, by Ned Mohan et-al, pp. 400-453 IEEE Press.

Bush, Lawrence (2001) Fuzzy Logic Control for the Inverted Pendulum Problem, (Rensselaer Polytechnic Institute.

Jenson, Jan (2005) Design of Fuzzy Controllers, European Network for Fuzzy Logic and Uncertainty Modeling in Information Technology, http://www.iau.dtu.dk/~jj/pubs/design.pdf, Last Accessed Jan. 28, 2005.

Johansen, T.A.; Slupphaug, O.; Ji-Chang Lo; Yu-Min Chen, Comment on (2000) Stability issues on Takagi-Sugeno fuzzy model-parametric approach [and reply], IEEE Transactions on Fuzzy Systems, Volume: 8 Issue: 3 , pp. 345 -346, June 2000.

Kahaner, D.K. (1993) Fuzzy systems activities in Japan, http://www.atip.org/public/atip.reports.93/fuzzy5.93.html, Last Accessed Jan. 28, 2005.

Layne R. Layne, Passino, Kevin M. (2001) A Fuzzy Dynamic Model based state estimator, Fuzzy Sets and Systems, Vol. 122, No. 1, Aug 16, 2001, pp. 45-72.

Liang Wang; Langari, R. (1995) Building Sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques, IEEE Transactions on Fuzzy Systems, Volume: 3 Issue: 4, pp. 454 -458, Nov. 1995.

Lin, Chin-Teng and Lee, George (1996) Neural Fuzzy Systems, Prentice Hall.

Nafis, Adeel (2001) Fuzzy Control of Inverted Pendulum on a Cart, The Institute of Electrical and Electronics Engineers (IEEE), Karachi Section Conference on: technology extravaganza, 11 August 2001 http://www.ewh.ieee.org/sb/iiee/pfcip.pdf, Last Accessed Jan. 28, 2005.

Passino, Kevin M. and Yurkovich, Stephan (1998) Fuzzy Control, Addison-Wesley.

Mathworks, Inc. (2002) Sugeno-Type Fuzzy Inference, Chapter 2 of Fuzzy Logic Toolbox for use with Matlab, The, July 2002.

Tanaka, K.; Hori, T.; Wang, H.O. (2003) Multiple Lyapunov function approach to stabilization of fuzzy control systems, IEEE Transactions on Fuzzy Systems, Volume: 11 Issue: 4, pp. 582 – 589, Aug. 2003.

28

Optimal Path Planning of Multiple Mobile Robots for Sample Collection on a Planetary Surface*

J.C. Cardema and P.K.C. Wang
University of California
Los Angeles, California
U.S.A.

1. Introduction

In the exploration of a planetary surface such as that of Mars using mobile robots, rock and soil-sample collection and analysis are essential in determining the terrain composition and in searching for traces of ancient life (Malin & Edgett, 2000). Several missions to Mars have already been sent. In the 1997 Mars Pathfinder mission (Mars Pathfinder Homepage), the Sojourner rover used an alpha-proton-X-ray spectrometer (APXS) to analyze rock and soil sample compositions. It also had a simple onboard control system for hazard avoidance, although the rover was operated remotely from Earth most of the time. The method for rock and soil-sample collection is as follows. After landing, the rover used its black-and-white and color imaging systems to survey the surrounding terrain. The images were sent back to Earth, and analyzed by a team of geologists to determine where interesting samples might be found. Based on that information, the next destination for the rover was selected and the commands to get there were sent to the rover via radio with transmission delays ranging from 10 to 15 minutes (depending on the relative orbital positions of Earth and Mars). The set of commands were sent out over a day with the rover moving only a small distance each time. This was done to allow the mission control to constantly verify the position, with time to react to unforeseen problems. When the rover finally reached its destination and analyzed the sample, it spent another day transmitting the information back to Earth. The cycle was repeated as soon as the geologists had decided on the next destination for the rover. Clearly, an automated system for rock and soil sample collection would expedite the process. In the 2004 Mars Exploration Rover (MER) mission, the Spirit and Opportunity rovers (Mars Spirit & Opportunity Rovers Homepage) featured an upgraded navigation system. Imagery from a stereo camera pair was used to create a 3-D model of the surrounding terrain, from which a traversability map could be generated. This feature gave the mission controllers the option of either directly commanding the rovers or

---

* This chapter is an enhanced version of the paper by J.C. Cardema, P.K.C. Wang and G. Rodriguez, "Optimal Path Planning of Mobile Robots for Sample Collection", *J. Robotic Systems*, Vol.21, No.10, 2004, pp.559-580.

allowing them to autonomously navigate over short distances. Consequently, the rovers were often able to traverse over 100 meters a day (Biesiadecki & Maimone, 2006). The rovers were also programmed to autonomously select interesting soil samples, but this feature was seldom used. Nevertheless, this was a significant first step toward fully automating the soil-sample collection process.

In this study, an attempt is made to formulate the path planning problem for single and multiple mobile robots (referred to hereafter as "rovers" for brevity) for sample collection as a mathematical optimization problem. The objective is to maximize the value of the mission, which is expressed in the form of a mission return function. This function contains the performance metric for evaluating the effectiveness of different mission setups. To the best of our knowledge, the problem of sample collection has not yet been studied in conjunction with optimal path planning. There are many considerations in the mathematical formulation of this problem. These include planetary terrain surface modeling, rover properties, number of rovers, initial starting positions, and the selection of a meaningful performance metric for rovers so that the performance of single versus multiple rovers in representative scenarios can be compared. The basic problem is to find a sample-collection path based on this performance metric. The main objective is to develop useful algorithms for path planning of single or multiple planetary rovers for sample collection. Another objective is to determine quantitatively whether multiple rovers cooperating in sample collection can produce better performance than rovers operating independently. In particular, the dependence of the overall performance on the number of rovers is studied. To clarify the basic ideas, we make use of the Mars rover rock and soil-sample collection scenario in the problem formulation and in the numerical study.

2. Problem Description

We begin with a discussion of the problem of planetary surface modeling, followed by various operational considerations of the rovers. To facilitate the mathematical formulation of the optimal path-planning problems, a few basic definitions will be introduced. Then, precise mathematical statements of the optimal path-planning problems will be presented for both single and multiple rover cases.

2.1 Planetary Surface Modeling

Assuming that a region on the planetary surface has been selected for detailed scientific study, the main task is to develop a suitable terrain surface model for rover path-planning. Initially, a crude surface model for the selected spatial region may be constructed from the aerial planetary survey data obtained by fly-by spacecraft or observation satellites such as the Mars Orbiter. Once the rovers are on the planetary surface, more refined models (usually localized model) may be constructed from the image-data generated by on-board cameras. Although the refined models may be useful for scientific studies, they may not be useful for practical optimal path planning. Therefore we resort to approximate models that simplify the mathematical formulation and numerical solution of the optimal path-planning problems. In our model, we assume that the area of the spatial domain for exploration is sufficiently small so that the curvature of the planetary surface can be neglected. Moreover, the surface is sufficiently smooth for rover maneuvers.

2.1.1 Approximate Surface Model

Let Ω be a bounded spatial domain of the two-dimensional real Euclidean space R^2 and the representation of a point in R^2 with respect a given orthonormal basis be denoted by x. Let $f = f(x)$ be a real-valued continuous function defined on Ω. Let $G_f \stackrel{\text{def}}{=} \{(x, f(x)) \in R^3 : x \in \Omega\}$ denote the *graph of* f, which represents the planetary surface under consideration. In this work, we use a polygonal approximation for the planetary surface G_f via triangulation that partitions G_f into adjacent, non-overlapping triangular patches, where each edge of a triangular patch is shared by exactly two triangular patches except on the boundaries of G_f. It has been proved that every C_1-surface defined on Ω with a sufficiently smooth boundary has a triangulation, although an infinite number of triangular patches may be required (Weisstein). Here we make use of the Delaunay triangulation, which produces a set of lines connecting each point in a given finite point set to its neighbors. Furthermore, it has the property that the triangles created by these lines have empty circumcircles (i.e. the circumcircles corresponding to each triangle contains no other data points). The Delaunay triangulation of G_f is a polygonal approximation of the original planetary surface. It can also be thought of as the projection of the planetary surface onto a mesh space. The domain of the triangulation is a mesh space denoted by $\hat{\Omega} \subset \Omega$, where $\hat{\Omega}$ is the discrete version of Ω. The resulting polygonal approximation of the planetary surface G_f will be denoted by $\hat{G}_f$. This approximate surface model will be used in formulating the optimal path-planning problem. Although one might use other forms of approximation for G_f that lead to smoother approximate surfaces, our choice provides significant simplification of the optimal path-planning problem, since the paths are restricted to lie on the edges of the triangular patches.

2.1.2. Rock and Soil-sample Properties

Rock and soil samples have different values to geologists based on the questions they are trying to answer. For example, in Mars exploration, sedimentary rocks are important since two of the primary questions about early Martian geological history are whether liquid water could exist on its surface and, if so, whether liquid water ever took the form of lakes or seas (Malin & Edgett, 2000). According to Malin and Edgett, outcrop materials are interpreted as Martian sedimentary rock, and they are of particular interest to geologists for answering these questions. The outcrop materials occur in three types: layered, massive, and thin mesas, which differ in visual tone, thickness, texture, and configuration. The locations of these outcrops are limited to specific regions mostly between ± 30 degrees latitude. One of the regions with large outcrop occurrence is in Valles Marineris. The terrain in a portion of this region is used for our case study. The three types of outcrops are speculated to come from different Martian ages. The rover should have the capability of identifying and distinguishing these different types. To model a portion of Valles Marineris, we assume that the rock and soil samples are randomly distributed over various sub-regions. An appropriate model should allow for the specification of any rock and soil-sample distribution on the given Mars terrain. A simple way to do this is to divide the terrain into sub-regions and assign weights to determine how many samples to uniformly

distribute within each sub-region. Moreover, we assume there are a finite number of samples, each with an assigned value in a prescribed range. A high sample value implies high scientific value. In practical situations, this task may be accomplished by a careful study of the aerial survey data.

2.2. Single Rover Case

2.2.1 Operational Considerations

In what follows, we consider the main factors that are relevant to the formulation of the path-planning problem.

2.2.1.1 Dynamic properties

The rover is modeled simply. We only consider mass m, maximum traversable slope or tilt angle θ, maximum speed $v_{\max}$, and maximum power $P_{\max}$. They are used to calculate the rover's traveling time on terrains with varying slopes. Higher-order dynamics involving the acceleration of the rover and the damping effects of the suspension system are not included, since the actual motion of the rover is relatively slow compared to that of mobile robots in a laboratory environment. In what follows, the term "sample collection" is used interchangeably with "sample analysis", although they do not necessarily have the same connotation. (i.e. sample collection can be thought of as the collection of sample data.) In the case where the rover physically picks up the sample, the mass of each collected sample is added to the overall mass of the rover. There is also a loading constraint that limits the number of samples that the rover can physically hold in its storage compartment. In this study, we do not consider the situation where the rover can only use its imaging system to identify and detect samples. This leads to the problem of determining a path that maximizes the visual coverage of the terrain (Wang, 2003, 2004).

2.2.1.2 Mission time limit

The mission length is an important consideration in path planning. For the 1997 Pathfinder mission, the planned mission duration was 30 days. The algorithm for path planning should verify that the time duration for sample collection is within the prescribed mission time limit (denoted by $\tau_{\max}$), which in turn determines the maximum terrain coverage.

There should be a clarification about the distinction between the overall mission time and the time it takes to execute a planned path. In practical situations, it would be difficult to plan a path for the entire mission duration. It would be more reasonable to plan paths of shorter duration that can be executed at specific intervals during the mission. However, to simplify our formulation, we do not make this distinction and assume that we can plan a path for the entire mission duration.

2.2.1.3 Sample analysis time

As mentioned earlier, the Sojourner rover in the Pathfinder mission was equipped with a spectrometer (APXS) for rock and soil-sample analysis. The sensor head of the APXS was placed on the sample for 10 hours during the analysis. To account for this, the sample analysis time τ_{wait} is introduced into our model. It represents the amount of time required to analyze the sample. To simplify the model, we assume that $\tau_{\max}$ is the same for every sample, regardless of its type. With the inclusion of the sample analysis time, the rover is

forced to consider more carefully which rock and soil samples to collect while still operating within the time limit.

2.2.1.3 Mission return function

To characterize the performance of a rover in rock and soil sample collection, we need to choose an appropriate mission return function to quantify the rover's performance throughout the mission. The mission return function used in this study is simply the sum of the collected sample values.

2.2.1.4 Terrain risk

Let $\theta_{\max}$ denote the angle of the maximum traversable slope corresponding to the maximum allowable tilt angle of the rover before it topples over. To determine if a point on the surface is too risky to traverse, the terrain slopes at that point in all directions are computed. If the magnitude of the slope angle in any direction exceeds $\theta_{\max}$, that point is deemed un-traversable. Usually, the rover is more susceptible to tipping over sideways than forwards or backwards, although the dimensions of the rover are not considered in this study.

2.2.1.5 Terrain properties

Ideally, the terrain surface in the spatial region chosen for exploration should be sufficiently smooth to facilitate rover maneuverability but also has features to indicate the possible presence of interesting samples. For rover traversability, the terrain texture and hardness are also important (Seraji, 2000). Terrains that are rough and rocky are avoided in favor of smoother ones. Terrain risk depends on both the terrain texture and hardness.

2.2.2. Definitions

Having specified an approximate surface in the form described in Sec. 2.1.1, a path can be specified on this surface. First, we introduce the notion of an admissible segment.

2.2.2.1 Definition 1

(*Admissible segment*): A segment γ connecting a point $(x_a, f(x_a)) \in \hat{\mathsf{G}}_f$ with an adjacent point $(x_b, f(x_b)) \in \hat{\mathsf{G}}_f$ along an edge of a triangular patch formed by the Delaunay triangulation of G_f is said to be admissible if it satisfies the following constraints induced by the terrain risk:

$$\theta_a = \arcsin\left|\frac{f(x_a) - f(x_{ai})}{d}\right| \le \theta_{\max} \quad \text{for all } x_{ai},$$

and

$$\theta_b = \arcsin\left|\frac{f(x_b) - f(x_{bi})}{d}\right| \le \theta_{\max} \quad \text{for all } x_{bi},$$

where x_{ai} and x_{bi} are the adjacent points of x_a and x_b, respectively, and $d = \left\|(x_i, f(x_i)) - (x_{i+1}, f(x_{i+1}))\right\|$ is the Euclidean distance between points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$. (i.e. The slopes of $\hat{\mathsf{G}}_f$ from x_a and x_b to all their neighboring points satisfy the maximum traversable slope angle constraint $\theta_{\max}$).

2.2.2.2 Definition 2

(*Admissible path*): A path Γ composed of connected segments in $\hat{\mathsf{G}}_f$ is said to be admissible if each segment is admissible.

Each admissible path can be represented by an ordered string of points that will be denoted by $\mathsf{S} \subset \hat{\Omega}$. The string of points S may include repeated points since partial backtracking along the path is allowed. To account for different sample values in the model, the samples are individually indexed. Each sample σ_k has a corresponding value λ_k, where k is the index number. We assume that the sample values as well as the sample distribution $D_{ss}(x)$ on the terrain defined below are known *a priori*.

2.2.2.3 Definition 3

(*Sample distribution*): The sample distribution $D_{ss} = D_{ss}(x)$ is a set-valued function defined as follows: If at a point $x \in \hat{\Omega}$, there are m samples with indices $\mathsf{J}_x = \{k_{x,1}, k_{x,2}, ..., k_{x,m}\}$, then $D_{ss}(x) = \mathsf{J}_x$. If there are no samples at $x \in \hat{\Omega}$, then $D_{ss}(x)$ is an empty set.

The entire set of all samples indices is denoted by E_{ss}. Along each admissible path Γ with the corresponding string S, there is a set of *collectable samples* C_{ss} that includes all the samples contained in S as defined below:

2.2.2.4 Definition 4

(*Collectible sample set):* If S_Γ is the string of points associated with an admissible path Γ, and $\mathsf{J}_\Gamma = \{k_1, k_2, ..., k_N\}$ is the set of sample indices such that

$$\mathsf{J}_\Gamma = \bigcup_{x \in \mathsf{S}_\Gamma} D_{ss}(x),$$

where N is the number of soil samples along the path, then the collectable sample set is $\mathsf{C}_{ss} = \mathsf{J}_\Gamma$.

Next, we define the attainable set associated with an initial point x_0 at time t_0.

2.2.2.5 Definition 5

(*Attainable set*): The attainable set at time t starting from $x_0 \in \hat{\Omega}$ at time t_0 (denoted by $\mathsf{A}(t; x_0, t_0) \subset \hat{\Omega}$) is the set of all points in $\hat{\Omega}$ that can be reached at time t via admissible paths initiating from x_0 at time t_0.

Since $\hat{\mathsf{G}}_f$ is time-invariant, we can set $t_0 = 0$. Successive points along the admissible path Γ will be restricted to this attainable set. An attainable set associated with a maximum mission time duration $\tau_{\max}$ will be denoted by $\mathsf{A}_{\max}(x_0) = \mathsf{A}(\tau_{\max}; x_0, 0)$, which will be referred to as the *maximal attainable set from* x_0.

Evidently, we can find the admissible paths and their collectible sample sets associated with a given $\mathsf{A}(t; x_0, 0)$. Once we determine the samples to collect along the path, we can introduce the notion of an *admissible tour*.

2.2.2.6 Definition 6

(*Admissible tour*): An admissible tour T_a is a pair $(\Gamma_a, \mathsf{L}_{ssa})$, where Γ_a is an admissible path with point set S_a and collectible sample set C_{ssa}, and L_{ssa} is the list of soil samples collected along the path Γ_a.

In an admissible tour, the time for traversing the path (including the sample analysis time) is within the mission time limit. If n is the number of samples collected, then the mission time τ_m is given by

$$\tau_m = \sum_i \tau_i + n \cdot \tau_{wait} \le \tau_{\max}, \tag{1}$$

where τ_{wait} is the time required to analyze each sample; $\tau_{\max}$ is the mission time limit; τ_i is the traveling time between the successive points x_i and x_{i+1} in S_a and is given by

$$\tau_i = \max\left\{\frac{m g_m h}{P_{\max}}, \frac{d}{v_{\max}}\right\}, \tag{2}$$

where g_m is the acceleration due to gravity of the planet, $h = f(x_i) - f(x_{i+1})$ is the difference in terrain elevation at the points x_i and x_{i+1}, and $d = \left\|(x_i, f(x_i)) - (x_{i+1}, f(x_{i+1}))\right\|$. Thus, τ_i corresponds to the maximum of the traveling times under the power and speed constraints. Physically, when climbing up a slope, the power constraint is used to compute τ_i. When the terrain is flat or sloping downward, the speed constraint is used instead.

For an admissible tour T_a, the sample list L_{ss} contains the index of each sample in the order of collection. Each sample σ_k in the sample list has value λ_k. Let $\mathsf{I} = \{k_1, k_2, ..., k_j, ..., k_n\}$ be the set of sample indices in the order of collection, where $\mathsf{I} \subset \mathsf{C}_{ssa}$ and k_j is the index of the j th sample collected along the path. Then, $\mathsf{L}_{ss} = \mathsf{I}$.

2.2.3. Problem Formulation

Assume that an approximate planetary surface $\hat{\mathsf{G}}_f$, initial starting point x_0, mission time limit $\tau_{\max}$, sample analysis time τ_{wait}, sample index set E_{ss}, samples σ_k's with values λ_k, $k \in \mathsf{E}_{ss}$, and sample distribution $D_{ss} = D_{ss}(x)$ are given.

2.2.3.1 Problem P1

Find an optimal tour T° *(with admissible path* Γ_a^o *and the corresponding sample list* L°_{ssa}*) that maximizes the mission return function*

$$V(\mathsf{L}_{ssa}) = \sum_{k \in \mathsf{L}_{ssa}} \lambda_k, \tag{3}$$

where λ_k *is the value of sample* σ_k*, i.e.*

$$V(\mathsf{L}^o_{ssa}) \ge V(\mathsf{L}_{ssa}) \tag{4}$$

for all L_{ssa} *associated with admissible paths* Γ_a.

To find an optimal tour, we must first consider all possible admissible paths from the given initial starting point x_0. The total traveling time τ_{at} along each admissible path Γ_a (with path point set S_a and collectible sample set C_{ssa}) satisfies the constraint:

$$\tau_{at} = \sum_i \tau_i \le \tau_{\max}, \tag{5}$$

where τ_i is the traveling time between successive points x_i and x_{i+1} in S_a.
Along each admissible path, we have a set C_{ssa} of collectable samples. Let $\mathsf{I} \subset \mathsf{C}_{ssa}$ be a set of sample indices in the order of collection such that

$$\tau_{am} = \sum_i \tau_i + n \cdot \tau_{wait} \le \tau_{\max}, \tag{6}$$

where τ_{am} is the total mission time, n is the number of samples in I, and each τ_i is the traveling time between successive points x_i and x_{i+1} in S_a along the path. All possible sets of sample indices I are considered.
For each admissible path Γ_a (with path point set S_a and collectible sample set C_{ssa}), we search through all possible sets of sample indices $\mathsf{I} \subset \mathsf{C}_{ssa}$, and find a $\mathsf{I}^* \subset \mathsf{C}_{ssa}$ that maximizes the mission return function

$$V(\mathsf{I}) = \sum_{k \in \mathsf{I}} \lambda_k, \tag{7}$$

i.e.

$$V(\mathsf{I}^*) = \max\{V(\mathsf{I}) : \mathsf{I} \subset C_{ssa}\} \text{ and } \mathsf{L}_{ssa} = \mathsf{I}^*. \tag{8}$$

Let Γ_A denote the set of all admissible paths, and Λ_A the corresponding set of sample lists. Once we have performed the maximization for each admissible path, the optimal sample list L^o_{ss} is found by taking the path and sample list that maximizes the mission return function, i.e.

$$V(\mathsf{L}^o_{ss}) = \max\{V(\mathsf{L}_{ss}) : \mathsf{L}_{ss} \in \mathsf{L}_A\}. \tag{9}$$

The optimal path Γ^o is the path associated with L^o_{ss}, and the optimal tour is $\mathsf{T}^o = (\Gamma^o, \mathsf{L}^o_{ss})$.
The optimal sample collection path generally depends on the initial starting position of the rover. Intuitively, we would like to place the rover as close as possible to the highest-valued samples. Since the distributions of the samples are known, we could also maximize the mission return function with respect to the starting position. But in practical situations, the starting position cannot be specified precisely. Moreover, the sample distributions are not known beforehand.

2.3. Multiple Rover Case

In the multiple rover case, it is necessary to introduce a few additional notions.

2.3.1. Operational Considerations

2.3.1.1 Starting positions

First, let us consider the attainable set of each rover corresponding to a given initial starting position at time $t = 0$. If the maximal attainable sets of two or more rovers overlap, then a decision has to be made on the assignment of rovers to cover the overlapping region. This decision can be made either by the mission planner (centralized operation) or by the rovers themselves (autonomous operation).
As in the single rover case, the choice of starting positions is an important issue. If the rovers are placed too close together, they could interfere with each other's collection tasks. If the

rovers are placed too far apart, then there is little cooperation among them since the overlapping region is small. Therefore, it is desirable to place the rovers such that a balance between cooperation and interference can be reached. The problem of finding the optimum starting positions for m rovers, with or without *a priori* knowledge of the sample distributions, is an interesting one. This problem will not be considered here. In what follows, we assume that the starting positions of the rovers are pre-assigned.

2.3.1.2 Interaction

The interaction between the rovers can lead to either cooperation or interference. In order to promote cooperation, the rovers can actively communicate with each other and decide among themselves on how to split up the terrain to make the collection process most efficient. Alternatively, a central supervisor or mission planner can make all these decisions beforehand and predetermine the rover's sample collection path. We expect that the performance of multiple rovers with interaction and cooperation is better than that of multiple rovers operating independently.

2.3.1.3 Centralized vs. Autonomous Operation

The distinction between the centralized and autonomous operations depends on information utilization. In the centralized operation, all the information about the terrain including sample locations is known beforehand. This information is analyzed and the optimum paths for the rovers are predetermined. This is a simplified version of the real-world scenario. In practical situations, the terrain details as well as the rock and soil-sample locations are not completely known, hence autonomous operation would be more desirable. Here, each rover must rely on its vision system to examine the surrounding terrain and to detect samples. Since the range of view of the rover is limited, cooperation between the rovers is more desirable. Using the data from its vision system, each rover would then plan its own optimum path while keeping in communication with the other rovers for promoting cooperation and avoiding interference. The autonomous operation could account for inaccuracies or uncertainties in the rover's terrain information. In what follows, we consider only the centralized operation.

2.3.2. Centralized Operation

The multiple rover case is similar to the single rover case when the rovers are placed far apart. Since there is little or no interaction, the problem can be reduced to one involving separate single rover cases. When the rovers are placed close enough such that interaction occurs, a new approach to the problem must be developed. We shall consider several different sample collection strategies for m rovers with given initial starting positions and sample distributions.

2.3.2.1 Best Route First

Here, we solve the single rover case for each rover and compute the optimal tours. We search for the rover whose tour has the highest mission return function value and keep only that tour. The tour is assigned to that rover and its collected samples are removed from the terrain. The process is repeated for the remaining rovers until a tour has been assigned to each rover. The drawbacks with this strategy include long computation time, especially for a large number of rovers and samples. Moreover, the tour value of each successive rover is less than that of the preceding rover in the iteration process.

2.3.2.2 Partition of Overlapping Attainable Set: Closest Rover

The maximal attainable sets of the m rovers are examined and the terrain is divided into overlapping and non-overlapping regions. The overlapping region includes all points on the terrain that can be reached by more than one rover within the given mission time limit. Since each sample in the overlapping region can be assigned to only one rover, the set of samples in the overlapping region must be partitioned according to some given criteria. Here, the samples in the overlapping region are each assigned to one of the rovers based on the distance from the sample's position to each rover's starting position. The rover whose starting position is closest to the sample's position is assigned to collect that sample. Once the partitioning of the samples in the overlapping region is completed, the problem reduces to m single rover problems with each rover limited to a subset of the collectable samples. However, a rover may be assigned more samples than it can collect within the prescribed mission time limit. In that case, after determining the rover's tour, the uncollected samples are passed on to the other rovers for consideration.

2.3.2.3 Partition of Overlapping Attainable Set: Closest Path

This strategy is similar to the previous one except that the criterion for partitioning samples in the overlapping region of the maximal attainable sets is different. The samples in the overlapping region are assigned to one of the rovers based on the distance from the sample's position to each rover's preliminary path (a path planned before the overlapping region is taken into account). The rover whose preliminary path comes the closest to the sample's position is assigned to collect that sample. This criterion makes the sample collection task easier, since it involves only a slight deviation from the preliminary path. Again, if a rover cannot collect all of its assigned samples, the uncollected samples are passed on to the other rovers.

One possible modification is to insert the sample into the preliminary path when considering the subsequent samples in the overlapping region. This may result in a better partitioning of the overlapping region. For simplicity, this is not done in this work. Rather, when considering other samples, we use the original preliminary path. In what follows, only the "Partition of Overlapping Attainable Set: Closest Path" strategy will be considered.

2.3.3. Definitions

The sample collection path for rover j is Γ_j and is associated with a collectible sample set $\mathsf{C}_{ss,j}$. Each path Γ_j is represented by an ordered set of points that will be denoted by $\mathsf{S}_j \subset \hat{\Omega}$. This set may include repeated points. Let the sample distribution be denoted by $D_{ss} = D_{ss}(x)$, and the entire set of sample indices by E_{ss}.

The attainable set of rover j at a particular time t, starting from an initial point $x_{0,j} \in \hat{\Omega}$ at t_0, is denoted by $\mathsf{A}_j(t; x_0, t_0) \subset \hat{\Gamma}$. Successive points along the admissible path Γ_j will be restricted to this attainable set. The attainable set of rover j associated with the maximum mission time $\tau_{\max}$ is denoted by $\mathsf{A}_{\max,j} = \mathsf{A}_j(\tau_{\max}; x_0, t_0)$. This attainable set $\mathsf{A}_{\max,j}$ is the maximum attainable set of rover j.

The spatial domain $\hat{\Omega}$ is partitioned into sub-regions based on the maximal attainable sets $\mathsf{A}_{\max,1},\ldots,\mathsf{A}_{\max,m}$. Let $\hat{y} \in \hat{\mathsf{S}}$ denote the set of points in the overlapping region of the maximal

attainable sets. (i.e. All points in $\hat{Y}$ can be reached by more than one rover.) Let M_j = $\mathsf{A}_{\max,j}$ - ($\mathsf{A}_{\max,j} \cap \hat{Y}$) be the sub-region of the spatial domain that can only be reached by rover j. The set of sample indices in the overlapping region will be denoted by $\mathsf{I}_{ss} \in \mathsf{E}_{ss}$. The sample indices in each sub-region M_j are denoted by $\mathsf{F}_{ss,j} \subset \mathsf{E}_{ss}$.

2.3.4 Problem Formulation

There are many possible formulations of the optimal path-planning problem for sample collection involving multiple rovers. We give one such formulation.

2.3.4.1 Problem P2

Given the approximate planetary surface $\hat{\mathsf{G}}_f$; m rovers with initial starting points $x_{0,1},...,x_{0,m}$; mission time limit $\tau_{\max}$; sample analysis time τ_{wait}; sample index set E_{ss}; samples σ_k with values λ_k, $k \in \mathsf{E}_{ss}$; and sample distribution $D_{ss} = D_{ss}(x)$, find optimal tours $\mathsf{T}_1^o,...,\mathsf{T}_m^o$ (each with path Γ_j^o and sample list $\mathsf{L}_{ssopt,j}$) that maximize the mission return function

$$V(\mathsf{L}_{ss,1},...,\mathsf{L}_{ss,m}) = \sum_{j=1}^{m} \sum_{k \in \mathsf{L}_{ss,j}} \lambda_k. \tag{10}$$

Consider the "Partition of Overlapping Attainable Set: Closest Path" strategy for finding the optimal tours. We need to first partition the terrain into sub-regions based on the maximal attainable sets. By considering the terrain without including the overlapping region $\hat{Y}$, we can solve m single rover problems: "Rover 1 starting at $x_{0,1}$ with sub-domain M_1 " to "Rover m starting at $x_{0,m}$ with sub-domain M_m." For each rover j, we maximize the mission return function

$$V(\mathsf{L}_{ss,j}) = \sum_{k \in \mathsf{L}_{ss,j}} \lambda_k \tag{11}$$

over the sub-domain M_j and the corresponding set of sample indices $\mathsf{F}_{ss,j}$. From these m single rover problems, we obtain m tours $\mathsf{T}_i = (\Gamma_i, \mathsf{L}_{ss,i})$, $i = 1,...,m.$ These tours are not optimal since we have not yet considered the samples in the overlapping region.

Next, we consider each sample in the overlapping region $\hat{Y}$. For each sample with index $k \in \mathsf{I}_{ss}$, we find the rover j whose path Γ_j (with path point set S_j) is closest to the sample. We minimize

$$d_p(x_i) = \min_j \min_{x \in \mathsf{S}_j} \left\| (x_i, f(x_i)) - (x, f(x)) \right\| \tag{12}$$

ith respect to x_i, the spatial coordinate of the ith sample in the overlapping region. The rover j that minimizes this function is assigned to collect the sample at x_i. This is repeated for all the samples in the overlapping region. After the assigning of the samples in the overlapping region, each rover j has a new set of samples to collect from. This new set of sample indices is denoted by $\mathsf{F}'_{ss,j}$.

Now, we repeat solving m single rover problems with each rover j limited to the sample set $\mathsf{F}'_{ss,j}$. The resulting tours are near-optimal.

2.4 Sample Collection Problem

The Sample Collection Problem (SCP) is an instance of the well-known Traveling Salesman Problem (TSP). A brief discussion of the TSP can be found in Appendix A. In the TSP, the problem is to find a path that visits all the nodes with minimum total traveling distance. In the SCP, the problem is to find a path that maximizes the value of the nodes visited within a specified maximum mission time limit. The differences between the TSP and the SCP are: (i) the TSP begins and ends at the same node while the SCP can end anywhere, (ii) the SCP has a waiting time associated with picking up a sample to account for the sample analysis time, (iii) the samples have different values, so different payoffs are associated with different nodes, (iv) instead of finding the minimum total distance, the SCP tries to maximize the value of all the collected samples, and (v) not all the nodes need to be visited. These modifications make the SCP a much more complex problem to solve than the original TSP, which is known to be NP-hard.

The heuristic used in solving the SCP is the maximum-value heuristic, which is similar to the minimum-distance heuristic used in solving the TSP. Instead of minimizing the total distance traveled, the maximum-value heuristic calls for maximizing the value of a weighting function that takes into account the value of each sample as well as the distance. At a given position x_i, the weighting function is used to decide on the next sample to collect.

3. Algorithms

As mentioned in Sec. 2.1.1, we only consider the rover's mass m, maximum traversable slope (or tilt) $\theta_{\max}$, maximum velocity $v_{\max}$, and maximum power $P_{\max}$. The samples consist of three different types, each type with a corresponding value (1, 3, or 9) representing its relative worth to geologists studying the planetary surface. The samples are randomly distributed on the terrain in the following way. The terrain is first divided into sub-regions and each sub-region j is assigned a weight $W_{i,j}$ for each sample type i such that

$$\sum_j W_{i,j} = 1 \quad \text{for all } i. \tag{13}$$

The total number of samples for each type $n_{i,tot}$ is given beforehand, and the numbers for each sub-region j, $n_{i,j}$, depend on the weights according to the following:

$$n_{i,j} = C\left(\frac{W_{i,j}}{\sum_j W_{i,j}} n_{i,tot} \right), \tag{14}$$

where the ceiling function $C(v)$ rounds v to the integer $\geq v$. Once the number of each sample type for each sub-region is given, the samples are uniformly distributed within that sub-region. Multiple samples are allowed to occupy the same spatial point. If a certain type of sample is collected, the values of the other samples of that type are not affected. This may be different from the real scenario where once one sample type has been collected; there is no need to find other samples of that type.

Starting from the rovers' initial positions, the maximal attainable sets corresponding to the mission time limit $\tau_{\max}$ are computed. These maximal attainable sets represent the maximum range of each rover when no samples are collected. As the maximal attainable

sets are computed, the best path to each point on the terrain is stored. All possible paths from the starting point are explored. We examine one path at a time and follow it until time expires. As the path reaches each point, we do the following: If this is the first time a path has reached this point, the traveling time to the point is stored. If this is not the first time, the traveling time to the point by the current path is compared to the previously stored time. If the current time is less than the stored time, the current time replaces the stored time. The point may be reached many more times as other paths are tried and the best time is always kept. In this way, the best times to each point are stored as the computation progresses.

In order to retrace a path given the starting and terminal points, we introduce the "previous path point" variable. Whenever the best time for a path point is stored, the path point the rover came from (or previous path point) is also stored. Therefore, by going from a previous path point to the previous path point, the path can be retraced. This method saves memory space, since only one previous path point variable has to be stored for each attainable point on the terrain instead of an entire list of path points.

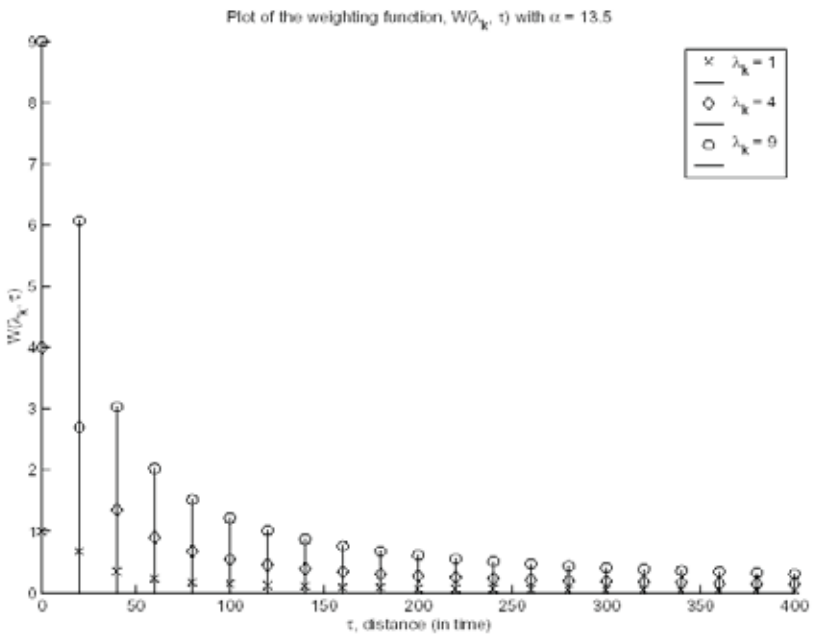


Fig. 1. Plot of the weighting function.

The set of best times (TimeMatrix), previous path points (PMatrix), and maximal attainable set (AMatrix) for each starting point are saved as Path Planning Data Sets (PPDS), which depends on the starting point and the maximum mission time. This

pre-computation saves time later when solving the SCP. The PPDS starting at each sample are also computed, which is the most time-consuming operation in the program.

Since considering all possible admissible paths is time consuming and memory intensive, an approximate solution is obtained by applying the "maximum-value heuristic" and "3-opt switching" to solve the SCP. The "maximum-value heuristic" is almost a "greedy" heuristic, which ignores all the samples except for the highest-valued ones.

The "maximum-value heuristic" is based on a weighting function that weights each sample based on its value and its distance from the starting point. Sample σ_k with value λ_k from a point x_0 is given and $\tau(x_0,\sigma_k)$ (the time it takes to get from point x_0 to sample σ_k). We define the weighting function as

$$W(\lambda_k,\tau(x_0,\sigma_k)) = \begin{cases} \lambda_k, & \text{if } \tau(x_0,\sigma_k)=0; \\ \alpha\lambda_k / \tau(x_0,\sigma_k), & \text{otherwise.} \end{cases} \tag{15}$$

We collect the sample that maximizes this weighting function. The value for α in (15) is determined by setting the value of a sample 1 meter away from x_0 to be 3/4 of the λ_k value. In Figure 1, the weighting function for α = 13.5 and different values of λ_k are plotted with $\tau(x_0,\sigma_k)$ as a variable.

The algorithm also looks ahead two steps and maximizes the weighting function for the next two samples to collect, as well as taking into account the remaining mission time. Looking ahead two steps helps steer the rover toward the higher concentrations of samples. Let $W_i(\lambda_{k,i},\tau_{i,i-1})$ be the weighting function value for the ith step, with sample k_i and where $\tau_{i,i-1}$ is the time required to traverse from step $i-1$ to step i. The algorithm weights the second step a fraction $1/\beta$ of the value of the first step. The algorithm maximizes the function

$$W_{tot} = W_1(\lambda_{k,1},\tau_{1,0}) + \frac{1}{\beta}W_2(\lambda_{k,2},\tau_{2,1}) \tag{16}$$

with respect to $\tau_{1,0}$ and $\tau_{2,1}$ satisfying

$$(\tau_{1,0}+\tau_{wait})+(\tau_{2,1}+\tau_{wait}) \le \tau_{\max}, \tag{17}$$

where $\tau_{1,0}$ is the time to reach sample k_1 from the starting position x_0, and $\tau_{2,1}$ is the time to reach sample k_2 from sample k_1. In our case studies, the value for β is found by setting $W_1(9,250) > W_1(1,500) + (1/\beta)W_2(9,0)$. The value of 500 for the time length was chosen since it is equal to the maximum mission time used for the flat terrain experimental case. The calculated value is $\beta \approx 20$.

After the list of collected samples has been determined, we apply "3-opt switching" (see Appendix A3) to obtain a more time-efficient solution. Then, we determine whether more samples can be collected. If so, the "3-opt switching" is performed again, repeatedly as necessary. The resulting tour is locally optimized, but is not necessarily optimal.

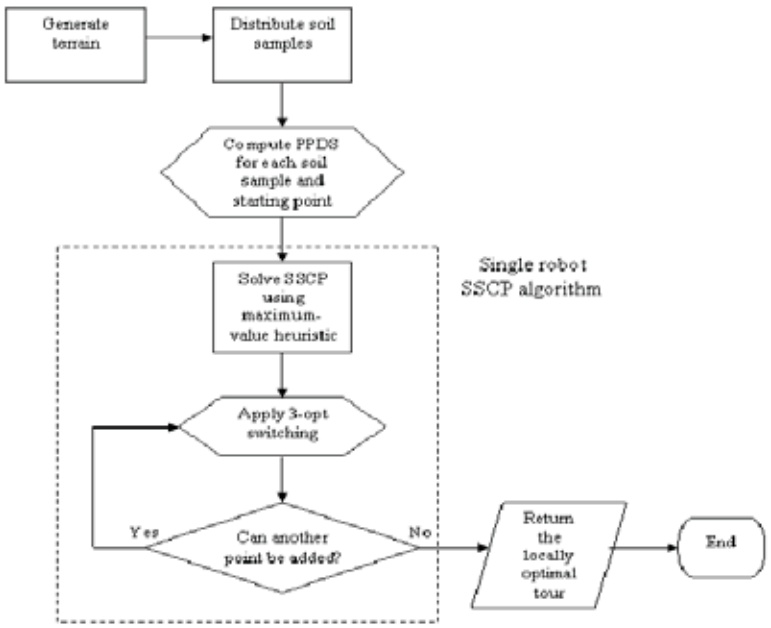


Fig. 2. Flow chart for the single rover case.

3.1 Single Rover Case

Starting from an initial point x_0 and a list of sample indices E_{ss}, an admissible tour T_a (with path Γ_a and sample list L_{ssa}) is composed by using the "maximum value heuristic" that looks ahead two steps. This tour is subject to the time constraint (6), i.e.

$$\tau_{am} = \sum_i \tau_i + n \cdot \tau_{wait} \le \tau_{\max}, \tag{18}$$

where τ_{am} is the elapsed mission time for the admissible tour and n is the number of samples in L_{ssa}.

Next, "3-opt switching" is applied to the sample list L_{ssa} to determine if the same samples can be collected in less time. Each remaining uncollected sample is then considered to determine if it can be collected and analyzed within the remaining mission time. We collect the remaining sample with the highest weighting function value that still satisfies the mission time constraint. Every time an additional sample is added to the sample list L_{ssa}, "3-opt switching" is applied. This process is repeated until the list is unchanged through one iteration. The resulting tour is locally optimized, but not necessarily optimal.

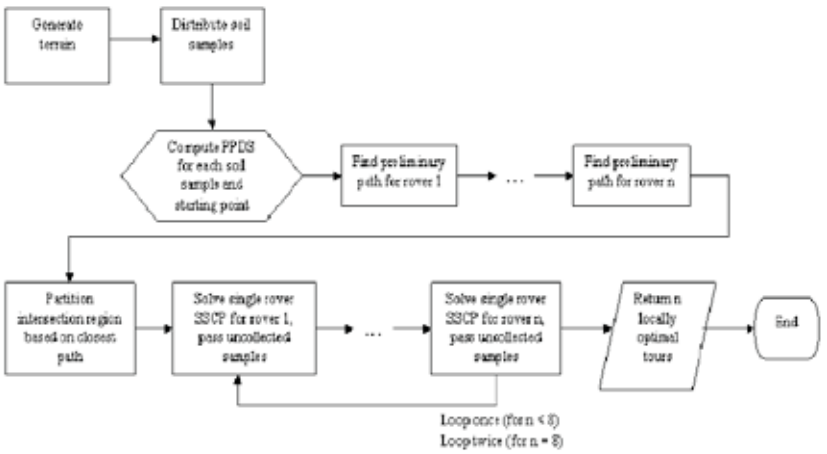


Fig. 3. Flow chart for the multiple rover case.

3.2 Multiple Rover Case

In the multiple rover cases, we assign the samples in the overlapping region to the rover whose preliminary path is closest to the sample as discussed in Sec. 2.2.1. Let $m \geq 2$ be the number of rovers. Each rover's preliminary path is found by removing the samples in the overlapping region and solving the SCP for each rover. To reduce the computation time, only the collected samples along the path are used instead of every point along the path when computing the distance to the samples in the overlapping region. If σ_j is the sample in the overlapping region in question, the distance from σ_j to rover i 's path, denoted by $\delta_{i,j}$, is found according to:

$$\delta_{i,j} = \min_{x_k} \left\| (x_j, f(x_j)) - (x_k, f(x_k)) \right\|,$$

where x_k is the position of sample $\sigma_k \in$ L $_{ssa}$, and x_j is the position of soil sample σ_j in the overlapping region $\hat{Y}$. After partitioning the overlapping region, each rover has its own assigned set of samples to collect, and each sample can only be assigned to one rover at a time.

After partitioning the samples in the overlapping region among the rovers, the multiple rover case reduces to solving m single rover cases. After the first rover's tour is determined, we assign its uncollected samples to the next rover, since considering this extra set of samples may result in a better tour for the second rover. Similarly, after each rover's tour is computed, the uncollected samples are always passed on for the next rover to consider. After the m th rover's tour is determined, its uncollected samples are passed back to the first rover and all the tours are computed again. For the one-, two-, and four-rover cases, this loop-back only occurs once. In the eight-rover case, the loop-back occurs twice. This ensures that each rover has a chance to consider all the uncollected samples.

One additional note is that collision avoidance was not implemented. Adding this consideration would further complicate the path-planning algorithm and may not be useful at this point, since each rover is assumed to be a point mass in this study. In the real-life scenario, collision avoidance must be included and the dimensions of the rover must be taken into account.

4. Case Study

In this study, real Mars terrain data obtained from the Mars Orbiter Laser Altimeter (MOLA) Science Investigation are used. The terrain area chosen is relatively flat with a high plateau. The terrain is assumed to be smooth. Delaunay triangulation is used to create a 24 × 24 m^2 mesh for approximating the terrain. From this point on, *terrain* is used to refer to the approximated version of the Mars data.

4.1 Flat Terrain

First, we consider the case of a flat 24 × 24 m^2 terrain after Delaunay triangulation, shown in Fig. 4. This case provides a test of the path-planning algorithm under idealized conditions. Here, each node on the terrain is labeled from 1 to 576, starting from the bottom left corner and moving up the vertical axis. This label will be referred to as the *x-reference*.

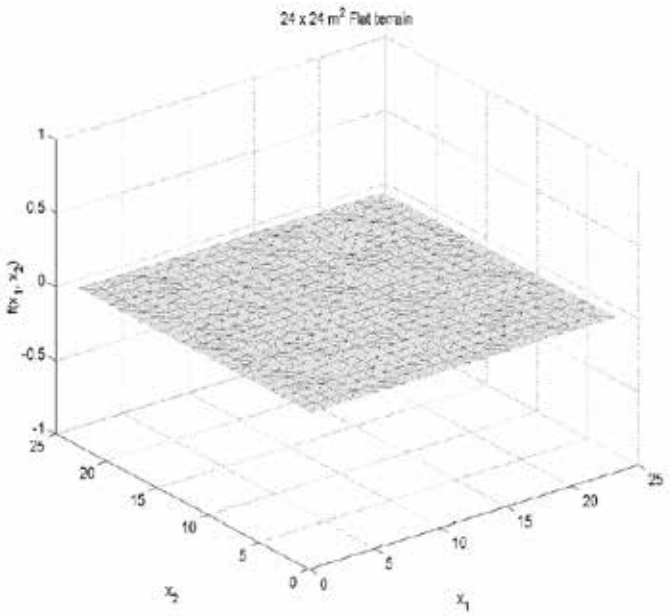


Fig. 4. Flat terrain after Delaunay triangulation.

The parameter values for the 24 × 24 m^2 flat terrain case are given in Table 1. They include the rover vehicle and mission specifications based on the NASA/JPL FIDO rover. Based on these data, the rover is capable of traveling a distance of 1 m in 17.99 seconds. These values

have been chosen to expedite the computer simulations and although they may exceed the actual attainable values, they can be scaled accordingly.

Maximum rover velocity	$v_{max} = 0.0556\ m/sec$
Maximum power constraint	$P_{max} = 100$ W
Maximum traversable slope	$\theta_{max} = 1.30$ radians
Rover mass	$M = 70$ kg
Mars gravity	$g_m = 3.7146\ m/sec^2$
Sample analysis time	$\tau_{wait} = 10$ seconds
Maximum mission time	$\tau_{max} = 500$ seconds

Table 1. Rover variables, flat terrain case.

The starting position is set at one of the four corners of the terrain. The maximal attainable sets corresponding to the maximum mission time $\tau_{\max}$ = 500 seconds starting from each of the four corner points are shown in Fig. 5. The maximal attainable set for each corner can cover most of the terrain, and the union of the maximal attainable sets starting from all four corner points is the entire surface. Thus, it is possible to produce a sufficiently large number of paths for single and multiple rovers.

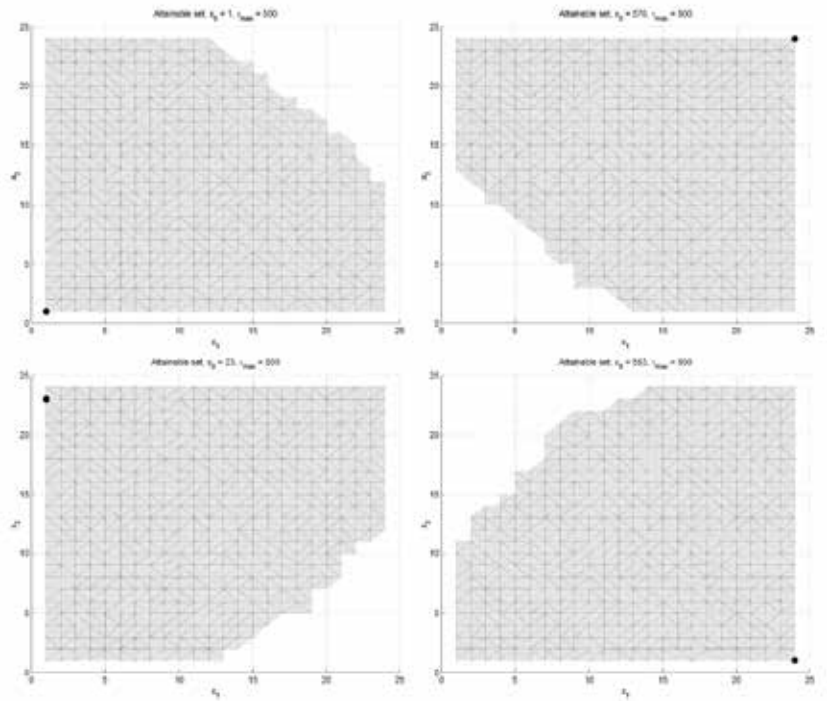


Fig. 5. Maximal attainable sets for the four different starting positions for the flat terrain, single rover case.

On this surface, we distribute 45 samples, as shown in Fig. 6. The samples are distributed by first dividing the terrain into five distinct regions. Each region is assigned a weight for each of three sample types. These weights are used to determine how many samples of each type to assign to each region. Once the number of samples for each region and type are determined, the samples are distributed uniformly. Samples are allowed to occupy the same spatial point on the terrain. There are three sample types with values 1, 4, and 9. The sample index numbers, x-references, and values are listed in Table 2. The soil samples with higher values have relatively smaller population.

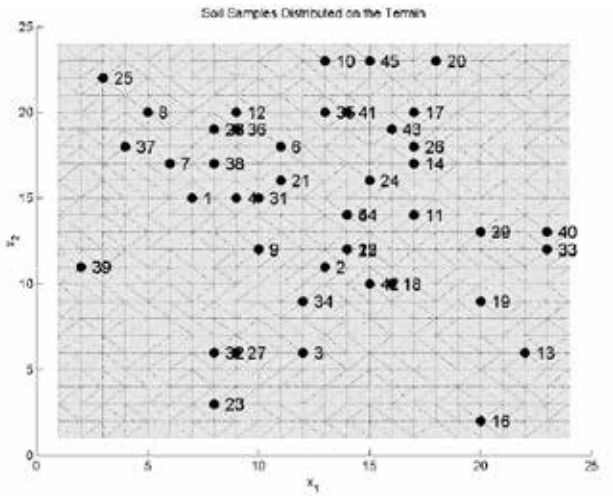


Fig.6. Sample distribution on 24 × 24 m²flat terrain.

Index	x-Reference	Value	Index	x-Reference	Value
1	159	1	24	352	4
2	299	1	25	70	4
3	270	1	26	402	4
4	207	1	27	198	4
5	326	1	28	187	4
6	258	1	29	469	4
7	137	1	30	469	4
8	116	1	31	231	4
9	228	1	32	174	4
10	311	1	33	540	4
11	398	1	34	273	4
12	212	1	35	308	4
13	510	1	36	211	9
14	401	1	37	90	9
15	324	1	38	185	9
16	458	1	39	35	9
17	404	1	40	541	9
18	370	1	41	332	9
19	465	1	42	346	9
20	431	1	43	379	9
21	256	4	44	326	9
22	324	4	45	359	9
23	171	4			

Table 2. Sample data for flat terrain case.

4.1.1 Single Rover Case

For the single rover case, we consider four different starting positions at each of the four corners of the terrain. We then solve the SCP for each starting position by using the "maximum value heuristic" and "3-opt switching". The resulting paths are given in Fig. 7. The sample collection lists are given in Table 3. The elapsed times for each vehicle are very close to the maximum mission time. We observe that there is a higher concentration of high-valued samples in the upper region of the terrain, so it makes sense that the rovers starting in this area result in a higher value of the mission return function. Out of the four starting positions, the one starting at the top left ($x_0 = 23$) gives the highest mission value with $V = 57$. A close examination of the decision process made by the rover for each step of the sample collection path is given in (Cardema *et al*, 2003).

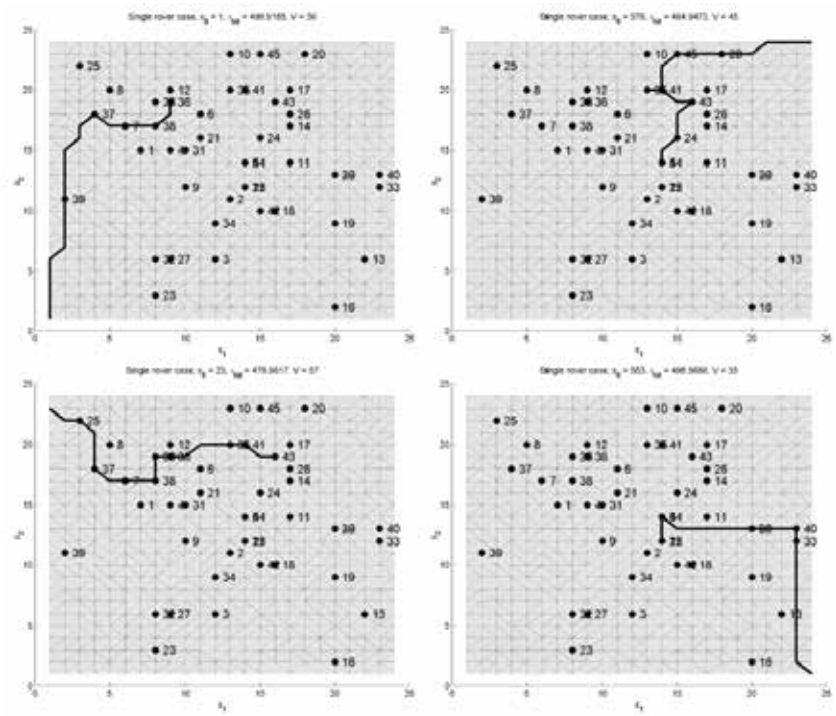


Fig.7. Four different starting positions for the flat terrain, single rover case.

x_0	$\mathcal{L}_{ss}$	τ_c	$V(\mathcal{L}_{ss})$
1	{39, 37, 38, 36}	490.92	36
576	{45, 41, 35, 43, 24, 5, 44}	484.95	45
23	{25, 37, 38, 28, 36, 35, 41, 43}	476.96	57
553	{33, 40, 29, 30, 5, 44, 22}	498.57	35

Table 3. Sample collection lists, flat terrain, single rover case.

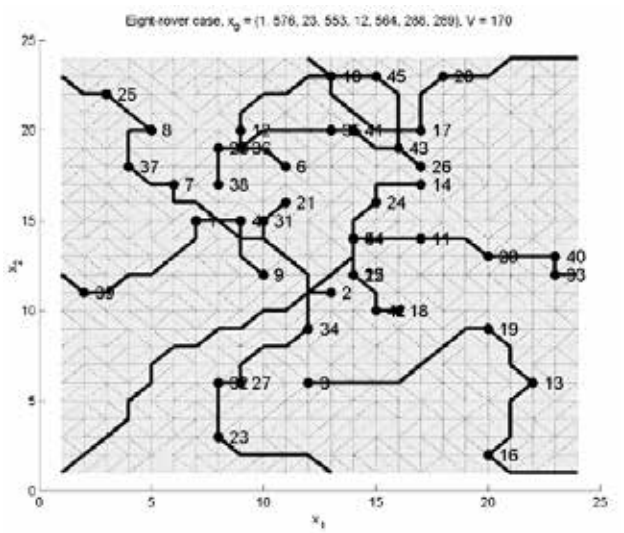


Fig. 8. Paths for the flat terrain, eight-rover case.

x_0	$\mathcal{L}_{ss}$	τ_c
1	{5, 24, 14}	489.65
576	{20, 17, 10, 12, 6}	497.83
23	{25, 8, 37, 7, 2}	433.34
553	{16, 13, 19, 3}	441.33
12	{39, 1, 4, 9}	293.08
564	{33, 40, 29, 30, 11, 44, 15, 22, 42, 18}	402.67
288	{45, 43, 26, 41, 35, 36, 28, 38}	466.43
289	{23, 32, 27, 34, 31, 21}	471.86

Table 4. Sample collection lists, flat terrain, eight-rover case.

4.1.2 Eight-Rover Case

The foregoing computations are also performed for two, four and eight-rover cases. For brevity, only the results for the eight-rover case are presented here. The results for other cases are described in (Cardema *et al*, 2003). In the eight-rover case, we try to employ a symmetric configuration for the rover starting positions to give each rover equal opportunity at sample collection. From Fig. 8, we observe that all the samples have been collected.

If we take the best values from the single rover (V = 57), two-rover (V = 92), four-rover cases (V = 138), and eight-rover cases (V = 170), the plot of mission value versus the number of rovers is sub-linear. If we take the worst values from the single rover (V = 35), two-rover (V = 71), four-rover (V = 138), and eight-rover (V = 170) cases, then the plot is close to linear, but diverges after four rovers. The plot of performance versus number of rovers is shown in Fig. 9. The linear projection shown in the figure is based on the lowest-valued single rover case.

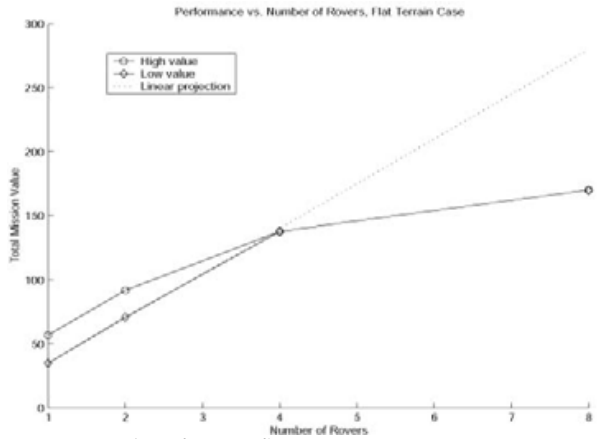


Fig.9. Performance vs. number of rovers, flat terrain case.

4.2. Mars Terrain

We now consider a 24 × 24 m² section of Mars terrain data obtained from the Mars Orbiter Laser Altimeter (MOLA) Science Investigation. This region is located near the Valles Marineris. It is chosen since the surface is smooth enough to facilitate rover movement, but has surface variations to provide an interesting example. The terrain is shown in Fig. 10. We define this as a 24 × 24 m² section, although the actual dimensions are much larger. The height has also been scaled down to facilitate rover movement. This is meant to be an illustrative example rather than a realistic one. The variables for the Mars terrain case are identical to those in Table 1, except here the maximum mission time $\tau_{\max}$ is 720 seconds.

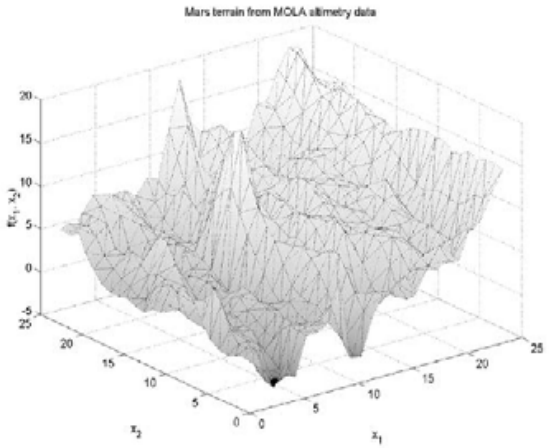


Fig.10. Mars terrain after Delaunay triangulation.

The four possible starting positions along with the maximal attainable sets based on the maximum mission time $\tau_{\max}$ = 720 seconds are shown in Fig. 11. The mission time has been extended from the flat case so that the maximal attainable sets overlap, but the starting positions remain the same. The overlapping region of the maximal attainable sets of the four rovers is shown in Fig. 12. The overlapping region is the set of all points that can be reached by two or more rovers. In this case, the overlapping region covers the entire area.

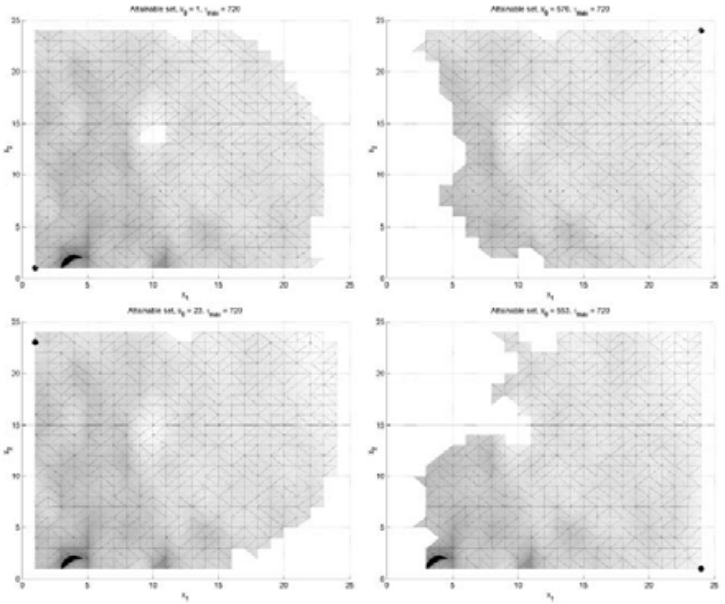


Fig.11. Maximal attainable sets for the four different starting positions for the Mars terrain, single rover case.

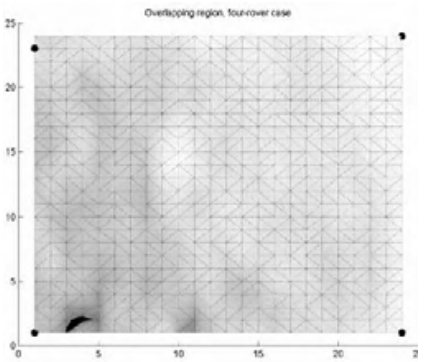


Fig. 12. Overlapping regions of the Mars terrain case.

On this surface, we distribute 45 samples, as shown in Fig. 13. The samples are distributed in the same way as in the flat terrain case. The index numbers, x-references, and values of the samples are listed in Table 5.

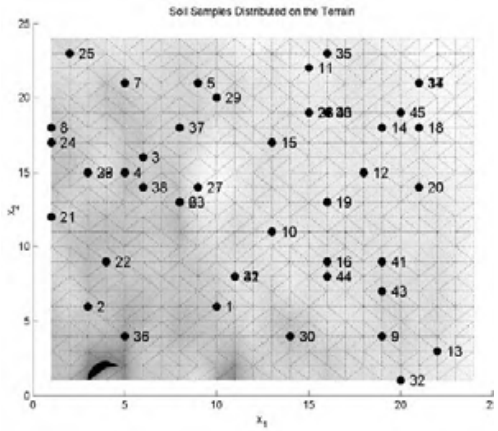


Fig.13. Sample distribution on Mars terrain.

Index	x-Reference	Value	Index	x-Reference	Value
1	222	1	24	17	4
2	54	1	25	47	4
3	136	1	26	63	4
4	111	1	27	206	4
5	213	1	28	355	4
6	181	1	29	236	4
7	117	1	30	316	4
8	18	1	31	248	4
9	436	1	32	457	4
10	299	1	33	379	4
11	358	1	34	501	4
12	423	1	35	383	4
13	507	1	36	100	9
14	450	1	37	186	9
15	305	1	38	134	9
16	369	1	39	63	9
17	501	1	40	379	9
18	498	1	41	441	9
19	373	1	42	248	9
20	494	1	43	439	9
21	12	4	44	368	9
22	81	4	45	475	9
23	181	4			

Table 5. Sample data for Mars terrain case.

4.2.1 Single Rover Case

For the single rover case, we consider four different starting positions and solve the SCP for each location by using the "maximum value heuristic" and "3-opt switching". The resulting graphs are given in Fig. 14. The sample collection lists are given in Table 6. Note that the rovers' paths tend to lie in the valleys and on the flatter regions. Out of these starting positions, the one starting at x_0 = 432 gives the highest mission value $V = 32$.

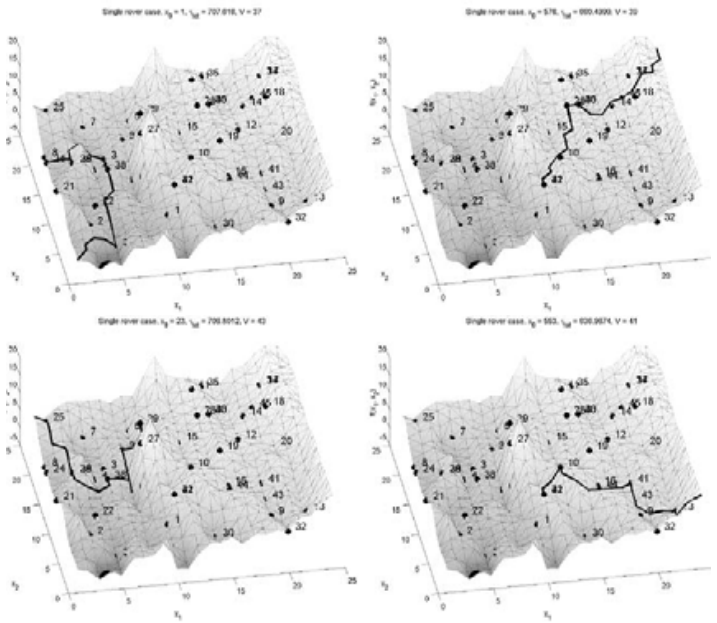


Fig. 14. Paths for four different starting positions for the Mars terrain, single rover case.

x_0	$\mathcal{L}_{**}$	τ_*	$V(\mathcal{L}_{**})$
1	{36, 38, 3, 39, 26, 24, 8}	707.62	37
576	{45, 40, 33, 28, 42, 31}	660.50	39
23	{25, 39, 26, 38, 23, 37, 29}	706.80	43
553	{43, 41, 44, 10, 31, 42}	636.97	41

Table 6. Sample collection lists, Mars terrain, single rover case.

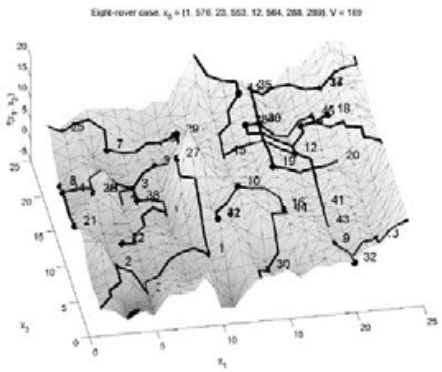


Fig. 15. Paths for Mars terrain, eight rover case.

x_0	$\mathcal{L}_{ss}$	τ_c
1	{2, 36, 1, 27}	690.12
576	{34, 17, 35, 19, 20}	692.27
23	{25, 7, 5, 29, 37, 4}	609.00
553	{13, 32, 9}	335.96
12	{21, 24, 26, 39, 3, 38, 6, 23, 22}	651.79
564	{45, 40, 33, 28, 41, 43}	676.74
288	{11, 15, 12, 14, 18}	641.02
289	{30, 44, 16, 10, 42, 31}	519.73

Table 7. Sample collection lists, Mars terrain, eight-rover case.

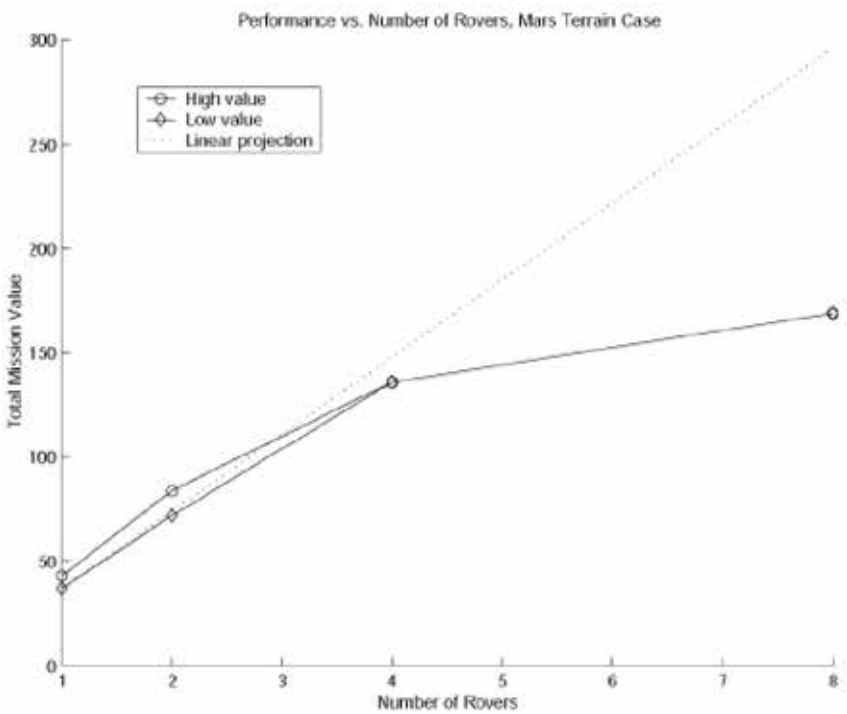


Fig. 16. Performance vs. number of rovers, Mars Terrain case.

4.2.2 Eight-Rover Case

Again, as in the flat terrain case, we only present results for the eight-rover case. The sample collection lists are given in Table 7. The results for two and four-rover cases are given in (Cardema *et al*, 2003). The starting positions in the Mars terrain for the eight-rover case are the same as those in the flat terrain case. All the samples

except for σ_8 (λ_8 = 1) have been collected. One high-valued sample in particular, σ_{27} (λ_{27} = 9), is only collected in this eight-rover case. This makes sense since the sample is located close to the highest peak. The rover starting at $x_0 = 1$ spends a long time climbing up to collect σ_{27} and only collects four samples as a result. Note that the rover starting at $x_0 = 553$ only collects three samples and has a very short mission time of 335.96 sec, although several high-valued samples are easily within reach. The algorithm does not try to distribute the task evenly, which may be a hindrance to higher performance. However, the results provide some insight on the nature of solutions to the SCP for multiple rovers. If we take the best values from the single rover (V = 43), two-rover (V = 84), four-rover (V = 136), and eight-rover cases (V = 169), the plot of mission value versus the number of rovers is sub-linear. If we take the worst values from the single rover (V = 37), two-rover (V = 72), four-rover (V = 136), and eight-rover cases (V = 169), the plot is again close to linear when the number of rovers is less than 4. Note that the spacing between the upper and lower bounds in the Mars terrain case is smaller than that of the flat terrain case.

When examining the performance versus the number of rovers, it makes sense that the graph is near-linear or sub-linear, since once the best path for the first rover has been found; there is not much room for improvement. In Fig 17, we observe that the flat terrain and Mars terrain results are near-linear or sub-linear and they are also very similar. The spacing between the upper and lower bounds for the Mars terrain is smaller than that of the flat terrain. This is due to the longer traveling times between samples in the Mars terrain case. The relatively close spacing of the rovers in our examples helps to ensure that nearly all the samples of interest are collected within the elapsed mission time, but may not be the best placement for obtaining high values for the mission return function. If higher values are desired, it may be better to have no interaction at all. In that case, the rovers are deployed in different areas resulting in independent single-rover cases. Since the rovers are not competing for the same high-valued soil samples, the resulting values for the mission return function may be higher than in the case where the rovers cooperate.

5. Remarks on Further Studies

The approach to optimal path planning studied here is an initial but nonetheless essential step toward the long-term goal of developing autonomous rovers that are capable of analyzing sensory data and selecting the optimal path to take based on autonomous assessment of the relative scientific value of the possible sampling sites in rovers' field of view. Such a scientific-value-driven autonomous navigation capability presents formidable challenges in autonomous rover development. One of the key challenges is how to assign relative scientific value to possible samples using data from the onboard sensors, and update the values on the basis of information that has been gathered at previous scientific sites. Sample selection is done very well by scientists on the ground, based on their extensive experience in field geology, but capturing their expertise in a set of algorithms is difficult.

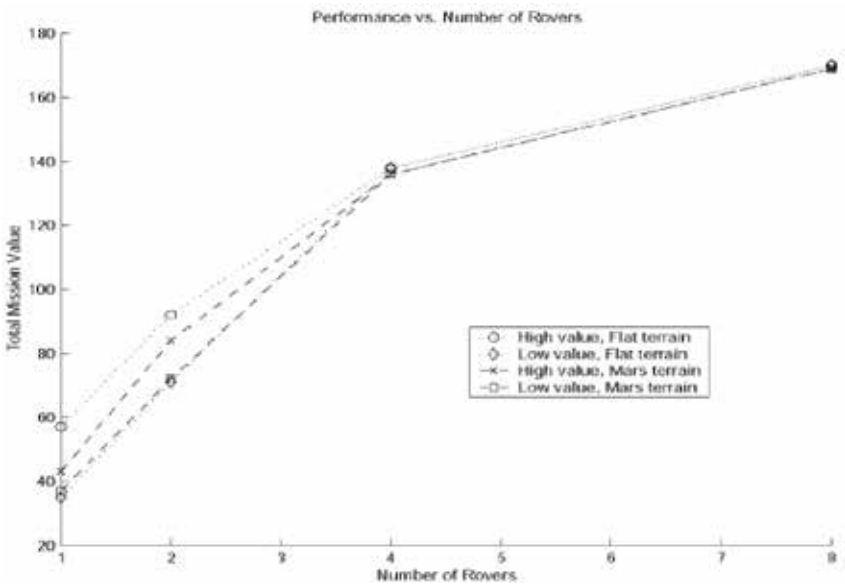


Fig. 17. Performance vs. number of rovers.

There have been some interesting studies aimed at selecting geological sampling sites and performing data acquisition autonomously, such as those performed by (Pedersen *et al,* 2001) and by (Huntsberger *et al*, 2002). But this area of study is in its infancy, and it will take some time to mature to the point that it can be considered for operational rovers. When it does become possible for rovers to automatically select the soil samples of interest, the path-planning problem will become a closed-loop process. When the rover initializes, it will perform a sensor scan of the surrounding area to create a three-dimensional terrain map, locate potential soil samples, evaluate their relative values, and formulate an initial path. As the rover navigates through the terrain, it can update its plan as it discovers new soil samples or alters the value of already detected ones. Mission performance will depend on the quality of the sensors, which affects the maximum detection range, sample localization, and accuracy of sample valuation.

The metric used in this study for mission performance was the total value of the collected soil samples. Instead of using the total sample value, other objectives could include minimizing the total collection time of a given number of samples, maximizing the probabilities of detecting interesting samples, or maximizing the total coverage of a given area.

During the past and on-going Mars missions, rovers typically received a new set of instructions sent daily from scientists and engineers on Earth. The rover was expected to move over a given distance, position itself with respect to a target, and deploy its instruments to take close-up pictures and analyze the minerals or composition of rocks and soil. For operational scenarios involving multiple rovers as considered in this study, the

above-mentioned challenges become even more formidable because of the need for coordinated path planning and execution.

The use of multiple rovers to aid sample collection leads to new interesting problems. In the MER mission, two rovers were used, but they were deployed in two separate locations. Consequently, the path-planning problem reduces essentially to two single-rover path-planning problems. In this work, we have begun to develop cooperative path-planning algorithms for interacting multiple rovers using our "best path first" and "partition of overlapping sets" heuristics. But this approach can also be viewed as the decomposition of the multiple-rover path-planning problem into multiple single-rover ones. Cooperative algorithms could be used instead. If the process is to be automated, communication between the rovers is critical in updating each one's knowledge of the terrain and soil samples. Each rover receives updates from the others, recomputes its optimum path, and makes adjustments as necessary. There are many issues to be resolved. A basic issue is to determine how close should the rovers be for maximum operational efficiency. Evidently, they should be sufficiently close so that the information they collect are relevant to the others, but not close enough to interfere with each other's actions. Another issue is to determine how should the tasks be divided. One can imagine a strategy where the rovers with different capabilities can be used for specialized functions, such as using a small fast rover only for gathering information about the terrain. This information is then relayed to the larger, more equipped rovers to perform the sample collection. The strategy used will depend on the mission objective (e.g. maximum value of soil samples collected versus maximum area covered by the rovers). Once the objective and strategy for multiple rovers have been determined, another interesting sub-problem is to find the optimal number of rovers to use.

The study of the multiple-rover problem would be similar to the work outlined here. Models would be developed to describe the planetary surface, each rover's dynamics, and the sensor capabilities and operation. A general framework should be implemented to serve as a test-bed for various multiple rover objectives and strategies, allowing for case studies involving different algorithms, sensor properties, surface conditions, and the number and types of rovers used.

The solution of the Sample Collection Problem (modified Traveling Salesman Problem) for both single and multiple rovers also presents some room for improvement. Besides the heuristic methods presented here, additional methods that could be explored include simulated annealing, genetic algorithms or other global optimization techniques.

6. Conclusion

In this work, we gave mathematical formulations of the sample collection problem for single and multiple robots as optimization problems. These problems are more complex than the well-known NP-hard Traveling Salesman Problem. In order to gain some insight on the nature of the solutions, algorithms are developed for solving simplified versions of these problems. This study has been devoted to centralized operation. If communication between the rovers is considered, as in autonomous operation, the nature of the result will be different. The problem posed here is simplified to facilitate mathematical formulation. To determine whether the strategies and algorithms discussed in this paper

can be applied to practical situations, extensive testing must be done with actual rovers on various terrains. The formulation presented in this paper could be used as a framework for future studies. In particular, the autonomous case discussed briefly in this paper deserves further study.

7. Appendix

Traveling Salesman Problem:

The problem of soil sample collection is an instance of the well-known Traveling Salesman Problem (TSP). In this problem, a traveling salesman is required to visit n cities before returning home (Evans & Minieka, 1992). He would like to plan his trip so that the total distance traveled is as small as possible. Let G be a graph that contains the vertices that correspond to the cities on the traveling salesman's route, and the arcs correspond to the connections between two cities. A cycle that includes each city in G at least once is called a *salesman cycle*. A cycle that includes each city in G exactly once is called a *Hamiltonian cycle* or *traveling salesman tours*. The TSP is NP-hard since the solution time increases exponentially with the number of cities *n*. Although there does not exist efficient algorithms to solve the TSP, it is nevertheless studied in depth because of its simplicity. For small values of n, each possible route can be enumerated and the one with the least total distance is the exact optimum solution. For large *n*, it becomes time-consuming and memory-intensive to enumerate each possibility. Thus, it becomes necessary to make use of heuristics to obtain near-optimal solutions. A few tour construction heuristics are described briefly in the sequel.

A1. Nearest-neighbor heuristic

Let $d(x, y)$ denote the distance between cities *x* and *y* and. In this heuristic, we begin at the starting point x_0 and find the next city x_1 such that $d(x_0, x_1)$ is minimized. Then, from x_1, find the next nearest neighbor x_2 that minimizes $d(x_1, x_2)$. We continue this process until all the cities have been visited. The last arc is from city x_n back to x_0, where *n* is the total number of cities visited. This heuristic rarely leads to the optimal solution.

A2. Nearest-insertion heuristic

Starting from x_0, we choose the nearest city x_1 and form the sub-tour $x_0 \to x_1 \to x_0$. At each iteration, find the city x_m not in the sub-tour but closest to the cities in the sub-tour that minimizes $d(x_0, x_m) + d(x_m, x_1) - d(x_0, x_1)$. Then x_m is inserted between x_0 and x_1. This insertion process is repeated with the next closest city and continued until all the cities have been visited. This method slowly builds on the original sub-tour by minimizing the distance added at each iteration.

A3. k-opt tour improvement heuristics

Given a traveling salesman tour, a *k*-opt tour improvement heuristic will change the ordering of up to *k* cities to find a more optimal solution. For example, if the original tour of 4 cities is 1-2-3-4-1, 2-opt switching will try all possible combinations of 2 switches (1-3-2-4-1, 1-4-3-2-1, 1-2-4-3-1) and keep the tour with the smallest total distance. For $k < n$, the

k -opt heuristic will take less time to implement than enumerating all possible orderings of n cities.

8. Acknowledgment

This work was partially supported under Contract 1243467 with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California.

9. References

Baglioni, P., Fisackerly, R., Gardini, B., Giafiglio, G., Pradier, A.L., Santovincenzo, A., Vago, J.L. & Van Winnendael, M. (2006). The Mars Exploration Plans of ESA (The ExoMars Mission and the Preparatory Activities for an International Mars Sample Return Mission), *IEEE Robotics & Automation Magazine*, Vol. 13, No.2, pp. 83-89.

Biesiadecki, J.J. & Maimone, M.W. (2006). The Mars Exploration Rover Surface Mobility Flight Software Driving Ambition. Proc. IEEE Aerospace Conf. March pp.1-15.

Cardema, J.C., Wang, P.K.C. & Rodriguez, G. (2003). Optimal Path Planning of Mobile Robots for Sample Collection, UCLA Engr. Report, No. UCLA ENG-03-237, June.

Cheng, Y., Maimone, M.W. & Matthies, L., (2006). Visual Odometry on the Mars Exploration Rovers (A Tool to Ensure Accurate Driving and Science Imaging), *IEEE Robotics & Automation Magazine*, Vol. 13, No.2, pp. 54-62.

Evans, J.R. & Minieka, E. (1992). Optimization Algorithms for Networks and Graphs, Second Edition, Marcel Dekker, Inc., Monticello, New York.

Huntsberger, T.; Aghazarian, H.; Cheng, Y.; Baumgartner, E.T.; Tunstel, E.; Leger, C.; Trebi-Ollennu, A. & Schenker, P.S. (2002). Rover Autonomy for Long Range Navigation and Science Data Acquisition on Planetary Surfaces, Proc. IEEE International Conference on Robotics and Automation, Wash. DC, May.

Malin, M.C. & Edgett, K.S. (2000). Sedimentary Rocks of Early Mars, *Science Magazine*, Vol. 290.

Mars Pathfinder Homepage, http://nssdc.gsfc.nasa.gov/planetary/mesur.html.

Mars Spirit & Opportunity rovers Homepage, http: //marsrovers.jpl.nasa.gov/home/

Naderi, F., McCleese, D.J. & Jordan, Jr, J.F. (2006). Mars Exploration (What Have We Learned? What Lies Ahead?), *IEEE Robotics & Automation Magazine*, Vol. 13, No.2, pp. 72-82.

Pedersen, L., Wagner, M.D., Apostolopoulos, D. & Whittaker, W.L. (2001). Autonomous Robotic Meteorite Identification in Antarctica, IEEE International Conf. on Robotics and Automation, May, pp. 4158-4165.

Seraji, H., (2000). Fuzzy Traversability Index: A New Concept for Terrain-Based Navigation, *J. Robotic Systems*, Vol. 17-2, pp. 75-91.

Wang, P.K.C. (2003). Optimal Path Planning Based on Visibility, *J. Optimization Theory and Applications*, Vol. 117-1, pp. 157-181.

Wang, P.K.C. (2004). Optimal Motion Planning for Mobile Observers Based on Maximum Visibility, Dynamics of Continuous, Discrete and Impulsive Systems, Vol. 11b, pp.313-338.

Weisstein, E.W. Delaunay Triangulation, Eric Weisstein's World of Mathematics, http://mathworld.wolfram.com

Wright, J., Hartman, F., Cooper, B.; Maxwell, S., Yen, J. & Morrison, J. (2006). Driving on Mars with RSVP (Building Safe and Effective Command Sequences), *IEEE Robotics & Automation Magazine*, Vol. 13, No.2, pp. 37-45.

29

Multi Robotic Conflict Resolution by Cooperative Velocity and Direction Control

Satish Pedduri+
K Madhava Krishna+
+ *International Institute of Information Technology, Hyderabad, India*

1. Introduction

Collision avoidance is one of the essential pillars of a wheeled robotic system. A wheeled mobile robot (called mobile robot for conciseness henceforth) requires for effective functioning an integrated system of modules for (i) map building, (ii) localization, (iii) exploration, (iv) planning and (v) collision avoidance. Often (i) and (ii) are entailed to be done simultaneously by robots resulting in a vast array of literature under the category SLAM, simultaneous localization and mapping. In this chapter we focus on the aspect of collision avoidance specifically between multiple robots, the remaining themes being too vast to even get a brief mention here.

We present a cooperative conflict resolution strategy between multiple robots through a purely velocity control mechanism (where robots do not change their directions) or by a direction control method. The conflict here is in the sense of multiple robots competing for the same space over an overlapping time window. Conflicts occur as robots navigate from one location to another while performing a certain task. Both the control mechanisms attack the conflict resolution problem at three levels, namely (i) individual, (ii) mutual and (iii) tertiary levels. At the individual level a single robot strives to avoid its current conflict without anticipating the conflicting robot to cooperate. At the mutual level a pair of robots experiencing a conflict mutually cooperates to resolve it. We also denote this as mutually cooperative phase or simply cooperative phase succinctly. At tertiary level a set of robots cooperate to avoid one or more conflicts amidst them. At the tertiary level a robot may not be experiencing a conflict but is still called upon to resolve a conflict experienced by other robots by modifying its velocity and (or) direction. This is also called as propagation phase in the chapter since conflicts are propagated to robots not involved in those. Conflicts are resolved by searching the velocity space in case of velocity control or orientation space in case of direction control and choosing those velocities or orientations that resolve those conflicts. At the individual level the search is restricted to the individual robot's velocity or direction space; at the mutual level the search happens in the velocity or direction space of the robot pair experiencing the conflict and at tertiary levels the search occurs in the joint space of multiple robots. The term cooperative is not a misnomer for it helps in achieving the following capabilities:

1 Avoid collision conflicts in a manner that conflicting agents do not come too near while avoiding one and another whenever possible. Thus agents take action in a fashion that benefits one another apart from avoiding collisions.
2 Provides a means of avoiding conflicts in situations where a single agent is unable to resolve the conflict individually.
3 Serves as a pointer to areas in the possible space of solutions where a search for solution is likely to be most fruitful.

The resolution scheme proposed here is particularly suitable where it is not feasible to have a-priori the plans and locations of all other robots, robots can broadcast information between one another only within a specified communication distance and a robot is restricted in its ability to react to collision conflicts that occur outside of a specified time interval called the reaction time interval. Simulation results involving several mobile robots are presented to indicate the efficacy of the proposed strategy.

The rest of the chapter is organized as follows. Section 2 places the work in the context of related works found in the literature and presents a brief literature review. Section 3 formulates the problem and the premises based on which the problem is formulated. Section 4 mathematically characterizes the three phases or tiers of resolution briefly mentioned above. Section 5 validates the efficacy of the algorithm through simulation results. Section 6 discusses the limitations of the current approach and its future scope and ramifications and section 7 winds up with summary remarks.

2 Literature Review

Robotic navigation for single robot systems has been traditionally classified into planning and reactive based approaches. A scholarly exposition of various planning methodologies can be found in (Latombe 1991). A similar exposition for dynamic environments is presented by Fujimora (Fujimora 1991). Multi-robot systems have become an active area of research since they facilitate improved efficiency, faster responses due to spread of computational burden, augmented capabilities and discovery of emergent behaviors that arise from interaction between individual behaviors. Multiple mobile robot systems find applications in many areas such as material handling operations in difficult or hazardous terrains (Genevose at. al, 1992)[3], fault-tolerant systems (Parker 1998), covering and exploration of unmanned terrains (Choset 2001), and in cargo transportation (Alami et. al, 1998). Collaborative collision avoidance (CCA) between robots arises in many such multi-robot applications where robots need to crisscross each other's path in rapid succession or come together to a common location in large numbers. Whether it is a case of navigation of robots in a rescue and relief operation after an earthquake or while searching the various parts of a building or in case of a fully automated shop floor or airports where there are only robots going about performing various chores, CCA becomes unavoidable.

Multi-robotic navigation algorithms are traditionally classified as centralized or decentralized approaches. In the centralized planners [Barraquand and Latombe 1990, Svetska and Overmars 1995] the configuration spaces of the individual robots are combined into one composite configuration space which is then searched for a path for the whole composite system. In case of centralized approach that computes all possible conflicts over entire trajectories the number of collision checks to be performed and the planning time tends to increase exponentially as the number of robots in the system increases. Complete recalculation of paths is required even if one of the robot's plans is altered or environment

changes. However centralized planners can guarantee completeness and optimality of the method at-least theoretically.

Decentralized approaches, on the other hand, are less computationally intensive as the computational burden is distributed across the agents and, in principle, the computational complexity of the system can be made independent of the number of agents in it at-least to the point of computing the first individual plans. It is more tolerant to changes in the environment or alterations in objectives of the agents. Conflicts are identified when the plans or commands are exchanged and some kind of coordination mechanism is resorted to avoid the conflicts. However, they are intrinsically incapable of satisfying optimality and completeness criterion. Prominent among the decentralized approaches are the decoupled planners [Bennewitz et. al, 2002], [Gravot and Alami 2001], [Leroy et. al 1999]. The *decoupled* planners first compute separate paths for the individual robots and then resolve possible conflicts of the generated paths by a hill climbing search [Bennewitz et. al, 2004] or by plan merging [Gravot and Alami 2001] or through dividing the overall coordination into smaller sub problems [Leroy et. al 1999].

The method presented here is different in that complete plans of the robots are not exchanged. The locations of the robots for a certain T time samples in future are exchanged for robots moving along arcs and for those moving with linear velocities along straight lines it suffices to broadcast its current state. The collisions are avoided by searching in the velocity or the orientation space (the set of reachable orientations) of the robot. In that aspect it resembles the extension of the Dynamic Window approach [Fox et. al, 1997] to a multi robotic setting however with a difference. The difference being that in the dynamic window the acceleration command is applied only for the next time interval whereas in the present method the restriction is only in the direction of change in acceleration over a time interval $t < T$ for all the robots.

The present work is also different from others as the resolution of collision conflicts is attempted at three levels, namely the individual, cooperative, and propagation levels. Functionally cooperation is a methodology for pinning down velocities or orientations in the joint solution space of velocities or orientations of the robots involved in conflict when there exists no further solution in the individual solution space of those robots. When joint actions in the cooperative phase are not sufficient for conflict resolution assistance of other robots that are in a conflict free state at that instant is sought by the robots in conflict by propagating descriptions of the conflicts to them. When such free robots are also unable to resolve the conflict collision is deemed inevitable. The concept of propagating conflict resolution requests to robots not directly involved in a conflict is not found mentioned in robotic literature. Such kind of transmission of requests to robots though not invoked frequently is however helpful in resolving a class of conflicts that otherwise would not be possible as our simulation results reveal.

The method presented here is more akin to a real-time reactive setting where each robot is unaware of the complete plans of the other robots and the model of the environment. The work closest to the present is a scheme for cooperative collision avoidance by Fujimora's group (Fujimora et. al, 2000) and a distributed fuzzy logic approach as reported in (Srivastava et. al, 1998). Their work is based on devising collision avoidance for two robots based on orientation and velocity control and extend this strategy for the multi robot case based on the usual technique of priority based averaging (PBA). However we have proved in an earlier effort of ours (Krishna and Kalra, 2002) that such PBA techniques fail when individual actions that get weighted and averaged in the PBA are conflicting in nature. The

work of Lumelsky (Lumelsky and Harinarayanan 1998) is of relation here in that it does not entail broadcast of plans to all other robots. It describes an extension of one of the *Bug* algorithms to a multi robotic setting. There is not much mention of cooperation or collaborative efforts between the robots except in the limited sense of "reasonable behavior" that enables shrinking the size of collision front of a robot that is sensed by another one.

3 Objective, Assumptions and Formulations:

Given a set of robots $R = \{R_1, R_2, \ldots, R_n\}$, each assigned a start and goal configuration the objective is to navigate the robot such that they reach the goal configuration avoiding all collisions.

While collisions could be with stationary and moving objects in this chapter we focus specifically how the robots could avoid collisions that occur amongst them in a cooperative fashion. For this purpose the following premises have been made:

a. Each robot *Ri* is assigned a start and goal location and it has access to its current state and its current and aspiring velocities. The current state of *Ri* is represented as $\psi_i = \{vc_i, vn_i, \theta c_i, \theta n_i\}$ where vc, vn represent its current and aspiring velocities and $\theta c, \theta n$ its current and aspiring directions. The aspiring direction to be reached at a given time *t* is the angle made by the line joining the current position to the position reached at *t* with the current heading. This is shown in figure 1 where a robot currently at P reaches a point N moving along an arc, the aspiring orientation is the angle made by the dashed line connecting P to N with the current heading direction.
b. All robots are circular and described by their radius
c. Robots are capable of broadcasting their current states to each other. They do so only to those robots that are within a particular range of communication.
d. Robots accelerate and decelerate at constant rates that are same for all. Hence a robot *Ri* can predict, when another robot *Rj* would attain its aspiring velocity vn from its current velocity vc if it does not change its direction.

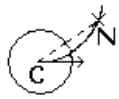


Fig. 1. A robot currently at location C moves along a clothoidal arc to reach position N. The aspiring orientation is computed as mentioned in premise a in text. The heading at C is indicated by the arrow

3.1 Robot Model

We consider a differential drive (DD) mobile robot in consonance with the robots available in our lab. Figure 2a shows an abstraction of a DD robot consisting of two wheels mounted on a common axis driven by two separate motors. Consider the wheels rotating about the current center C, at the rate ω as shown in figure 2. The coordinates of the axis center is (x, y) and the robot's bearing is at θ with respect to the coordinate frame. The distance between the two wheels is L and the radius of curvature of the robot's motion is R (distance from C to robot's center). Given that the left and right wheel velocities are v_l, v_r the following describe the kinematics of the robot:

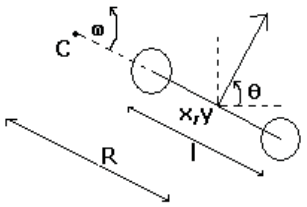


Fig. 2a. A differential drive robot with left and right wheels driven by two motors that are independently controlled.

$$\omega = \frac{v_r - v_l}{l} \quad \cdots\cdots \quad (3.1.1); v = \frac{v_r + v_l}{2} \quad \cdots\cdots \quad (3.1.2); R = \frac{l(v_r + v_l)}{2(v_r - v_l)} \quad \cdots\cdots \quad (3.1.3)$$

Here v, ω represent the instantaneous linear and angular velocity of the robot. Given the current configuration of the robot's center as x_0, y_0, θ_0 at t_0 the coordinates reached by the robot at time *t* under constant linear and angular acceleration (a, α) is given by

$$x = x_0 + \int_{t_0}^{t} (v(t_0) + at)\cos\left(\theta_0 + \omega t + \alpha t^2\right)dt \quad \cdots\cdots \quad (3.1.4) \text{ and}$$

$$y = y_0 + \int_{t_0}^{t} (v(t_0) + at)\sin\left(\theta_0 + \omega t + \alpha t^2\right)dt \quad \cdots\cdots \quad (3.1.5).$$

Integrals 3.1.4 and 3.1.5 require numerical techniques to compute. Hence in a manner similar to (Fox et.al, 1997) we assume finite sufficiently small intervals for which the velocity is assumed to be a constant and the coordinates of the robot are then computed as

$$x(t_n) = x_0 + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v_i \cos(\theta(t_i) + \omega(t - t_i))dt \quad \cdots\cdots \quad (3.1.6)$$

$$y(t_n) = y_0 + \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} v_i \sin(\theta(t_i) + \omega(t - t_i))dt \quad \cdots\cdots \quad (3.1.7)$$

In the collision avoidance maneuver it is often required to check if the robot can reach to a location that lies on one of the half-planes formed by a line and along a orientation that is parallel to that line. In figure 2b a robot with current configuration x_s, y_s, θ_s with velocity v_s wants to reach a position on the left half plane (LHP) of line *l* along a direction parallel to *l*. For this purpose we initially compute where the robot would reach when it attains either the maximum angular velocity shown in angular velocity profiles of figures 2c and 2d under maximum angular acceleration. The positions reached at such an instant are computed through (3.1.6) and (3.1.7). Let the maximum angular velocity in a given velocity profile as determined by figures 2d and 2e be ω_{aM} and the location reached by the robot corresponding to ω_{aM} be x_{aM}, y_{aM}. ω_{aM} is not necessarily the maximum possible angular velocity ω_M and is determined by the time for which the angular acceleration is applied.

Consider a circle tangent to the heading at x_{aM}, y_{aM} with radius $\frac{v_s}{\omega_{aM}}$, this circle is shown dashed in figure 2c. Consider also the initial circle which is drawn with the same radius but which is tangent to θ_s at x_s, y_s, which is shown solid in 2c. Evidently the initial circle assumes that the robot can reach ω_{aM} instantaneously. Let the displacements in the centers of the two circles be $d_{s,aM}$. Then if the initial circle can be tangent to a line parallel to *l* that is at-least $kd_{s,aM}$ from *l* into its LHP then the robot that moves with an angular velocity profile shown in figures 2d or 2e can reach a point that lies in the LHP of *l* along a direction parallel to *l*. We found *k*=2 to be a safe value. It is to be noted checking on the initial circle is faster since it avoids computing the entire profile of 2d or 2e before concluding if an avoidance maneuver is possible or not.

3.2 The Collision Conflict

With robots not being point objects a collision between two is modeled as an event happening over a period of time spread over an area. The collision conflict (CC) is formalized here for the simple case of two robots moving at constant velocities. The formalism is different if velocity alone is controlled or direction control is also involved. Figure 3 shows the CC formalism when velocity control alone is involved.

Shown in figure 3, two robots R1 and R2 of radii r1 and r2 and whose states are $\psi_1 = (vc_1, vn_1)$ and $\psi_2 = (vc_2, vn_2)$ respectively, where vc_1, vc_2 are the current velocities while vn_1, vn_2 are the aspiring velocities for R1 and R2 respectively. The orientations are omitted while representing the state since they are not of concern here. Point C in the figure represents the intersection of the future paths traced by their centers. For purpose of collision detection one of the robots *R1* is shrunk to a point and the other *R2* is grown by the radius of the shrunken robot. The points of interest are the centers C21 and C22 of R2 where the path traced by the point robot *R1* becomes tangential to *R2*. At all points between C21 and C22 *R2* can have a potential collision with *R1*. C21 and C22 are at distances $(r1+r2)\cos ec(|\theta_1 - \theta_2|)$ on either side of C. The time taken by *R2* to reach C21 and C22 given its current state (vc_2, vn_2) is denoted by t_{21} and t_{22}. Similar computations are made for R1 with respect to R2 by making R2 a point and growing R1 by r2. Locations C11 and C12 and the time taken by R1 to reach them t_{11} and t_{12} are thus computed. A collision conflict or CC is said to be averted between R1 and R2 if and only if $[t_{11}, t_{12}] \cap [t_{21}, t_{22}] \in \Phi$. The locations C11, C12, C21 and C22 are marked in figure1.

A direct collision conflict (*DC*) between robots *R1* and *R2* is said to occur if *R1* occupies a space between C11 and C12 when the center of *R2* lies between C21 and C22 at some time t.

For direction control the CC is formalized as follows. Consider two robots R1 and R2 approaching each other head on as in figure 4a and at an angle in figure 4b. The points at which the robots are first tangent to one another (touch each other exactly at one point) correspond to locations C11 and C21 of R1 and R2's center. The points at which they touch firstly and lastly are marked as P in 4a and P1, P2 in 4b. Let t_{c1}, t_{c2} denote the times at which they were first and lastly tangent to each other. We expand the trajectory of R2 from all points between and including C21 and C22 by a circle of radius r1 while R1 is shrunk to a

point. The resulting envelope due to this expansion of the path from C21 to C22 is marked E. All points outside of E are at a distance r2+r1 from R2's center when it belongs to anywhere on the segment connecting C21 to C22. The envelope E consists of two line segment portions $\overline{E1E2}$, $\overline{E3E4}$ and two arc segment portions $E1A1E4$, $E2A2E3$ shown in figures 4a and 4b. We say a CC is averted if R1 manages to reach a location that is outside of E with a heading θ_a for the time R2 occupies the region from C21 to C22 and upon continuing to maintain its heading guarantees resolution for all future time.

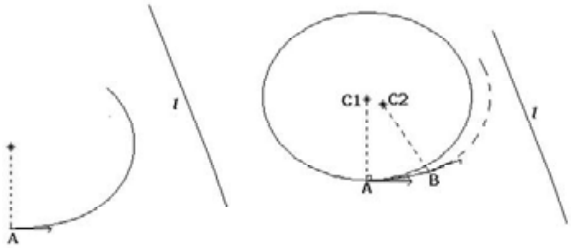


Fig. 2b. A robot at A heading along the direction denoted by the arrow wants to reach a position that lies on the LHP of line *l* along a orientation parallel to *l*. Its angular velocity should reach zero when it reaches a orientation parallel to *l*.
Fig. 2c. In sequel to figure 2b, the robot at A takes off along a clothoidal arc approximated by equations 3.1.6 and 3.1.7 and reaches B with maximum angular velocity. It then moves along a circle centered at C2 shown dotted and then decelerates its angular velocity to zero when it becomes parallel to *l*. The initial circle is drawn centered at C1 tangent to the robot's heading at A. The distance between C1 and C2 decides the tangent line parallel to *l* to which the robot aspires to reach.

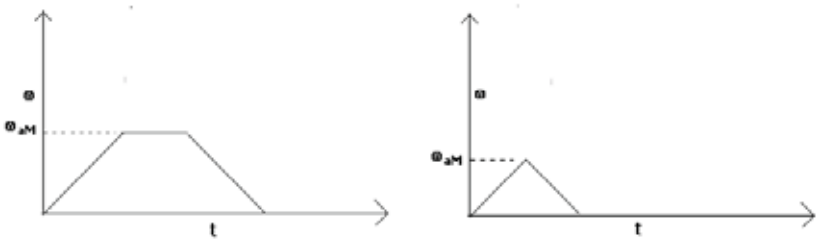


Fig. 2d and 2e. Two possible angular velocity profiles under constant acceleration. Figure 3d corresponds to a path that is a circle sandwiched between two clothoids, while Figure 3e corresponds to the path of two clothoids.

For example in 4a R1 reaches the upper half plane of the segment $\overline{E1E2}$ or the lower half plane of $\overline{E3E4}$ before R2 reaches P then it guarantees resolution for all future times provided R2 does not change its state. Similarly in figure 4b by reaching a point on the lower half plane of $\overline{E3E4}$ with a heading parallel to $\overline{E3E4}$ collision resolution is guaranteed. It is obvious R2 would not want to maintain its heading forever, for it will try to reach its actual destination once the conflict is resolved.

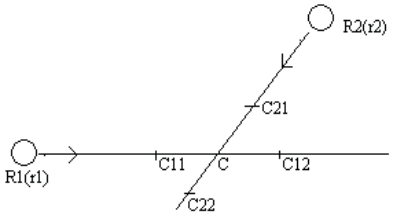


Fig. 3. Two robots *R1* and *R2* with radii r1 and r2 along with their current states are shown. When *R1* is shrunk to a point and *R2* grown by radius of R1, C21 and C22 are centers of *R2* where the path traced by *R1* becomes tangential to *R2*.

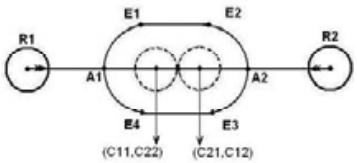


Fig. 4a. Situation where two Robots approaching head on.

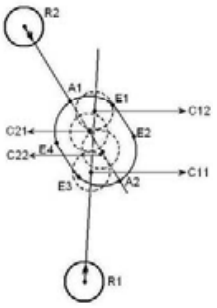


Fig. 4b. Situation where two Robots approaching at an angle.

4 Phases of Resolution

Let S_T be the set of all possible solutions that resolve conflicts among the robots involved. Depending on the kind of control strategy used each member $s_i \in S_T$ can be represented as follows:

i. An ordered tuple of velocities in case of pure velocity control i.e. $s_i = \{v_{1i}, v_{2i}, \ldots, v_{Ni}\}$, for each of the N robots involved in the conflict. Obviously the set s_i is infinite, the subscript *i* in s_i is used only for notational convenience.

ii. An ordered tuple of directions in case of pure direction control i.e. $s_i = \{\theta_{1i}, \theta_{2i}, \ldots, \theta_{Ni}\}$.

iii. An ordered tuple of velocity direction pairs in case of velocity and direction control, $s_i = \{\{v_{1i}, \theta_{1i}\}, \{v_{2i}, \theta_{2i}\}, \ldots, \{v_{Ni}, \theta_{Ni}\}\}$ in case of both velocity and direction control.

Conflicts are avoided by reaching each component of s_i, i.e. the velocities or directions or both within a stipulated time tuplet $\{t_{1i}, t_{2i}, \ldots, t_{Ni},\}$. For purely velocity control the velocities to be attained involved not more than one change in direction of acceleration, i.e., they are attained by an increase or decrease from current acceleration levels but not a combination of both. For purely direction control the final orientation aspired for involves not more than one change in turning direction. However the final direction attained could be through a sequence of angular velocity profiles such as in figures in 2d & 2e that involve only one change in turning direction.

The *cooperative space* is represented by the set $S_C \subseteq S_T$, i.e., the cooperative space is a subset of the total solution space and where every robot involved in the conflict is required to modify its current aspiring velocity or direction to avoid the conflict. In other words robots modify the states in such a manner that each of the robot involved has a part to play in resolving the conflict. Or if any of the robots had not modified its velocity it would have resulted in one or more collisions among the set of robots involved in the conflict.

The *cooperative phase* in navigation is defined by the condition $S_C = S_T$, where each robot has no other choice but to cooperate in order to resolve conflicts. In individual resolution robots choose velocities in the space of $S_I = S_T - S_C$, where the entailment for every robot to cooperate does not exist. When $S_I = \Phi$, the null set, we say the navigation has entered the cooperative phase.

Figures 5a-5d characterize the changes in solution space due to velocity control alone for evolving trajectories of two orthogonal robots while those of 6a-6e do the same for orientation control of robots that approach each other head on. Figure 5a shows evolution of trajectories of two robots, marked R1 and R2, moving orthogonal to one another. The arrows show the location of the two robots at time $t = 0$ sample. The robots move with identical speed of $v_{R1} = v_{R2} = 2.5$ units. The states of the two robots are represented as $\psi_1 = (v_{R1}, v_{R1}, 0, 0)$ and $\psi_2 = (v_{R2}, v_{R2}, -90, -90)$. The equality in the current and aspiring velocities merely indicates that the robot moves with uniform velocity and is not a loss of generality from the case when the aspiring velocity differs from the current. The subsequent discussion holds equally for the case when the current and aspiring velocities differ. Corresponding to this location of the robots at the beginning of their trajectories, figure 5b depicts the total space of velocities bounded within the outer rectangle (shown thick) whose length and breadth are 5 units respectively. In other words each robot can have velocities in the interval $[0,5]$ units. The abscissa represents the range for one of the robots (R1) and the ordinate the range for the other (R2). The center of the figure marked as O indicates the location corresponding to their respective velocities of 2.5units each. The strips of shaded region represent those velocities not reachable from O due to the limits of acceleration and deceleration. The inner rectangle, marked ABCD, represents the region of velocities where a possible solution can be found ***if and only if*** both robots alter their velocities. For $v_{R1} = 2.5$ corresponding to R1's velocity on the abscissa, R2 must possess a velocity, which lies either above or below the segments AB and CD of the rectangle when projected onto the

ordinate. Similarly for $v_{R2} = 2.5$ on the ordinate, robot R1 must possess a velocity either to the right or left of the segments BC and AD when projected onto the abscissa to avert collision. We denote the velocities that make R1 reach the velocities at D and C from O as v_{11} and v_{12} respectively, while the velocities that make R2 reach A and D from O by v_{21} and v_{22} respectively. With reference to figure 3 v_{11} and v_{12} correspond to velocities that enable R1 to reach C11 and C12 in the time R2 reaches C22 and C21 respectively without R2 changing its current aspiring velocity from v_{R2}.

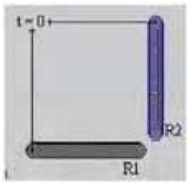


Fig. 5a. Two robots approach each other along orthogonal directions.

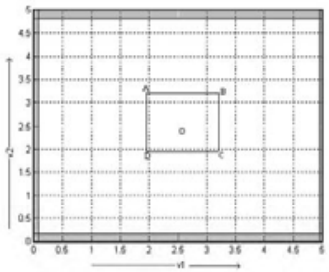


Fig. 5b. The possible range of velocities for robots R1 and R2 shown along the x and y axis. The inner rectangular area being cooperative region.

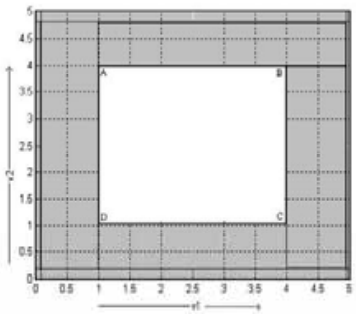


Fig. 5c. At t=25 the conflict area occupies the entire possible space of velocities.

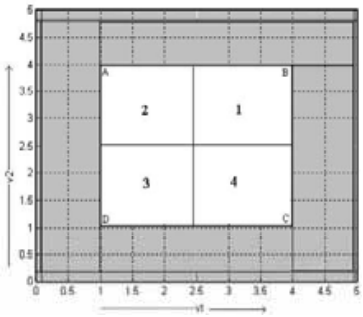


Fig. 5d. Search is limited to quadrants 2 and 4 where robot actions are complementary.

Figure 6a shows the snapshot at time $t = 0$ or $t_{c1} = 19$ (the time left for the robots to become tangent to one another for the first time) when robots approach each other head on. Figure 6b shows the collision region marked on θ axis for R1. All θ values in the interval [b,d] on the right and [a,c] on left are reachable and collision free. Values in the interval [d,M] and [m,c] are not accessible or unattainable due to the limits on angular acceleration of the robot, while those in [a,b] conflict with the impinging robot. Figure 6c shows the conflicting and inaccessible orientations overlap in intervals [a,c] and [d,b] for time $t_{c1} = 14$. Figure 6c shows the need for cooperation since the entire θ axis of R1 is either conflicting or

inaccessible or both. The values of θ to the left of O (corresponding to current heading of R1) on the θ axis of R1 are those obtained by turning R1 right in figure 6a & while those on the right of O on the θ axis are obtained by turning R1 to its left in figure 6a. While depicting the solution space in terms of θ for a robot the current heading is always 0 degrees for convenience.

Fig. 6a. Robots R1 and R2 approaching Head on.

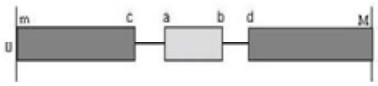


Fig. 6b. Collision and accessible regions on θ axis for robot R1 where [a, b] being the collision range.

Fig. 6c. Collision and accessible regions on θ axis. Dark area showing the overlapped collision and inaccessible regions.

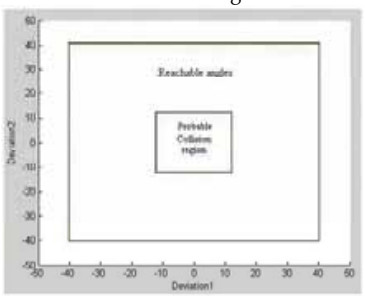


Fig. 6d. Joint orientation space for robots R1 and R2 in terms of θ_1 and θ_2. Outer rectangle representing accessible combination.

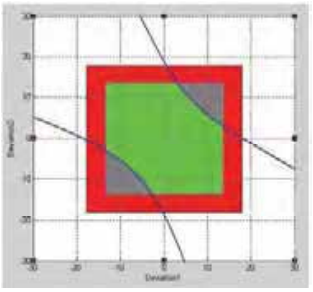


Fig. 6e. Joint orientation space for robots R1 and R2, where accessible region is inside the collision region where gray region representing cooperation zone.

Figures 6d and 6e depict the joint orientation solution space for robots R1 and R2 in terms of θ_1 (abscissa) and θ_2 (ordinate). Figure 6d corresponds to the situation for time $t = 0$ or $t_{c1} = 17$; the shaded parts of the rectangle comprises of regions inaccessible to both R1 and R2. R2 must reach a orientation on the ordinate that is either above or below segments AB and CD while R1 should reach a orientation that is either to the right of BC or left of AD. These orientations are denoted as θ_{11}, θ_{12} for R1 and θ_{21}, θ_{22} for R2 in a manner similar to velocity control discussed before. With reference to figure 4a θ_{11}, θ_{12} correspond to directions that enable R1 to reach the upper half plane of the segment $\overline{E1E2}$ or the lower half plane of $\overline{E3E4}$ before R2 reaches C21 without R2 changing its current aspiring orientation that is 0 degrees with respect to itself and 180 degrees with respect to a global reference frame, F shown in 6a.

4.1 Individual Phase for Velocity Control

A pair of robots *R1* and *R2*, which have a DC between them are said to be in individual phase of navigation if the conflict is resolved by either of the following two means:

(i) *R1* controls its velocity to v_{12} such that it is able to get past C12 before *R2* reaches C21 with its aspiring velocity as v_{R2} or *R1* controls its velocity to v_{11} such that it does not reach C11 before *R2* reaches C22 without changing its aspiring velocity from v_{R2}.

(ii) *R2* controls its velocity to v_{22} such that it is able to get past C22 before *R1* reaches C11 with its current aspiring velocity as v_{R1} or *R2* controls its velocity to v_{21} such that it does not reach C21 before *R1* reaches C12 without changing its aspiring velocity from v_{R1}.

In both cases it would suffice that only one of the two robots controls or modifies its aspiring velocity. This indeed is the crux of the individual phase where at-least one of the two robots is able to individually avoid the conflict without requiring the other to take action. Thus the range of velocities that permit individual resolution of conflict by *R1* is given by: $v \in [0, v_{11}] \cup [v_{12}, v_{1M}]$, where v_{1M} represents the maximum permissible velocity for *R1*, which is 5 units in figure 2b. They are given by: $v_{11} = vc_1 + a_{-m}t_{22} \pm \sqrt{(vc_1 + a_{-m}t_{22})^2 + (vc_1^2 + 2a_{-m}s)}$ Here s denotes the distance from *R1's* current location to C11, a_{-m} is the maximum possible deceleration and t_{22} is the time taken by R2 to reach C22 given its current state ψ_2. In the same vein the velocity that causes *R1* to be ahead of C12 when *R2* reaches C21 under maximum acceleration, a_m, is given by:

$v_{12} = vc_1 + a_m t_{21} \pm \sqrt{(vc_1 + a_m t_{21})^2 + (vc_1^2 + 2a_m s')}$, where, s' the distance from R1's current location to C12 can also be written as $s' = s + (r1 + r2)\cos ec(|\theta_1 - \theta_2|)$ and t_{21} is the time taken by R2 to reach C21 given its current state ψ_2. In a similar fashion velocities v_{21} and v_{22} are computed. Thus some of the possible sets of solutions from the set S_T are enumerated as:

$s_1 = \{v_{11}, v_{R2}\}$, $s_2 = \{v_{12}, v_{R2}\}$, $s_3 = \{v_{R1}, v_{21}\}$, $s_4 = \{v_{R1}, v_{22}\}$, $s_5 = \{v_{11}, v_{22}\}$, $s_6 = \{v_{21}, v_{12}\}$.

From the above list the first four solutions involve change in velocities of only one of the robots while the last two solutions involve change in velocities of both the robots. The last two solutions are examples of collaboration even in the individual phase as robots involve in a combined effort to avoid conflict even though they are not entailed to do so. The collaboration in the individual phase achieves the first capability mentioned in section 1 of avoiding conflicts in a manner that conflicting agents do not come too near while avoiding one and another. Amongst the last two solutions (s_5, s_6) that one is selected which involves minimal change from the current state of the respective robots. The last two solutions indicate that collaboration involves complementary decision making since one of the robots accelerates from its current velocity the other decelerates.

Henceforth for any robot the lower velocity is denoted as v_1 and the higher velocity by v_2 with the robot index dropped for notational simplicity. In other words the lower and upper velocities are denoted as v_1 and v_2 instead of v_{21} and v_{22} for R2 or instead of v_{I1}, v_{I2} for RI.

It is to be noted that the phrase that a robot change or modify its velocity is more precisely stated as the robot control or modify its *aspiring* velocity.

4.1.2 Individual Phase for Direction Control

Unlike velocity control a unique way of characterizing θ_{11}, θ_{12} is difficult depending on the angular separation between the robots and their directions of approach. However certain commonalities can be observed, namely (i) the robot to be avoided can be encapsulated within a planar envelope E (section 3.2), (ii) the robot that avoids has essentially two turning options either to turn left or right, (iii) the robot can reach a point in the plane that has no overlaps with E by reaching a heading, can in principle continue with the heading and avoid conflict forever with the same robot. Based on the above observations we formulate a conservative resolution criteria based on the angular separation between the two robots.

In purely velocity control a closed form solution to the values v_{11} & v_{12} was possible to ascertain, whereas in direction control a closed form expression for θ_{11}, θ_{12} is very difficult to obtain due to following reasons. Firstly in velocity control the robot had to reach a particular point for the limiting case. Whereas in direction control the robot is can reach any point on a line as long as its orientation is the same as that line in the limiting case. This leads to several velocity profile choices for the same target criteria. Secondly in the velocity control scheme it is possible to reach a particular linear velocity and maintain that as the aspiring velocity, however in direction control the eventual angular aspiring velocity needs to be zero for any avoidance maneuver. Hence it is easier to work in the space of directions than in space of angular velocities. For computing the solution space an exhaustive search mechanism is resorted by changing the time for which an acceleration command is applied for the same linear velocity. These are the solution spaces shown in the chapter under the assumption current linear velocity remains unchanged since those depicted are those for purely direction control. In case of the actual algorithm running real-time few sample points in the $\{v, \alpha\}$ space are computed before a conclusion regarding which phase of resolution is to be resorted to. The basis or the motivation for selecting the candidate points will be discussed elsewhere.

Figures 7a and 7b are similar to those of 4a and 4b. Figure 7a depicts the head on case while 7b portrays the case when angular separation between the robots lies in the interval [90,180). Both the cases have been discussed in detail in section 3.2 and early parts of this section when figures 6a – 6d were discussed. For the sake of completion we briefly mention them here. For 7a θ_{11}, θ_{12} are easily computed and correspond to directions that enable R1 to reach the lower half plane of the segment $\overline{E3E4}$ or the upper half plane of $\overline{E1E2}$ before R2 reaches P. For a given linear velocity of R1 θ_{11}, θ_{12} are symmetric on either sides of the current heading of R1 and this is expected as there are equal opportunities to avoid a conflict on both sides of the current heading. For figure 7b the conflict is best resolved if R1 reaches a point with a heading parallel to $\overline{E3E4}$ in the lower half plane of $\overline{E3E4}$ that does not contain R2. This can be achieved by either turning to its left or right. R1 can also aspire to reach a location in the upper half plane formed by $\overline{E1E2}$ that does not contain R2 before R2 reaches C21. This would once again involve R1 turning right. Hence θ_{12} corresponds to the value that is collision free by turning left whereas θ_{11} corresponds to the value that is collision free by turning right and reaching a point either on the upper half plane of $\overline{E1E2}$

before R2 reaches C21 or the lower half plane of the same $\overline{E1E2}$ without entering the envelope E during the time R2's center occupies the space from C21 to C22.

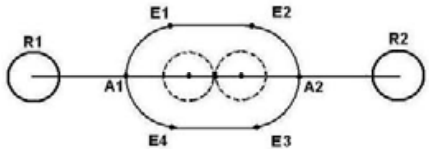


Fig. 7a. Robots R1 and R2 approaching head on.

Fig. 7b. Robots R1 and R2 approaching at an angle in range [90,180).

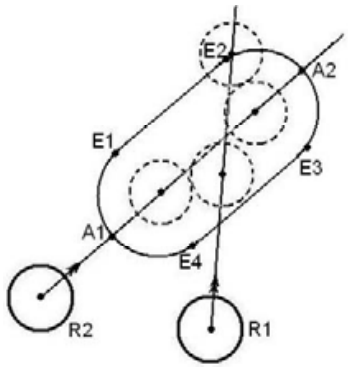


Fig. 7c. Robots R1 and R2 approaching at an angle less than 90 degrees.

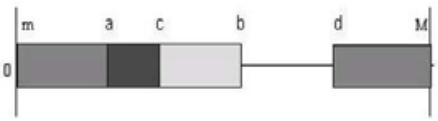


Fig. 7d. Collision and accessible regions on θ axis for robot R1 where [a,b] being the collision range.

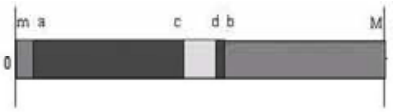


Fig. 7e. Collision and accessible regions on θ axis. Dark area showing the overlapped collision and inaccessible regions

Figure 7c depicts the case when the angular separation between the robots at the first instance of collision lies in (0,90]. Once again conflicts are resolved if the robot reaches a point in the half plane formed by $\overline{E3E4}$ along a orientation parallel to $\overline{E3E4}$ without entering the half-

plane that contains R2. Figures 7d and 7e have exactly the same connotations as figures 6b and 6c except that they are the plots for robots approaching each other not head on but as in figure 7c. The θ axis is depicted as shown before. Figure 7d corresponds to $t_{c1} = 31$. Note the entire reachable space lies on the right of current heading of R1. This indicates only turns to the left avoid conflicts or a value for θ_{11} does not exist even very early in resolution. This is only expected since a cursory glance of figure 7c indicates most of the turns of R2 to its right could only collide with R1. Figure 7e indicates the onset of cooperation with $t_{c1} = 11$ where all the reach orientations all are in conflict with R2.

4.3 Mutual (Cooperative) Phase for Velocity Control

The area enclosed within the rectangle ABCD of figure 5b is termed as conflict area for the pair of velocities $\{v_{R1}, v_{R2}\}$ for time $t = 0$ and denoted as $CA(v_{R1}, v_{R2}, t = 0)$. Let $V_{r1} = [v_{l1}, v_{h1}]$ represent the range of velocities for which there is a collision for robot R1 when R2 possesses a velocity v_{R2}. Similarly let $V_{r2} = [v_{l2}, v_{h2}]$ represent the range of velocities for which there is a collision for robot R2 when robot R1 possesses a velocity v_{R1}. We define the conflict area for the velocity pair $\{v_{R1}, v_{R2}\}$ for a given time t as $CA(v_{R1}, v_{R2}, t) = \{v_{R1}, v_{R2} \mid v_{R1} \in V_{r1}, v_{R2} \in V_{r2}\}$. The velocities v_{l1}, v_{h1} for R1 and v_{l2}, v_{h2} for R2 are arbitrarily close to their respective upper and lower control velocities v_1, v_2 that are used for resolving conflicts. In other words $|v_{l1} - v_1| < \varepsilon$ *for* R1, $|v_{l2} - v_1| < \varepsilon$ *for* R2 and similarly $|v_{h1} - v_2| < \varepsilon, |v_{h2} - v_2| < \varepsilon$ where ε is any arbitrarily low value. With progress in time if control actions to avoid conflicts were not resorted to the conflict area expands to occupy the entire space of possible velocities. This is shown in figure 5c where the conflict area fills up the entire velocity space. Any combination of velocities outside the rectangle ABCD now falls inside the shaded border strips, which are not accessible from O due to the limits imposed by acceleration and deceleration. Hence individual resolution of conflicts by any one of the robots is ruled out since the upper and lower velocities v_1 and v_2 for both R1 and R2 now lie inside the shaded area.

Since the upper and lower velocities are situated well inside the shaded area the velocity pairs corresponding to the vertices ABCD of the conflict area are unknown. Hence a cooperative search ensues for finding the pair of velocities that would resolve the conflict. Cooperation between robots averts an exhaustive search and restricts it two quadrants 2 and 4 (figure 5d) of the conflict area where robot actions are complementary and yield best results for conflict resolution. Since a search is nonetheless time intensive the rules (i) and (ii) mentioned below where robots resort to maximum acceleration and deceleration in a complementary fashion offer the boundary value solutions. A failure of the solutions at the bounds implies a failure anywhere inside and a pointer to resort to conflict propagation as the last resort.

A pair of robots R1 and R2 are said to be in mutual phase of navigation if and only if they are able to resolve the collision conflict between the two through either of the following rules:

(i) R1 is able to get past C12 under maximum acceleration before R2 can get to C21 under maximum deceleration.
(ii) R2 is able to get past C22 under maximum acceleration before R1 can get to C11 under maximum deceleration.

The difference between the above rules and those mentioned in section 4.1 is that in section 4.1 R1 finds a control velocity that avoids conflict with R2 under the premise that R2 would not alter its aspiring velocity. Similarly R2 finds a control velocity under the impression R1 is dumb. However in the cooperative phase R1 anticipates a modification in the aspiring velocity of R2 such as in rule 1 where R2 modifies its state (and hence its aspiring velocity) such that it reaches C12 under maximum deceleration. Under this anticipation of change in R2's control action R1 tries to attain the corresponding control velocity that would avoid conflict.

4.4 Mutual phase for direction control

As in velocity control figure 6e shows the situation when cooperation is inevitable since the entire accessible area (inner green rectangle) lies completely within the conflict area. The outer rectangle is the conflict area. The areas between the inner and outer rectangle are shown in red. The areas shown in gray within the rectangle are the solution pairs for which resolution is possible. Like in velocity control the solutions exist in opposing quadrants. The gray areas in first quadrant correspond to R1 and R2 turning left in figure 6a and 7a, while those in third quadrant correspond to R1 and R2 turning right. Once again if a solution does not exist at the top right and bottom left corners of the inner rectangle implies lack of solutions anywhere inside the inner rectangle.

Individual resolution through direction control fails because R1(R2) is unable to get out of E onto the half planes discussed earlier before R2(R1) reaches C21(C11). In such a situation the perpendicular distance from R1's (R2's) location to R2's (R1) trajectory is still less than r1+r2. Hence R2(R1) also changes its orientation to reach a location that would be r1+r2 away from each others trajectory by the instant it would have reached C21(C11) on the original trajectory had it not changed its direction. Hence a pair of robots can avoid conflicts mutually only if turning with maximum angular accelerations they can orient their trajectories by t_{c1} such that the perpendicular distance between a robot's position and the other robot's trajectory is at-least r1+r2. If the robots cannot reach such a location by t_{c1} under maximum angular acceleration applied till maximum angular velocities are attained then cooperative resolution would fail for all other values of α. Failure at ω_M obtained under maximum acceleration implies failure at the corners of the inner rectangle and hence a failure of the mutual phase to resolve conflicts.

4.5 Tertiary (Propagation) Phase for Velocity Control

Figure 8a shows the velocity axis for a robot RN. RN's current velocity is shown as O in the figure. The portions of the velocity axis shown shaded are those portions of the velocity forbidden from the current state of RN either because they are not reachable or they conflict with other robots. For example portions AB and FG on the axis are not reachable while portions BC, CD and EF conflict with robots R1, R2 and R3 respectively. At O, RN enters into a new conflict with a robot RM. Individual resolution of RN's conflict with RM results in conflict with R1 on the lower side and enters forbidden region on the upper side. Similarly RM's individual resolution leads to conflict with other robots or results in access of forbidden regions. When RN cooperates with RM to resolve the conflict it again results in

conflict with R2 on the lower side and R3 on the upper side. In such a scenario RN propagates cooperation request to R1, R2 and R3. The tree structure of figure 8b depicts this propagation. All nodes on the left of RN are requests arising due to lower aspiring velocities while nodes on the right of RN are requests that arise due to higher aspiring velocities. This convention would be followed for all robots involved in the propagation phase. Thus robot RN's resolution of its DC (Direct Conflict) with RM results in indirect conflict (IDC) with robots R1, R2 and R3 and hence RN is considered to be in IDC with R1, R2 and R3. When R1 or R2 try to collaborate in conflict resolution of RN by changing their aspiring velocities it can lead to further conflict with other robots to whom requests are transmitted by R1 or R2 for collaboration. Thus propagation can be recursive and results in a multiple tree like or forest data structure shown in figure 8c. A graph like propagation is avoided since a robot-node that has already propagated a request to another node below does not entertain any new requests.

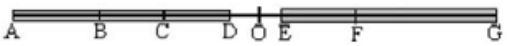


Fig. 8a. The velocity axis of the robot whose current velocity is at O. Shaded represents the inaccessible velocities due to conflicts.

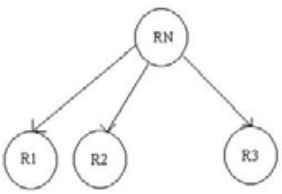


Fig. 8b. RN propagates requests to R1 and R2 on the left due to conflicts with lower velocities and on the right to R3 due to higher velocity.

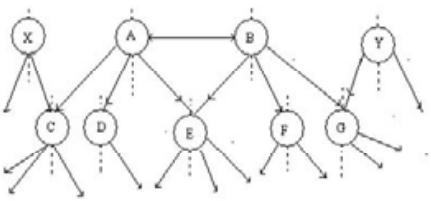


Fig. 8c. Propagation can result in a generalized multiple tree or forest structure whose links represent the flow of conflicts between robots.

Thus any robot has the following functionalities with regard to propagating requests which are taken up for discussion below

- Transmit requests
- Receive requests
- Reply to requests
- Receive replies

Transmitting Requests: A robot RT transmits a request to another robot RR a packet of that contains the following information:
Source: The robot that originally sourced the request.
T-robot: The robot that is currently transmitting the request, which is itself (RT).
R-robot: The robot to which the request is transmitted (RR).
V-aspire: The velocity which the transmitting robot RT, would aspire to have in order to avoid conflict which it has currently with some robot, R1, but which results in conflict with the robot to which the request is transmitted, RR.
t-collide: The minimum time to collision that RT currently has with R1
Mode: If the aspiring velocity V-aspire is higher than RT's current velocity then *mode* takes the tag *high* else it is assigned the tag *low*.
S-mode: If the S-mode has the tag *high* then it indicates that RT and RR would be the right descendants of the source robot else it indicates that they are left descendants.
RT transmits a request to RR only if RR is in a state of entertaining requests else the request is not transmitted to RR. A robot RR accepts a request to collaborate to resolve RT's *DC* with another robot only if RR itself is not involved in a DC.
Receiving Requests: A robot RR can receive single or multiple requests. A robot that receives requests from more than one robot to participate in its conflict such as *C* receives requests from *A* and *X* in figure 8c, prioritizes the requests in order of time to collision of *A* and *C* with the robots with which *A* and *C* are in conflict. The requests are processed in the order of their priorities. If a request could be resolved a success reply is propagated back to the robot that transmitted the request. A success reply indicates that RR intends to modify its aspiring velocity with respect to that request. Hence it cannot modify its velocity to the remaining requests it has received and hence propagates a failure back to the remaining robots that had requested RR. If a request is not solved it is either propagated to another robot or a failure transmitted back to the robot that transmitted. Unless all the requests had resulted in a failure being transmitted RR does not entertain any new request for that sample. In other words if RR has managed to solve at-least one request or passed at-least one to another robot it does not accept any new request for that sample. A sample is one complete execution of the entire reactive loop or module across all robots.
Replying requests: A request is replied back as success or failure to the robot that transmitted in the manner described in the previous paragraph.
Receiving replies: A robot RT that had transmitted requests to other robots receives a success or failure reply from the robots to which it had transmitted. If a success reply is received RT sees whether the reply is from its left or right child. From the side on which the success was received a check is made if all other robots that had received the request from RT with respect to that particular aspiring velocity of RT have also replied a success. If all other children with respect to that v-aspire from that side (left or right accordingly) have propagated a success then RT propagates a success to the parent whose request to RT has now succeeded. It removes links with all its remaining children since it has already achieved a success on one of its v-aspire, which would become its new aspiring velocity. To its remaining parents it propagates a failure reply. On the other hand if RT receives a failure reply from its left or right child, it propagates a failure reply to RT's parent responsible for that request. Simultaneously it removes all other children on the particular side from which the failure reply was received with respect to that aspiring velocity.

This process of replying requests and receiving replies is recurred back till the original source or the root.

4.6 Tertiary Phase for Direction Control

The tertiary phase for direction control is a replica of the velocity control scheme but for the following minor changes

Transmitting Requests:

i. While transmitting requests θ-aspire the aspiring orientation of RT to avoid a conflict with R1 but which results in a conflict with RR is transmitted instead v-aspire. This is indeed obvious
ii. Mode: If the aspiring orientation of RT requires RT turning left the mode tag takes high else it is low.

Receiving Requests:

A robot that receives multiple requests tries to modify its orientation according to following heuristics:

i. Prioritize the requests such that the request from the robot that has the shortest collision time receives highest priority. Requests are serviced in the order of priority. Once a request is resolved other requests are not attempted to be resolved, they are either propagated to other robots or a failure is propagated back to the parent which had propagated the request.
ii. A robot tries to resolve as many requests as possible by appropriately finding a new aspiring orientation that overcomes all those conflicts
iii. A robot tries to see the impact value of a request. A request's impact value varies inversely with number of robots that need to modify their aspiring states for a conflict of the robot that transmitted the request. Hence a robot tries to resolve those requests that do not simultaneously require other robots also to modify their states since such requests have the highest impact.

The results reported in the subsequent section are those that incorporate the first heuristic.

4.7 Cost function

Often conflicts are resolved in multiple ways leading to multiple solutions. Let the set of solutions identified be $S_I = \{s_1, \ldots, s_N\}$ where each $s_i \in S_I$ is either a velocity tuple $\{v_{i1}^a, v_{i2}^a, \ldots, v_{ini}^a\}$ or an orientation tuple $\{\theta_{i1}^a, \theta_{i2}^a, \ldots, \theta_{ini}^a\}$ depending on the control methodology invoked. Here ni is the number of robots that were involved in the resolution for that tuple. The superscript 'a' indicates the aspiring nature of the element in a tuple. The tuple selected could be based on the following criteria for velocity control:

i. $s_j = \min(Dev(s_i))$ where $Dev(s_i) = \sum_{j=1}^{ni} \left| v_{ij}^a - v_{ij} \right|$, where v_{ij} is the current velocity of the robot. Here we look for the solution where the sum of changes in velocities over all robots is a minimum.

ii. $s_j = \min[(Dev(s_i))]$, where $[Dev(s_i)]$ indicates the number of changes in velocity entailed in a solution set s_i. Hence that solution is preferred where the number of robots changing their state is a minimum.

iii. $s_j = \min[(Dev(s_i))]$, where $Dev(s_i) = \frac{1}{n_s}\sum_{j=1}^{ni}\left|v_{ij}^{a} - v_{ij}\right|$, where n_s is the number of robots that had changed their state. This is in contrast to the cost function in ii since it promotes the case where small changes in states by many robots over large changes due to fewer ones. This cost function is more intuitive with a participatory cooperative mechanism and is what is used in the results presented in section 5.

The criteria for choosing a solution tuple for orientation control is along same lines except that $Dev(s_i)$ is computed as the maximum deviation of the trajectory of a robot computed by dropping a perpendicular onto the original trajectory from the location reached by the robot in the new trajectory at t_{c1}.

4.8 Local Planning as an Alternative

The attractiveness of a decentralized collision scheme decreases as the number of transmissions and replies between robots increase, consuming a lot of bandwidth eventually leading to a breakdown. In such a case the role of a local planner running onboard the robot or within their vicinity needs to be explored. All robots involved in a conflict either directly or indirectly can be brought within the ambit of a planner and local plans guaranteeing collision freeness for the next T instants can be computed and disseminated. The tradeoff however is as the number of robots increase local planners need to resort to some kind of search techniques such as hill climbing to come up with collision free plans and the real-time nature of such methods is under scrutiny. One of the future scopes of the current effort is to evaluate situations where the role of such a local planner enhances the performance of a multi-agent system.

Fig. 9a. Robots Moving in Orthogonal angle

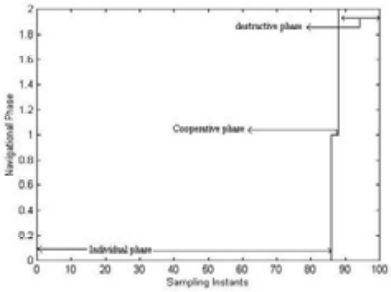


Fig. 9b. Various phases of navigation versus sampling instants for orthogonal separation.

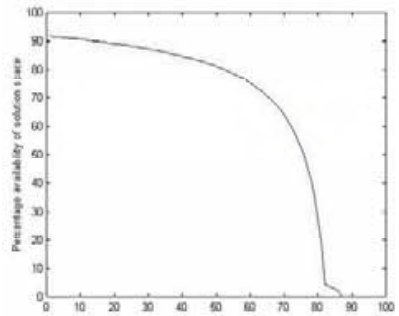


Fig. 9c. Percentage availability of solution space versus sampling instants

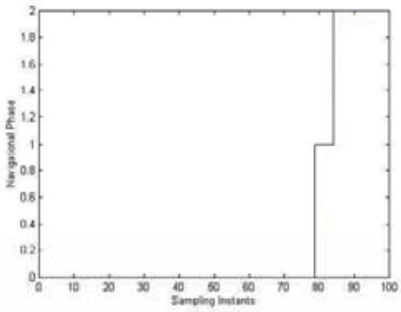


Fig. 9d. Phases of navigation for angular separation of 45^0.

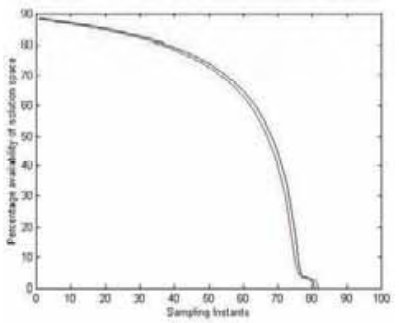


Fig. 9e. Percentage availability of the solution space does not overlap precisely in this case for the two robots and hence the demarcation between the two plots.

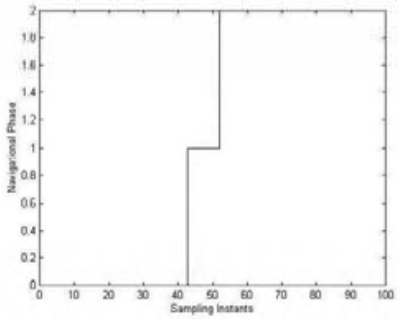


Fig. 9f. The cooperative phase becomes prominent for an angular separation of 15^0.

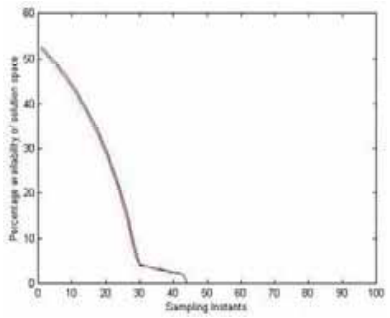


Fig. 9g. Percentage availability of solution space versus sampling instants for an angular separation of 15^0.

5 Simulation Results

This section is organized as follows. Initially the existence of the cooperative phase in a multi-robot navigation system is portrayed in section 5.1 and the effects of parametric variations on the time span of the cooperative phase is presented. In section 5.2 the inevitability of cooperative phase is discussed. Section 5.3 presents results of multi-bodied system and illustrates the effects of scaling up of the number of robots on the requirement to cooperate and propagate

5.1 Existential Evidence

Velocity Control

The existence of the cooperative phase in navigation and its time span of existence vis-à-vis the angular separation between robot heading angles, $(|\theta_1 - \theta_2|)$, for the two bodied case is first presented. Robots are made to approach each other at various angular separations and the percentage of solution space available for choosing control velocities that could avoid collision is computed. The percentage availability of solution space is computed as $\frac{L_{US}}{L_T} \cdot 100$, where L_{US} is the length of the line that is not shaded on the velocity axis and L_T refers to the total length of the velocity axis.

However the robots do not chose these velocities but continue to proceed until the solution space dries up completely indicating the onset of cooperative phase. If the robots continue to navigate without entering into a cooperative scheme for collision avoidance, a stage arises where even cooperation would not prevent collision. This final phase is termed as the destructive phase, where the robots inevitably have to collide into each other.

Figure 9a depicts a two-bodied case where the robots approach each other with an angular separation of 90 degrees. Figure 9b illustrates a graph that takes discrete values on the y-axis versus sampling instants on the x-axis. Sampling instants denote the onset of a new reactive loop of the algorithm. The delays are appropriately introduced in the algorithm to make the time-length of every reactive cycle and hence every sample constant. For all the simulations portrayed in this section (6.1) the maximum velocity of either of the robots is 5 pixels per sample and the maximum acceleration for both the robots is 2 units. The discrete values on the ordinate (y-axis or vertical axis) of figure 9b indicate the various phases of robot navigation. An ordinate value of 0 denotes the *individual phase* where the robot can avoid collision individually without entering into cooperation. An ordinate value 1 signifies the *cooperative phase* of navigation where the solution space has dried up and the robots needs to cooperate for averting collision. Finally value 2 on the ordinate implies the *destructive phase* where the robots inevitably need to collide or have already collided.

In figure 9b the individual phase spans for 86 sampling instants from the start of navigation while the cooperative phase extends for only two instants after which the robots enter their destructive phase. Figure 9c depicts the percentage availability of solution space for choosing control velocities corresponding to the various navigational states of the robot in figure 9b. It is evident from figure 9c that the range of options available in the solution space decreases with time and hits zero in the 86th sample where correspondingly in figure 9b the robot enters the cooperative phase of navigation on that instant. Equivalently the conflict area expands to occupy the entire space of possible velocities as depicted in 5c. Figures 9d and 9e depict the phases of navigation and the availability of solution space when robot pair

approaches one another with an angular separation of 45 degrees, while figures 9f and 9g depict the same for a separation of 15 degrees. These figures indicate that the cooperative phase onsets earlier as the angular separation decreases and correspondingly the range of options on the solution space reduce to zero faster. The span of the cooperative phase also increases with decrease in angular separation and in figure 9f it becomes rather prominent. It is also worthwhile to note in figures 9e and 9g the percentage availability of the solution space does not overlap precisely for the robot pair over sampling instants. Hence the appearance of two distinct plots corresponding to the two robots. In figure 9e the percentage availability of solution space hits zero for one of the robots ahead of the other. However, the system itself enters a cooperative phase only when the individual solution space exhausts for both the robots. The analysis indicates that the need to resort to cooperative phase for conflict resolution would increase when robots approach one another with reduced angles of separation. This is expected since the distance between C11 and C12 (C21 and C22) increases as the angular separation between the robots decreases. With increasing distances the conditions (i) and (ii) for individual resolution of conflicts mentioned in section 4.1 becomes more difficult to be met. Equivalently the percentage of individual solution space becomes less for the same reaction time for considering conflicts.

Orientation Control

Figures 10a, 10b and 10c depict the percentage availability of solution space during individual and mutual resolution of conflicts for cases of robots approaching each other head on (that we denote as 180 degrees), orthogonal to each other and when they approach one another with a separation of 30 degrees. In contrast to velocity control the individual phase lasts for the least for robots approaching at 90 degrees to one another among the three cases and the amount of leverage gained by resorting to orientation control is also the least for this case. This conclusion is from the graph 10b for orthogonally approaching robots that shows the percentage of solution space available when a robot attempts to resolve conflict individually is the least. The percentage availability of the individual solution space (the gray regions within the inner green rectangle in figure 6e) is highest for the 30degree case. Whereas the percentage availability of the mutual solution space when individual resolution ends is highest for the head on case. These somewhat counterintuitive results can be explained as follows. Consider R1 approaching R2 head on with R1 trying to avoid the conflict. Turns into both half planes formed by the line along the current heading of R1 passing through R1's center can yield solutions.

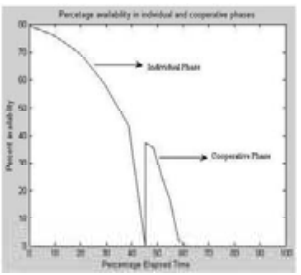


Fig. 10a. Percentage availability in both individual and cooperation phases versus percentage time Elapsed, for head on case.

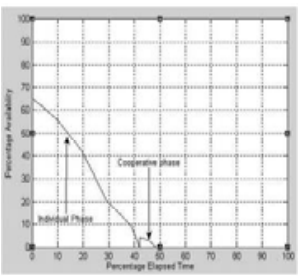


Fig. 10b. Percentage availability in both individual and cooperation phases versus percentage time elapses, for 90 degree case.

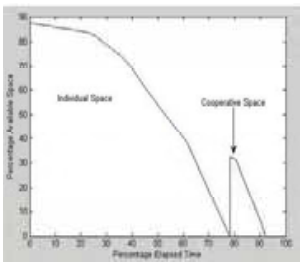


Fig. 10c. Percentage availability in both individual and cooperation phases versus percentage time elapses, for 30^0.

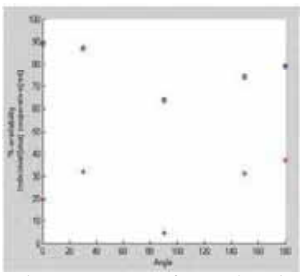


Fig. 10d. percentage of initial individual and cooperative solution space available versus various angles.

However since the relative velocity between the two robots is also the maximum the solution space decreases rapidly as many of the turns into both quadrants do not resolve conflicts with passage of time. This explains the fastest decrease in individual solution space for the head on case and the slowest decrease for the 30 degrees case (since the relative velocity is the least for this case). A cursory look at the slopes of the curves shows that they are most steep for head on and least steep for 30 degree case. For reasons why the highest percentage of solution space is available for 30 degree case initially the following explanation is given. Albeit the fact that turns into one half plane are most likely to yield collision (the half plane that contains R2) almost all the turns into other half plane (that does not contain R2) are collision free since those turns are not steep and easily attainable. For the head on case as explained in previous sections reaching to either the half plane above $\overline{E1E2}$ or the one below $\overline{E3E4}$ are of same steepness. So the gains made in the solution space on a turn into one of the half planes over the 30 degree case is compensated for the turns into other half plane for which the 30 degree case is entirely conflict free. The net result being that the percentage of solution space initially available for individual resolution is highest for the head-on case. The turns to avoid conflict on either half planes are steep for the orthogonal case and hence many of the turns are unable to avoid the conflict leading to the least amount of solution space available initially for the orthogonal case.

The higher percentage of mutual or cooperative solution space for the head-on over 30 degree case is arguably due to the fact that the gray solution areas exist both in the 1st and 3rd quadrant in for the head on case (figure 6e). Whereas for 30 degree angular separation most of the solution space would be confined to one of the quadrants.

Figure 10d shows a plot of percentage of initial individual solution space available versus angular separation. The plot shows that the highest percentage is available for robots at zero degrees (one behind the other) and least for orthogonal case. These discussions suggest that velocity control is apt when angular separation is close to 90 and orientation control is apt when angular separation closes in to zero or 180 degrees.

5.2 Inevitability of cooperation

While the existential evidence of the mutual phase or cooperative phase is established how essential the need for it is.

Requirement for cooperation in two-bodied system

For the two-bodied system discussed in last section cooperation could have been avoided if robots took preemptive actions before the onset of the cooperative phase. Table 1 illustrates under what set of parameters did an invocation of a cooperative scheme for collision avoidance became unavoidable. The table suggests for the case of 90 degrees separation in robot heading directions cooperation becomes inevitable only when the robot's reaction time is considerably reduced to 5seconds and when it possesses awful dynamic capabilities such as when it cannot accelerate faster or decelerate slower than $0.15\, pixels/sample^2$. However when the angular separation was 15 degrees even default parameters entailed the cooperative phase. *Hence the requirement of a cooperative scheme in real-time navigation is not artificial even for a simple two-bodied system.*

Angular Separation (degrees)	Reaction Time (seconds)	Maximum Acceleration, Deceleration pixel/s2	Maximum velocity pixel/s
90	5	0.15,-0.15	5
45	5	0.45,-0.45	3
15	12	2,-2	1

Table 1. The reaction time for various Angles, in case of velocity control.

Angular Separation (degrees)	Reaction Time (seconds)	Maximum Angular Acceleration rad/s2	Velocity pixel/s
180	17	0.0045	3
90	16	0.0045	3
30	11	0.0045	3

Table 2: The reaction time for various Angles, in case of orientation control.

Table 2 depicts the parameters for which mutual cooperation is inevitable in case of orientation control. All things being same with respect to table 1 the third column is the angular acceleration in radians per second square. The fourth column shows the constant linear velocity with which the robot moved in those cases. The maximum angular velocity was also same for all the test runs.

5.3 Simulation with Multiple Robots

Velocity Control

For all the simulations portrayed in this section the maximum velocity of either of the robots is 5 pixels per sample and the maximum acceleration for both the robots is 2 $pixels/sample^2$. The reaction time t_r is fixed at 12 samples. All robots are capable of communicating to one another within a range of 100 pixels.

Figure 11a shows an instant during the navigation of a system of five robots where robots 1 and 3 are unable to resolve their conflicts between them individually as well as cooperatively as cooperative solutions lead to indirect conflict with robot 4. Hence 1 and 3 propagate a request to resolve their conflict to 4 thereby embarking on the propagation phase as the last attempt to resolve their conflicts. Robot 4 accepts requests from 1 and 3 and is able to solve the request of 1 by modifying its current velocity such that 1 and 3 are able to avoid their mutual direct conflicts. This scenario is depicted in figure 11b where 4 moves

faster in such a way 1 and 3 are able to avoid their mutual direct conflict. Figure 11c shows the space-time evolution of trajectories for the robots of figures 11a and 11b. The x and y axes indicate the regions in the x-y plane occupied by a robot every time it samples the environment. Robot samples of the environment in time are shown along the z axis as sampling instants. The five solid lines of the figure correspond to the trajectories of the five robots. The figure shows that the robot trajectories do not overlap in space-time confirming that all collision conflicts were resolved by the algorithm.

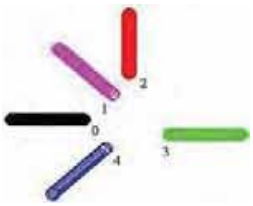


Fig. 11a. Snapshot of system of five robots.

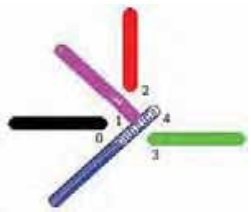


Fig. 11b. Robots 1 and 3 propagate requests to resolve their conflicts to robot 4, which accepts the request and moves faster such that 1 and 3 are able to avoid their mutual direct conflict.

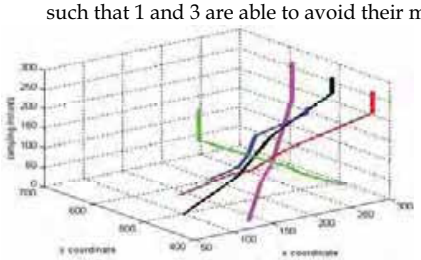


Fig. 11c. Space-time evolution of trajectories for the five robot system.

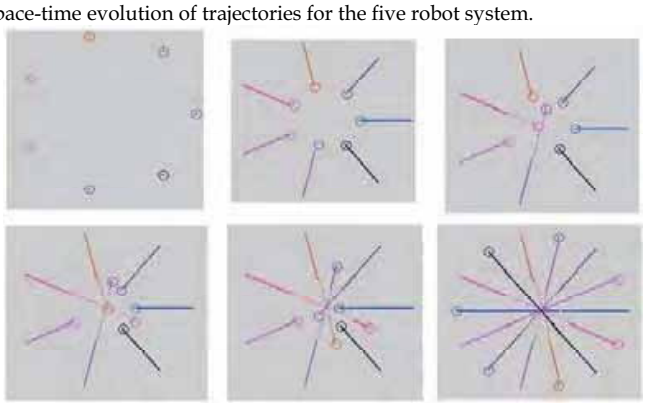


Fig. 12. Sequence of snapshots arranged from left to right with the second row following the depicting navigation of a system of eight robots. The third and fourth snapshots depict instances where propagation of conflicts was resorted for conflict resolution.

Figure 12 shows a sequence of snapshots during the navigation of a system of 8 robots. The sequence is ordered left to right with the second row sequence following the first row. The traces of the robot are shown by thin lines rather than by the size of the robot. The rightmost snapshot in the first row and the leftmost snap shot on the second row are instances when propagation phase was effected for conflict resolution. The first and the last snapshots represent the initial and final configuration of the robots.

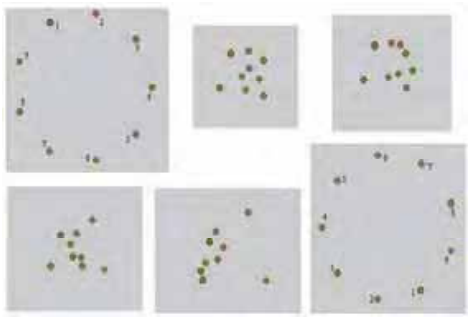


Fig. 13. Sequence of snapshots during navigation of a system of nine robots.

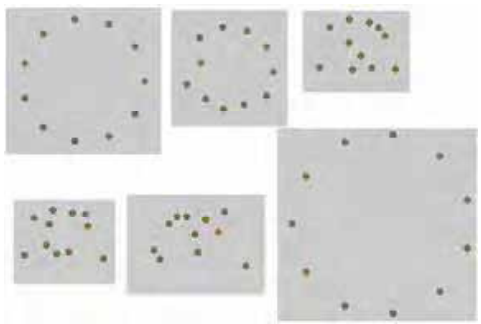


Fig. 14. Sequence involving 11 robots. The second of the snapshots indicates the instance when robots begin to react to each other's presence.

Figure 13 shows yet another sequence of six snapshots of a system of nine robots arranged in the same order as in figure 12. The first and the last snapshots are the initial and final configurations of the nine robots. The robots are labeled 1 to 9 in the first and last figures. The traces of the robots are not shown for clarity. The initial and final configuration resembles a clock like structure. In other words a robot placed at position 3 in a clock needs to get to a goal location which is near 9 and a robot placed near nine initially has its goal configuration near 3. These examples depict simulations with increasing difficulty as the number of robots increase and all of them converge towards a common junction. Hence the trajectory of every robot intersects with every other robot and hence the number of collision conflicts of the total system is high. It is also worth emphasizing that robots consider collision conflicts only within a reaction time of 12 samples by which time the robots have converged sufficiently close to one another.

The sequence of snapshots shown in figure 14 highlight a more difficult example involving 11 robots at similar initial and final configurations as in figure 12. When the number of robots were increased beyond 11 some of the conflicts could not be resolved and hence collisions were encountered between the robots. The second of these snapshots represent the instant when robots first begin to react to each other's presence by embarking a strategy for resolving conflicts.

Fig. 15a. Initially red and blue robots detect collision and avoiding collision cooperatively.

Fig. 15b. Two more robots arrived in the scenario.

Fig. 15c. In order to minimize the optimization function robots black and cyan change their directions and avoid collision.

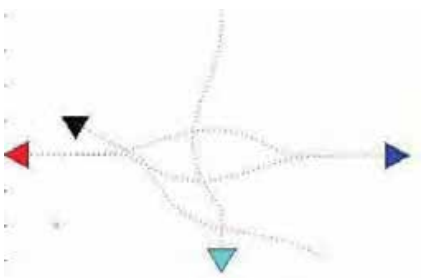


Fig. 15d. All Robots after avoiding collision go through their actual path.

Direction Control

Figures 15a-15d shows a sequence of snapshots of four robots red, blue, black and cyan avoiding collisions by direction control. Initially (figure 12a) red and blue are within communication distance of one another and begin to avoid conflict in the cooperative mode to minimize deviation suffered by one robot alone. After a while both cyan and black enter the communication zone of blue and red with cyan in conflict with blue and black with red. Red has already deviated to its left in avoiding blue and same is the case with blue. To avoid black and cyan the best recourse for red and blue is to turn right on their current heading that brings them into conflict once again. Hence the only solution left is for both black and cyan to modify their trajectories which are shown in the remaining snapshots.

6. Summarizing Comments

A method by which the velocity and orientation axis of the robot can be dissected into various portions and these portions labeled as conflict free or conflicting with a particular set of robots, unreachable and reachable is presented. The conflict free intervals along the axis that are also represent the solution space for the robot. When the entire reachable space is conflicting with one or more robots it points to the need for cooperation between robots. For a pair of robots that are in conflict with one another and either of them unable to resolve the conflict individually plot of the joint solution space demarcates area where cooperative change of velocities or directions can result in collision freeness. For a pair of robots the entire solution space need not be searched, if a solution is not possible at the corners of the rectangular portions demarcated there is no possibility to resolve the conflict by mutual cooperation. Often a velocity or direction control strategy that solves a conflict with one robot results in conflict with others there by bringing in more robots into the resolution scheme. In such situations the space to be searched for finding a solution increases exponentially. However a method by which requests are passed between robots to resolve the conflicts drastically reduces the space to be searched. We call this as a three-tiered strategy of individual, mutual and tertiary levels. Simulation results confirm the efficacy of the method. Existential and inevitable nature of cooperation is also presented and analyzed in detail

7. References

Alami. R; Fleury. S; Herbb. M; Ingrand. F and Robert F (1998). "Multi Robot Cooperation in the Martha Project", *IEEE Robotics and Automation Magazine,* 5(1) Fujimura K, (1991). *Motion Planning in Dynamic Environments*, Computer Science Workbench, Springer-Verlag

J. Barraquand and J. C. Latombe. , (1990) A monte-carlo algorithm for path planning with many degrees of freedom. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA).*

Bennewitz, M.; Burgard W. and Thrun S., (2002). Finding and Optimizing Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots. *Robotics and Autonomous Systems*, 41 (2), 89-99

Choset H.,(2001). "Coverage for robotics - a survey of recent results". *Annals of Mathematics and Artificial Intelligence*, 31:113-126.

Fujimori A; Teramoto M.; , Nikiforuk P.N. and Gupta M M (2000), Cooperative Collision Avoidance between Multiple Mobile Robots, *Journal of Robotic Systems 17(7),* 347-363

Fujimura . K (1991), Motion Planning in Dynamic Environments, *Computer Science Workbench*, Springer-Verlag

Fox D, Burgard W, and Thrun S.(1997) The Dynamic Window Approach to Collision Avoidance.
IEEE Robotics & Automation Magazine, 4(1).

Genevose V; Magni R and L. Odetti (1992). Self-organizing Behavior and Swarm Intelligence in a Pack of Mobile Miniature Robots in Search of Pollutants, *Proc. 1992, IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems,* Raleigh, NC, 1575-1582

F. Gravot and R. Alami (2001). An extension of the plan-merging paradigm for multi-robot coordination. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2001.

Latombe J C, (1991). *Robot Motion Planning*, Kluwer Academic Publishers, 1991

Lumelsky V.J., & Harinarayan K.R. (1998), "Decentralized Motion Planning for Multiple Mobile Robots: The Cocktail Party Model", *Autonomous Robots*, 4:121-135.

Madhava Krishna K and Kalra P.K (2002), "Detection Tracking and Avoidance of Multiple Dynamic Objects", *Journal of Intelligent and Robotic Systems*, 33(3), 371-408, Kluwer Academic

Parker .L.E. (1998). ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation, *IEEE Transactions on Robotics and Automation*, 14 (2).

S. Leroy, J. P. Laumond, and T. Simeon (1999). Multiple path coordination for mobile robots: A geometric algorithm. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Srivastava P, Satish S and Mitra P (1998).: A distributed fuzzy logic based *n*-body collision avoidance system, *Proc. of the 4th Internaional. Symposium on Intelligent Robotic Systems*, 166-172, Bangalore.

Sveska P. and Overmars M , (1995). Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*.

30

Robot Collaboration for Simultaneous Map Building and Localization

M. Oussalah and D. Wilson
University of Birmingham, Electronics, Electrical and Computer Engineering
Edgbaston, Birmingham B15 2TT, UK

1. Introduction

Collaboration has been acknowledged as an excellent tool to compensate for limitations and shortcoming of individuals in order to achieve complex tasks. Yet, robotics collaboration has been recognized as an independent entity on its own within robotics community as suggested by the emerging literature and the growing applications like RoboCup, FIRA competitions (Kitaneo, 1997), autonomous vehicles for space/submarine exploration (Todd and Pomerleau, 1996). This promises a leading future for this field in a medium term. However, the development of effective collaboration schemes is subject to several challenges. This concerns aspects related to robot localization (absolute and/or relative localization), environment map building, sensor modelling and fusion, game-theoretic scenarios, collaboration/cooperation modes, user's interface and control modes, among others, see, for instance, (Mataric, 1998). This chapter aims to contribute at least to the first two aspects of the aforementioned challenges where the issue of dynamic localization and map building using two miniature Khepera® robots is tackled. An extended-Kalman filter based approach is developed and implemented in order to model the state of the robot and various observations as well as to determine and update the positioning estimates of both robots together with the identified landmarks in the environment. A virtual representation of the map and robots is also put forward using OpenGL for 3D representation. While the developed interface uses enhanced help capabilities in case of unsafe or non-tolerated manipulations by the user.

The issue of mobile localization and map building has been a challenging issue that faced the robotics community since the eighties due the debatable issues related to the state and observation modelling, map initialization and building, and convergence of the estimation process, among others. This led to the development of several techniques to overcome the above challenges. Since the pioneering work of Smith and Cheesman (1986), a bridge from geometrical features and stochastic models has been established, which led to a variety of algorithms, mainly using Kalman filter (Geb, 1986) or its variants, whose feasibility and satisfactory performances have been demonstrated both from theoretical and practical perspectives through the convergence properties of the algorithms and the successful applications.

The concept of robot localization and map building is often referred to as SLAM (Simultaneous Localization And Mapping), in which both the environment map represented as a set of landmarks and the robot states are estimated simultaneously by augmenting the state vector to

include both robot's state and landmark states (Leonard and Durrant-White, 1991a; Dissanayake et al., 2000, 2001; Thrun et al., 1998; Bailey and Durrant-White, 2006). This trivially increases the autonomy of the robot(s) as it permits consistent robot navigation without requiring a priori map. Besides, the study and experiments carried out in the above citations, among others, demonstrated the feasibility of SLAM both from theoretical and practical viewpoints despite the challenging issues related to complexity, map building and data association, etc. In contrast to the stochastic models which govern the construction of the majority of the proposed SLAM models, one shall mention the increasing literature in soft-computing based approaches like fuzzy/possibility-based approach (Oussalah et al., 2003), neural network (Nonato et al., 2006), genetic algorithms (Higuchi, 1996), inference systems (Begum et al., 2005). Reference (Murphy, 2000) provides a good overview of such methodologies.

The vast majority of SLAM implementations utilise expensive sensor equipment that has good range detection and high accuracy, typically laser range finder, ultrasonic sensors and/or vision systems. Although the use of ultrasonic sensors causes specular reflection while the ultrasonic beam deteriorates the measurement accuracy. On the other hand, the vision system induces high computational complexity, which opens new areas for research and investigations in order to achieve high balance in terms of cost-effectiveness ratio. Besides, almost all the implementations so far restricted to a single robotic system (Bailey and Durrant-White, 2006; Durrant-White and Bailey, 2006). Consequently the use of a group of Khepera robots together with the limited sensor capabilities and restricted range of infrared sensors makes the SLAM problematic even more challenging. For this purpose, similarly to (Dissanayake et al., 2000, 2001) a stochastic SLAM based approach was developed to account for the multi-robotic systems. The underlying SLAM model depends on the robot collaboration mode. For instance, in case where the vision robot restricts its movement to rotation to identify possible objects, the state vector includes both the state of both robots as well as that of landmarks. While in case where both robots achieve non-negligible movement, the state vector includes state of each robot together with each environment map constituted by its set of landmarks. The latter is made of Cartesian points, which are transformed into feature landmarks - mainly segment lines and corners-. Performances of the proposal will be illustrated through experimental setup. Section 2 of this chapter describes the overall system setup providing an overview of the system, concept of robots' collaboration, mapping and user's interface. Section 3 recalls the basis of stochastic SLAM model and develops the proposed filter estimation algorithm. Section 4 examines the experimental performances of the developed algorithm. Next overall conclusion is provided.

2. System Setup

2.1 System Overview

Two Khepera robots were used in our experiment. Both robots are equipped with encoders and infrared sensors. Besides, one of the robots is equipped with a linear vision turret, which allows detection of far away objects. One refers Vision Robot (VR) to the robot equipped with vision turret and Blind Robot (BR) to the other one. The two robots BR and VR are controlled via the two serial connections COM1 and COM2, respectively, of a regular windows-based PC platform.

Each of the robots only has 512k of memory on-board. This small amount of memory and the limited processing powers of the robots entail that almost all of the map construction and robot control are accomplished on the PC platform while a

behavioural safety oriented navigation scheme is embedded in robot platform executed by local micro-controller in order to prevent robots to crash with obstacles or objects using the infrared sensors. The user interacts with the system through the control program interface. Figure 1 shows a typical system setup using a PC with two serial extension cables and a custom made testing environment.

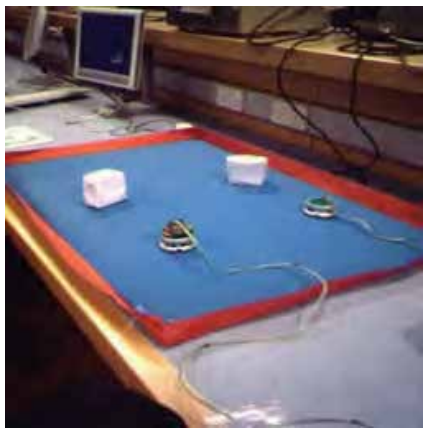


Fig. 1. Instance system setup.

2.2 Robot Collaboration: Concept

Building on the ability of the two robots to individually add objects to the map based on their tracked location, the robots are able to collaborate their efforts to achieve the shared goal of mapping the environment. This ability also helps make use of the vision turret, which one of the robots is equipped with. For example the VR can send a BR to a typical location viewed through its vision turret in order to explore the neighbourhood of the underlying object using the infrared sensor the BR is equipped with. This is especially useful when, for example, the two robots are on opposite sides of an object. So, by calling the other robot to the location of the detected object, if the other robot comes from the other side of the object, it will encounter the other edge of the object perimeter in its travels, and add that to the map accordingly. On the other hand as the range of the infrared sensors is very limited (only few centimeters), the use of vision turret whose range may go up to 1 meter allows us to compensate for such limitation. The diagram in Figure 2 below illustrates this point where the arrows show direction of robot travel.

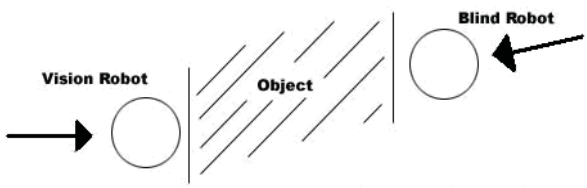


Fig. 2. Robots collaborating to map opposing edges of same object.

Especially, when the action to search for an object is sent to the VR, the latter starts turning via fixed angular increments until the vision turret detects an object, then either the robot moves toward it or sends a command to the BR to go to object location according to the determined azimut and explore it via its infra-red sensors. The process of detecting an object with the turret is based on: i) the number of pixels (out of 64 maximum) that detected the object; ii) calculus of the target pixel, which defined the edge of the object; iii) turning the VR until the turret detects an object at the target pixel; vi) storing the orientation angle. The above process effectively ensures that of all the pixels that do not detect an object, half are on the left and half are on the right, leaving the object-detected pixels in the centre of the field of view.

2.3 Mapping

Due to low resolution of sensors equipping the robots, the SLAM approach of representing the object as virtual points in the x-y coordinates sounds appealing. However, these virtual objects, if detected by vision turret, will get further explored using infrared sensors in which the virtual points are, under some geometrical constraint, linearly fitted together, which will form the environment map. These points will act as landmarks for the robots, which will then be added to the vector state model containing the x-y and pose of the robot. As special interest, one notices the fact that the second robot will not be used as an extra object in the environment but rather will use the information of 2nd robot's positioning to update the current robot location. This builds a bridge towards the issue of relative positioning which has focused much of interest in recent years. From a geometrical perspective, as far as range sensors were used, the detected object is modelled as a rectangular object centred around the virtual point identified by the sensor and whose direction is perpendicular to sensor beam as illustrated in Figure 3.

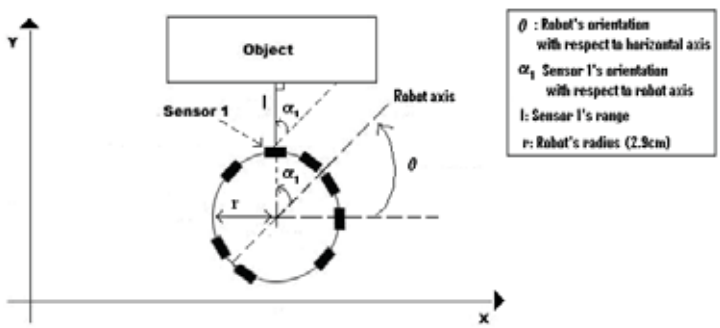


Fig. 3. Robot configuration and object modelling.

The systems mapping ability is combined with the robot localisation, which provides further evidence on whether the object needs to be added to the map or is already an observed landmark or just a false alarm that needs to be discarded. Besides, the object once added to the map will be used again in the observation model to correct the robot positioning as will be detailed later on. On the other hand, since the map contains several objects, the mapping-localization mechanism should be endowed with retrieval capabilities to search the closest

objects in the map to the current robot positioning in order to avoid using the whole set of objects, which increases substantially the complexity of the algorithm. This leads to a local map of the environment, used for updating and data association purposes. This helps in the decision making process of adding a new object to the map or not. Indeed, if, for example, the sensors are observing an object at much closer range than the closest mapped object, then the observed object is added to the map as described earlier.
With the ability to know the robots location and correctly add objects to the map around that location the system can map out the environment the robots are in. To map the perimeter of the environment a robot will travel forwards constantly checking its sensors. When a sensor detects an object any necessary adjustments are made to the map as described above, then the robot turns to align itself with the object and then continues to travel forward. At set intervals (set by the required resolution of the map and robot speed) the robots location is updated and the object detected is added to the map.

2.4 User's Interface

The interface consists of three parts: the console, the display and the menu. These can be seen in Figure 4. The console is the most diverse aspect of the interface, in that it has the most uses. Firstly the internal workings of the system can be displayed in text format using the console. This can range from simply displaying the current state of the system (such as robot coordinates and orientation), to the most recent recorded values (such as sensor, turret and odometer readings), to the actual values being calculated and used in a process. The console also allows the user to enter custom data into the system, such as providing a filename to save a map as. Aside from the need to enter filenames of maps to load or save the console can be mainly ignored for general system usage.

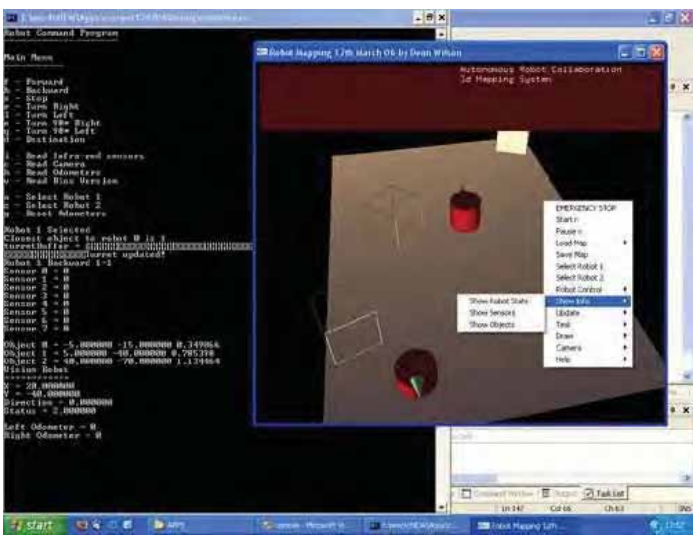


Fig. 4. Example of screenshot showing the three components of the user interface.

The display shows a 3D map which represents the environment as it is known by the system. The map includes the ground, objects and the two robots. On the other hand, the interface also includes some help functionalities in the event the user was unsure how to start using the system. It is fairly brief and only covers the most common mishaps. The user can also display the system credits, which states the program author and completion date.

2.5 Optimal Resource Allocation

Due to discrepancy between processing capability of PC and robot's controller, it was necessary to take this into account when designing the interaction between host PC and robots. This can be ensured by using a delay function to ensure the sensors are being queried at reasonable time. Besides to avoid inconsistency in requesting information from different robot's sensors, another small delay of around 0.05 seconds between sending a request for a sensor or odometer update and reading in characters from the receive buffer (reading the response) is inserted. The turret returns many more characters, so it was necessary to use such delay, anything less and some of the characters do not get received.
To allocate the best use of available resources, the 'reading' process was split into 'update' and 'show'. Rather than have the program poll the robot continually every time it wanted to make a decision, the readings are updated once and then stored on the PC. The program can then access these stored readings as many times as it wants, as fast as it wants without putting further strain on the robots. 'Show' refers to accessing these stored readings, whereas 'update' refers to polling the robots to update the stored readings with current data. Obviously the update process needs to be called periodically before the stored readings get too out of date. This design improved system efficiency greatly. It also allows the system to fully analyse a specific time index before moving onto the next. For example when checking the sensors for object detection the stored sensor readings can be updated once. An individual analysis of the reading of each of the sensors at that time index can then be made, and any necessary processing done. A separate update process eliminates the need to poll the robot once for each of the 8 sensors, the polling of which would incur a 0.05second delay for each sensor.

3. Kalman filter and SLAM models

The aim of this section is to investigate the stochastic models underlying the SLAM or simultaneous robot localization and map building. First let us describe the standard Kalman filter approach without recourse to SLAM.

3.1 State model

Using the incremental moving l_k^r and l_k^l of the right and left wheel, respectively, obtained by reading the encoder sensor of the robot, one can estimate the pose of the robot given in term of x-y coordinate of a reference point in the robot, usually taken as the centre of the robot and the orientation of the robot with respect to horizontal axis as it can be seen in Figure 3. The prediction model giving the state of the robot $(x_k, y_k, \theta_k)^T$ based on previous state $(x_{k-1}, y_{k-1}, \theta_{k-1})^T$ and the incremental encoder readings is given by the expression:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \frac{l_k^r + l_k^l}{2}\cos\theta_k \\ y_k + \frac{l_k^r + l_k^l}{2}\sin\theta_k \\ \theta_k + \frac{l_k^r - l_k^l}{E} \end{bmatrix} + \eta_k \tag{1}$$

where n_k stands for Gaussian zero-mean noise pervading the state components x, y and θ; that is, $\eta_k \mapsto \mathrm{N}([0\,0\,0]^{\mathrm{T}}, Q)$, where Q is a 3x3 noise variance-covariance matrix, usually taken as a fixed symmetric definite matrix. E is the distance between the wheels (left and right wheels) of the robot. Expression (1) assumes that the robot trajectory is linear between two consecutive time increments k and k+1, while the incremental moving of θ_k is assimilated to an arc of circle.

One designates $X_R(k) = [x_k \;\; y_k \;\; \theta_k]^T$ the state vector of the robot positioning. So, (1) can be rewritten as

$$X_R(k+1|k) = F_k(X_R(k|k)) + \eta_k \tag{2}$$

The quantity $F_k(X_R(k|k))$ represents the prediction of the estimate on X_R denoted $\hat{X}_R(k+1|k)$. Due to randomness pervading the estimation of X_R expressed in the form of additive Gaussian noise with known statistics (zero mean and Q variance-covariance matrix), the entity $X_R(k+1|k)$ is attached a Gaussian distribution probability with mean $\hat{X}_R(k+1|k)$ and variance-covariance matrix

$$P_{k+1|k} = \nabla F . P_{k|k} . \nabla F^T + Q \,. \tag{3}$$

Where ∇F indicates the Jacobian (with respect to x_k, y_k and θ_k) of the state transition function F, i.e.,

$$\nabla F = \begin{bmatrix} \frac{\partial x_{x+1}}{\partial x_k} & \frac{\partial x_{x+1}}{\partial y_k} & \frac{\partial x_{x+1}}{\partial \theta_k} \\ \frac{\partial y_{x+1}}{\partial x_k} & \frac{\partial y_{x+1}}{\partial y_k} & \frac{\partial y_{x+1}}{\partial \theta_k} \\ \frac{\partial \theta_{x+1}}{\partial x_k} & \frac{\partial \theta_{x+1}}{\partial y_k} & \frac{\partial \theta_{x+1}}{\partial \theta_k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\frac{l_k^d + l_k^g}{2}\sin\theta_k \\ 0 & 1 & \frac{l_k^d + l_k^g}{2}\cos\theta_k \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

So, $X_R(k+1|k) \mapsto \mathrm{N}(\hat{X}_R(k+1|k), P_{k+1|k})$

Observation Model

The exteroceptive sensors of the robot consist of infrared range sensors and vision turret (for only one of the robots). Therefore, the observation consists of the range –distance d_i from the sensor location within the robot platform to the i^{th} object (whose x-y coordinates are (x_{B_i}, y_{B_i}) while the information issued from the vision sensor can be translated into the azimuth β_i indicating the pose of the object with respect to the horizontal axis. Notice that the distance d_i can also be measured from the centre of robot as suggested by Figure 3 due to knowledge of radius r of the robot. Now relating the state variables to the observation

leads to the following expression of the observation model

$$d_i(k) = \sqrt{(x_k - x_{B_i})^2 + (y_k - y_{B_i})^2} + r + v_1(k) \tag{5}$$

$$\beta_i = a\tan\left(\frac{y_{B_i} - y_k}{x_{B_i} - x_k}\right) - \phi_k + v_2(k) \tag{6}$$

Or in matrix formulation

$$\begin{bmatrix} d_i(k) \\ \beta_i(k) \end{bmatrix} = \begin{bmatrix} \sqrt{(x_k - x_{B_i})^2 + (y_k - y_{B_i})^2} + r \\ a\tan\left(\frac{y_{B_i} - y_k}{x_{B_i} - x_k}\right) - \phi_k \end{bmatrix} + v_k \tag{7}$$

Or, more generally,

$$z_j(k+1) = H_k(\hat{X}_R(k+1|k)) + v_k \tag{8}$$

The set of all measurements available at current time k+1 is denoted by $Z(k+1) = (z_1(k+1) \ \ldots \ z_n(k+1))$, where n stands for the total number of observations at time (k+1).

Similarly, $v_k \mapsto \mathrm{N}([0\,0]^{\mathrm{T}}, R)$, where R is a 2x2 noise variance-covariance matrix, usually taken as symmetric definite matrix.

It should be noticed that not both measurement equations are used necessarily simultaneously due to possible non-availability of either distance reading or camera reading. In such case, one only uses either v_1 or v_2 noise expressions, which are one-dimensional entities.

Kalman filter or extended Kalman filter (in case of nonlinear state or measurement equation) aims at finding the estimation $\hat{X}_R(k+1|k+1)$ of the robot's state $X_R(k+1|k+1)$ of the current state of the vehicle given the set of measurements. This is typically given as the expectation given the set of observation Z, i.e., $\hat{X}_R(k+1|k+1) = E[X_R(k+1|k+1)|Z]$. The uncertainty on such estimation is provided by the state variance-covariance matrix $P_{k+1|k+1}$, given as covariance on error of estimate:

$$P_R(k+1|k+1) = E[(X(k+1|k+1) - \hat{X}_R(k+1|k+1))^T (X(k+1|k+1) - \hat{X}_R(k+1|k+1))|Z].$$

These entities are determined using Kalkan filter equations, which proceeds recursively in two stages –prediction and update- whose expressions are given below:

$$P_R(k+1|k) = \nabla F.P_R(k|k).\nabla F^T + Q_k \qquad \text{(prediction of state covariance matrix)} \tag{9}$$

$$\hat{X}_R(k+1|k) = F(\hat{X}_R(k|k)) \quad \text{(prediction of state vector)} \tag{10}$$

$$S_R(k+1) = \nabla H.P_R(k+1|k).\nabla H^T + R \qquad \text{(variance-covariance of innovation matrix)} \tag{11}$$

$$K_R(k+1) = P_R(k+1|k).\nabla H^T.S_R^{-1}(k+1) \quad \text{Gain matrix} \tag{12}$$

$$P_R(k+1|k+1) = P_R(k+1|k) - K_R(k+1)\nabla H.P_R(k+1|k) \quad \text{(state covariance update)} \tag{13}$$

$$\hat{X}_R(k+1|k+1) = \hat{X}_R(k|k) + K(k+1).(Z(k+1) - H(\hat{X}_R(k+1|k))) \quad \text{(state update)} \tag{14}$$

Where ∇H represents the Jacobian of the measurement equation H, which in case that both distance and landmark location were used, is given by

$$\nabla H = \begin{bmatrix} \frac{\partial d_i}{\partial x_k} & \frac{\partial d_i}{\partial y_k} & \frac{\partial d_i}{\partial \theta_k} \\ \frac{\partial \beta_i}{\partial x_k} & \frac{\partial \beta_i}{\partial y_k} & \frac{\partial \beta_i}{\partial \theta_k} \end{bmatrix} = \begin{bmatrix} -\frac{x_{B_i} - x_k}{\sqrt{(x_{B_i} - x_k)^2 + (y_{B_i} - y_k)^2}} & -\frac{y_{B_i} - y_k}{\sqrt{(x_{B_i} - x_k)^2 + (y_{B_i} - y_k)^2}} & 0 \\ \frac{y_{B_i} - y_k}{(x_{B_i} - x_k)^2 + (y_{B_i} - y_k)^2} & -\frac{1}{(x_{B_i} - x_k)^2 + (y_{B_i} - y_k)^2} & -1 \end{bmatrix} \quad (15)$$

In case where only one single type of observation is available, then one uses on a single row of matrix ∇H .

So, the estimation of $X_R(k+1|k+1)$ follows the Gaussian probability distribution $N(\hat{X}_R(k+1|k+1), P_R(k+1|k+1))$.

Kalman equations (9-14) are recursive, so they only depend on previous state. At time 0 where initially no observations were made, one requires an initial guess of the state vector and variance-covariance matrix, $X_R(0|0)$ and $P_R(0|0)$, respectively, this allows us to determine the new estimate $X_R(1|1)$ and $P_R(1|1)$ given the observation vector Z(1).

Notice that since the measurements are usually sampled at lower rate than encoders (almost five to 10 times less), the prediction equations (9) and (10) are applied several times before calling up for update stage using expressions (11-14).

3.2 SLAM mode

The preceding development of Kalman filter model assumes that the landmarks (observed objects) and robot positioning are independent. For instance, if the absolute locations of landmarks are fully known, then the previous Kalman filter approach does make sense. However, in reality, as far as the construction of global map of environment is concerned and no absolute knowledge of the landmark location is priori given, the estimations of landmarks positioning are correlated and strongly influenced by the uncertainty pervading the robot's location. Indeed, as the robot moves forth and back through the environment, the uncertainty pervading the landmarks' locations will be influenced and since the overall set of landmarks are linked through geometrical entities like wall, corners, etc, such uncertainty would propagate through overall set of landmarks. On the other hand, as all the observations (landmarks) are implicitly linked to robot state such uncertainty would also affect the robot state estimate X_R. This has given rise to the idea of simultaneous mapping and localization using estimation-theoretic methods known as SLAM. Work by Smith and Cheesman (1986) and Durrant-White (1988) established a statistical basis for describing relationships between landmarks and manipulating geometric uncertainty. Smith et al. (1990) established that if a mobile robot is moving through unknown environment and taking relative observations of landmarks, then the estimates of these landmarks are all necessarily correlated with each others because of common error in estimated robot location. As a result of this, a consistent estimation would require a joint state composed of both robot state and each landmark position leading to an augmented state vector. However as a result of increasing number of landmarks, the dimension of such state vector increases accordingly, which often induces further challenges in terms of computational complexity, convergence behaviour, conflict resolution, among others (Durrant-White and Bailey, 2006; Martinili et al., 2003).

More specifically, let $X_{L_i} = (x_{l_i}, y_{l_i})^T$ be the state of the ith feature or landmark given in terms of x-y Cartesian coordinates. First, one assumes the environment be static. This assumption is very common and trivial if the objects are not dynamic. Indeed, tracking moving objects is not considered of much value for the navigation purpose. So, the dynamic model that includes both landmark and robot's state becomes

$$\begin{cases} X_R(k+1|k) = F(X_R(k|k)) + \eta_k \\ X_L(k+1|k) = X_L(k|k) \end{cases} \tag{16}$$

Where $X_L = ((x_{L_1}\ \ y_{L_1})\ (x_{L_2}\ \ y_{L_2}) \ldots (x_{L_N}\ \ y_{L_N}))^T \in \Re^{2N}$ represents the set of all N landmarks identified up to current time. Loosely speaking, in some literature N is set as an arbitrary total number of landmarks that may exist in the environment, while it is common that the value of N varies within time due to update of the environment and addition of new landmarks to the map. So, the new state vector will be $X = (X_R, X_L)^T \in \Re^{2N+3}$.

The augmented state transition model for the complete system can now be written as

$$\begin{bmatrix} X_R(k+1|k) \\ x_{L_1} \\ y_{L_1} \\ . \\ . \\ x_{L_N} \\ y_{L_N} \end{bmatrix} = \begin{bmatrix} F_k & 0 & \ldots & 0 \\ 0 & 1\ 0 & \ldots & 0 \\ 0 & 0\ 1 & \ldots & 0 \\ . & & . & \\ . & & . & \\ 0 & 0 \ldots & 1\ 0 & \\ 0 & 0 \ldots & 0\ 1 & \end{bmatrix} \begin{bmatrix} X_R(k|k) \\ x_{L_1} \\ y_{L_1} \\ . \\ . \\ x_{L_N} \\ y_{L_N} \end{bmatrix} + \begin{bmatrix} \eta_k \\ 0 \\ 0 \\ . \\ . \\ 0 \\ 0 \end{bmatrix} \tag{17}$$

Accordingly, the new state Jacobian matrix ∇F^e (one denotes F^e for extended state transition F) will be

$$\nabla F^e = \begin{bmatrix} \nabla F & 0_{3xN} \\ 0_{Nx3} & 1_{2Nx2N} \end{bmatrix} \tag{18}$$

Where 0_{3xN} stands for 3 x N zero matrix , similar definitions hold for 0_{Nx3} and 1_{2Nx2N}.

The new observation model can be written

$$z_i(k+1) = H_k(\hat{X}_R(k+1|k)) + 0_{Nx1}.X_L + v_k \tag{19}$$

Similarly, the new Jacobian ∇H^e of the observation model reads as (assuming that only a single landmark is observed at a given time):

$$\nabla H^e = \begin{bmatrix} \dfrac{\partial \mathrm{H}}{\partial \mathrm{X_R}} & 0_{2x1} \ldots 0_{2x1} & \dfrac{\partial \mathrm{H}}{\partial (\mathrm{x_{L_i}}, \mathrm{y_{L_i}})} & 0_{2x1} \ldots . 0_{2x1} \end{bmatrix}, \tag{20}$$

For example in the case of both (5-6) were used, we have

$$\nabla H^e = \begin{bmatrix} -\dfrac{x_{B_i} - x_k}{\Delta_i} & -\dfrac{y_{B_i} - y_k}{\Delta_i} & 0\ 0 \ldots 0 & \dfrac{x_{B_i} - x_k}{\Delta_i} & -\dfrac{y_{B_i} - y_k}{\Delta_i} & 0 \ldots . 0 \\ \dfrac{y_{B_i} - y_k}{\Delta_i^{\,2}} & -\dfrac{1}{\Delta_i^{\,2}} & -1\ 0 \ldots 0 & -\dfrac{y_{B_i} - y_k}{\Delta_i^{\,2}} & \dfrac{1}{\Delta_i^{\,2}} & -1\ 0 \ldots 0 \end{bmatrix} \tag{21}$$

With $\Delta_i = \sqrt{(x_{B_i} - x_k)^2 + (y_{B_i} - y_k)^2}$

(21) can also be rewritten as

$$\nabla H^e = [H_1 \ \varnothing_1 \ \mathrm{H}_2 \ \varnothing_2] \tag{22}$$

Where $\varnothing_1$ and $\varnothing_2$ stand for all null elements located in (20) or (21). Notice that most elements of both ∇F^e and ∇H^e are null elements.

From implementation perspective of the (extended) Kalman filter in the sense of expressions (9-13), a naïve implementation consists to compute the predicted state variance-covariance:

$$P_{k+1|k} = \nabla F^e . P_{k|k} . \nabla F^{e^T} + Q \tag{23}$$

Strictly speaking the above operation induces a cubic complexity in the number of landmarks. However, intuitively since only the robot state variables are involved in the observation, the covariance should be simplified accordingly. For this purpose, by distinguishing parts related to robot state and those linked to landmark state in matrix P as

$$P = \begin{bmatrix} P_R & \mathrm{P_{RL}} \\ \mathrm{P_{RL}^T} & \mathrm{P_L} \end{bmatrix}, \tag{24}$$

so the prediction stage (23) boils down to

$$\begin{aligned} P_{k+1|k} &= \begin{bmatrix} \nabla F & \varnothing \\ \varnothing^T & 1 \end{bmatrix} \begin{bmatrix} P_R(k) & \mathrm{P_{RL}}(k) \\ P_{RL}^T(k) & \mathrm{P_L(k)} \end{bmatrix} \begin{bmatrix} \nabla F^T & \varnothing^T \\ \varnothing & 1^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} Q & \varnothing \\ \varnothing & \varnothing_1 \end{bmatrix} \\ &= \begin{bmatrix} \nabla F . P_R(k) . \nabla F^T & \nabla F . P_{RL}(k) \\ (\nabla F . P_R(k))^T & \mathrm{P_L}(k) \end{bmatrix} + \begin{bmatrix} Q & \varnothing \\ \varnothing & \varnothing_1 \end{bmatrix} \end{aligned} \tag{25}$$

It has been shown that the evaluation of this matrix requires approximately 9(N+3) multiplications (Guivant and Neboit, 2001).

Similarly, in the updating stage, by rewriting $P_k . \nabla H^{e^T} = P_R H_1^T + P_L H_2^T$ leads to a cost, which is proportional to (N+3), so the evaluation of the covariance update is $\sim O(N^2)$.

Moreover, it has been shown that it is not necessary to perform full SLAM update when dealing with a local area. So, the complexity can even get substantially reduced accordingly. More formally, assuming the state vector is divided as $X = [X_A \ \mathrm{X_B}]^T$ with $X_A \in \Re^{N_A+3}$ and $X_B \in \Re^{N_B+3}$, $N = N_A + N_B$ (Guivant and Nebo, 2001). The states X_A can be initially selected as the state of all landmarks located in the neighborhood of the vehicle in addition to the three states of the vehicle, while X_B corresponds to the states of all remaining landmarks. The hint is that at a give time, the observations are only related to X_A. Accordingly,

$$\nabla H^e = \begin{bmatrix} \dfrac{\partial \mathrm{H}}{\partial \mathrm{X_A}} & \dfrac{\partial \mathrm{H}}{\partial X_B} \end{bmatrix} = [H_A \ \varnothing] \tag{26}$$

Consequently, given $P = \begin{bmatrix} P_A & \mathrm{P_{AB}} \\ P_{AB}^T & \mathrm{P_B} \end{bmatrix}$, one induces

$$\mathrm{S=H.P.H^T + R} = H_A . P_{AA} . H_A^T + R \tag{27}$$

And the filter gain

$$W = \begin{bmatrix} P_{AA} H_A^T S^{-1} \\ P_{AB}^T H_A^T S^{-1} \end{bmatrix} = \begin{bmatrix} W_A \\ W_B \end{bmatrix} \tag{28}$$

In other words, the innovation matrix and matrix gain W_A are independent of remaining landmarks X_B. When the vehicle departs from this local area the information will be propagated to global landmark. So, the entities X_B, P_{AB} and P_{BB} will only be determined when the vehicle moves away from the local area. It has been shown that the complexity of update in such case is of order $O(N_A^2)$ and since N_A is in general much smaller than N_B, the gain in terms of complexity becomes significant. This reduction method is known as compressed (extended) Kalman filter in (Guivant and Nebo, 2001). Williams (2001) has put forward the Constrained Local Submap Filter approach in which both the relative state of each landmark with respect to local map as well as its global coordinate with respect to the global map are carried out. The method maintains an independent, local submap estimate of the features in the immediate vicinity of the vehicle.

An ultimate problem which arises from the above submapping is the selection of the local area. Several approaches have been investigated for such purpose. One conventional approach consists of dividing the global map into rectangular regions with size at least equal to the range of the external sensor. So, at each position, one may consider for instance the eight or twenty fourth neighbouring cells as suggested in (Guivant and Nebo, 2001).

In the context of our work, we rather adopted an approach close to that developed by Dissanayake et al. (2001). In this course, given a time interval h_T, a two-stage selection process is carried out:

- First, one maintains all landmarks that have been seen by the vehicle within the time interval h_T. Alternatively, authors in (Dissanayake et al., 2000) used a predefined distance travelled by the vehicle.
- Next, among the above set of landmarks, one selects only those, which are the most informative in the sense of landmark variance-covariance matrix. For this purpose, the reciprocal of the trace of such variance-covariance matrix was used as a tool to evaluate the extent of the information content. Consequently, from the set of landmarks, only those landmarks whose information content in the above sense is beyond some threshold are considered. The value of the threshold is here taken to be a function of the information content associated to the fully defined prior landmarks concerning the border of the environment as will be pointed in the map initialization section.

3.3 Convergence properties

As far as the construction of the submap is concerned, the aspect of convergence becomes crucial. From theoretical perspective, some appealing results have been reported by Dissanayake et al. (2001). Especially given the decomposition (24), it has been proven that

i) The determinant of any submatrix of the map covariance matrix P_L decreases monotonically as successive observations are made
ii) In the limit case (at time infinity), the determinant of P_R tends towards zero, so the landmark estimates become fully correlated.
iii) In the limit case, the lower bound on the covariance matrix associated with any single landmark is determined only by the initial covariance of the vehicle estimate P_R.

The above testifies on the steady state behavior of the convergence of the landmark estimates. Especially, it stresses that as the vehicle moves on the environment the

uncertainty pervading the landmark estimations reduces monotonically. The estimation of any pair of landmarks becomes fully correlated in the sense that if one landmark is known with full certainty, the others can be known with full certainty too. The individual landmark variances converge toward a lower bound determined by initial uncertainties in vehicle position and observations as indicated by matrices $P(0|0)$ and R.

On the other hand, Julier (2003) has investigated the effect of adding noise to the long term behaviors of SLAM and has shown that:

i) If the steady state covariance will not be degraded, the computational and storage cost increase linearly with the number of landmarks in the map;
ii) Even if the steady state covariance is preserved, local performance can be unacceptably high;
iii) If the solution causes the steady state covariance to degrade, the addition can only be a finite number of times.

This entails that it is more appropriate to maintain the full correlation structure of all landmarks within the operational area of the vehicle.

On the other hand, from the observability perspective, it has been shown that the Riccati equation in P (that follows from update expression (13)), e.g., (Andrade-Cetto and Sanfeliu, 2004), which can be rewritten as:

$$P = \nabla F.(P - P.\nabla H^T(\nabla H.P.\nabla H^T + R)^{-1}.\nabla H.P).\nabla F^T + Q$$

converges to a steady state covariance only if the pair $(\nabla F, \nabla H)$ is fully observable. In addition if the pair $(\nabla F, I)$ is fully controllable, then the steady state covariance is a unique positive definite matrix, independent of the initial covariance $P(0|0)$.

3.4 Map Initialization

Initialization is required to infer the number of landmarks N as well as their x-y coordinates, which will be used in the SLAM model. Several studies have explored the initialization of the map through sensor scan, using, for instance, sonar-like measurements (Chong and Kleeman, 1999; Ip and Rad, 2004), which an initial value of landmarks. While other studies assumed the initial map is initially empty, and as soon as an observation gets reinforced by other observations, it will be promoted to a landmark (Dissanayake et a., 2001). Both approaches can be used in our study. Indeed, the use of initial mapping using a single sensor can be accomplished using the vision sensor. So, in the light of the emerging works in the bearing-only SLAM, one can think of the robot using a single rotation at discrete sample intervals, repeated at two different robot's locations, would allow us in theory to determine initial set of landmarks. However, the data association problem in such case becomes difficult. While the second approach is trivially straightforward where the initial state vector reduces to robot state vector. In our study, in order to make use of the geometrical environment constraints at one hand, and on the other hand, avoid the nontrivial data association problem due to the limited sensory perception, one assumes that the boundary of the environment is fully known. Consequently, the four corners of the rectangular environment are taken as fully determined landmarks. This also allows us to set up a geometrical consistency test in the sense that as soon as the perceived landmark is located beyond the border limit, it is systematically discarded.

Therefore, initially, the state vector is $X = [X_R \quad x_{L_1} \; y_{L_1} \; x_{L_2} \; y_{L_2} \; x_{L_3} \; y_{L_3} \; x_{L_4} \; y_{L_4}]^T$

3.5 Data Association and Map building

Data association has always been a critical and crucial issue in practical SLAM implementations. That is because it governs the validation of the new landmarks and the matching of the observation (s) with the previously validated landmarks. On the other hand, an incorrect association of the observation to the map can cause the filter to diverge. Given the knowledge of the geometrical boundary of the environment, two validation tests are carried out:

- *Geometrical validation test*: This is a basic check to test whether the location of the observation is within the environment boundary. This is mainly meant to remove possible outliers and noise measurement observations.
- *Statistical validation test*: This uses the statistical properties of the observations as well as landmarks as a tool to achieve the matching. Especially, the nearest neighbour association is taken as the closest association in statistical sense. For this purpose, one first needs to translate the range/bearing observation into landmark locations. In case where measurement coincides with range measurement, e.g., $z(k) = r_L$, we have

$$\begin{cases} x_{L_z} = x_1(k) + r_L.\cos(\theta_k + \alpha_j) \\ y_{L_z} = y_1(k) + r_L.\sin(\theta_k + \alpha_j) \end{cases} \tag{29}$$

With $X_R(k) = [x_1(k) \;\; x_2(k) \;\; \theta_k]^T$ and α_j stands for the azimuth of the jth robot's sensor that detected the underlying landmark, with respect to robot axis. Putting (29) in matrix formulation as $X_{L_z} = T(X_R(k), r_L)$, the variance-covariance of the landmark estimate is given by

$$P_{L_z} = \nabla T_{x_1 y_1 \theta_k}.P_R.\nabla T_{x_1 y_1 \theta_k}{}^T + \nabla T_{r_L}.R.\nabla T_{r_L}{}^T \tag{30}$$

Where R stands for the z(k)'s covariance, and P_R for the (updated) robot state vector variance-covariance matrix as determined by the filter.

Now given the vector X_{L_i} sympolized by (29) and given a set of confirmed landmarks $(L_1, P_{L_1}),...,(L_m, P_{L_m})$, where (L_i, P_{L_i}) stands for the first and second statistics of the ith landmark, the measurement z(k) is associated with the jth landmark if:

$$(X_{L_z} - L_j)^T (P_{L_z} + P_{L_j})^{-1} (X_{L_z} - L_j) \le d_{\min} \tag{31}$$

where $d_{\min}$ is some validation gate. The threshold $d_{\min}$ can be determined by noticing that the left hand side part of the inequality in (31) is χ^2 distributed, so by choosing the null hypothesis and the confidence level, the $d_{\min}$ value is straightforward.

Therefore, if the above condition holds only for one single landmark, then the underlying observation z(k) is associated with that landmark. Otherwise, if the inequality holds for more than one landmark, the observation is then omitted, meaning that under the current level of confidence, the statistical test cannot lead to a matching. Obviously, it is still possible to narrow the confidence level such that the validation gate $d_{\min}$ decreases, which may result in resolving the conflict among the possible candidates.

On the other hand, if the above inequality cannot hold indicating that there is no landmark that may match the current observation, then such observation can be considered as a new landmark. Once the validated landmarks are constituted of a set of Cartesian points, a

geometrical fitting allows us to group these features into high level geometrical entities constituted of two main feature landmarks: line segment, if the underlying points are sufficiently aligned up, and corner. The former is modelled by the extreme points of the segment while the corner by the x-y coordinates of the underlying Cartesian point.
In other studies, e.g., (Guivant and Neboit, 2001), a non-matched observation will be treated as a potential landmark (so maintaining at each time a set of confirmed landmarks and a set of tentative landmarks), and will not be promoted to a confirmed landmark until sufficient number of observations are found matching this potential landmark in the above statistical sense. Strictly speaking such reasoning cannot be applied in our experiment due to the lack of redundant data and limited navigation tasks. Therefore, as soon as the statistical test fails for all set of landmarks, the new observation is automatically promoted to a confirmed new landmark, unless the geometrical validation test fails as well in which case, the observation is fully ignored. Adding a new landmark to the new set of already confirmed landmarks will obviously result in an augmented state vector.

3.6 Discussions

- It should be noted that the above data association reasoning relies heavily on the range of sensors because the information about the landmark location can be directly inferred as according to (29). However, the use of bearing sensor would be beneficial if the two robots were equipped with vision turret. In this course, the use of bearing information from two different robot locations would allow us to infer the x-y coordinate of the associated landmark, assuming the data association is very simple in the sense that at each robot location the sensory information identified the same object, which in reality is not always the case.
- The representation of the landmark in this study is made easy by choosing a Cartesian point as a geometric primitive, which are later combined to form more generic feature like segment line and corner. However, such choice, even if it is motivated by the limited sensory modality of Khepera, can also be questioned. Dufourd and Chatila, (2004) provide a comparison of space-based, grid-based and feature based map formats. Lisien et al. (2003) suggested to combine topological and feature based mapping where topological methods are used for planning feature based mapping. This leads to what is referred to as hierarchy SLAM.
- The restriction concerning the validation of new landmark using only geometrical and statistical tests is also shown to be limited. Indeed, it can make sense for more point-based-landmarks but it is difficult to be justified for more realistic geometric patterns. Several studies using SLAM with range-only sensors (Leonard et al., 2003) and bearing-only sensors (Lemaire et al., 2005; Deans and Hebert, 2000) proved that a single measurement is insufficient to constrain landmark location, instead several observations are necessary to confirm or delete the tentative landmark.
- The use of Mahalanobis distance as in (31), even it has proven to be successful, can also be questioned. Alternatives include Multiple hypotheses tree using Bayes' theorem, but, this raises the complexity of the algorithm due to the cost of maintaining separate map estimates for each hypothesis, and the pruning decision. Montemerlo and Thrum (2003) suggested a fast SLAM algorithm based on the idea of exact factorization of posterior distribution into a product of conditional

landmark distributions and a distribution over robot paths. Instead of geometrical feature landmarks, Nieto et al. (2005) has suggested a methodology to deal with features of arbitrary shapes, where each landmark is defined by a shape model incorporating an embedded coordinate frame, so, the map is constituted of a set of landmark frame locations. Eliazar and Parr (2004) advocated the use of grid cell representation in conjunction with particle filter. Nieto et al., (2004) also used occupancy grid structure where each grid-cell is determined by a set of local landmarks in the overall SLAM map. The data association problem in such cases boils down to ambiguity in cell allocation. The latter can be solved by Bayes' like approach. However, the suggested algorithmic representation sounds very context dependent. Also, the choice of grid cell posterior as well as adapting the number of cells is very debatable in the literature. Wijesoma et al. (2006) advocated the use of optimization problem-based approach where the data association is formulated as a generalized discrete optimization problem where the cost function is constructed from joint likelihood of measurements in multiple frames and features. The minimization is subject to some environmental and rational constraints.

- The issue of landmark selection in suboptimal filtering as detailed is very debatable as well. Indeed, this boils down to the difficult trade-off of maintaining sufficient representation of the map which allows good estimates of robot pose versus reducing the map size to its nominal representation in order to reduce the computational complexity. Indeed, the crucial question is how much should we be looking back into the past such that all the visited landmarks will be maintained? Typically, there is no exact answer to this question as it is very much context dependent; that is, it requires knowledge of how often the vehicle visits the already perceived landmarks. The aspect of information content discussed in previous section requires also further analysis. Indeed, we adopted, similarly to Dissanayake et al. (2001), the reciprocal of the trace of the covariance matrix. However, other alternatives are also possible. This includes, for instance, Shannon entropy, Fisher entropy, among others.
- The map initialization adopted in this study contrasts with alternatives approaches in which either no prior knowledge is assumed leading to zero initial landmarks and the full scanning of the environment where the obtained landmark states will be updated as far as further observations reinforce or delete the initial knowledge. This study by assuming fully known boundary landmarks offers on one hand, an appealing opportunity for geometrical validation test in data association stage, and, on the other hands, allows more accurate association and filter update estimation as soon as one of these landmarks is part of the suboptimal map, which is part of the state vector. This is due to the fact that the variance-covariance matrices associated to these landmarks are close to null evaluation, which, in turn, affects, the estimation process of the filter. Kwork and Dissanayake (2004) used a multiple hypothesis filter to initialise landmarks based on a number of hypotheses.
- The issue of when the filter will be updated is also debatable. Indeed, while the computational complexity requirement tends to postpone the update as late as possible (Knight et al., 2001), the requirement of building a complete and a consistent map tends to prevent such postponement. However, this aspect is rather very context dependent.
- The extended Kalman filter has often been criticised in case of high nonlinearity of either the state or measurement equations, which led to the rapidly developing

Monte-Carlos based approaches (Montemerlo et al., 2003). However, the computational complexity get increased, while in case of relatively small robot speed, the performance of the extended Kalman filter still are acceptable.

3.7 Multiple Robot Localization

Important feature of our robotic system is the use of two different robots. Consequently, this offers the possibility of mutual collaboration in updating their current states. Intuitively, the basic scenario consists of making the vision robot turning around until the second robot is identified, and next the coordinates of the robots are matched, and updated accordingly. The identification of the robot by the vision turret is made possible through appropriate choice of black/white cover, which was selected different from objects in the environment.
More formally, let $X_R^i = (x_i \;\; y_i \;\; \theta_i)^T$ be the state vector attached to the ith robot. Similarly let P_R^i designate the associated variance-covariance matrix. So, for the two robots, the dynamic models can be rewritten in the light of (2) as:

$$X_R^1(k+1|k) = F_k^1(X_R^1(k|k)) + \eta_k^1 \tag{32}$$

$$X_R^2(k+1|k) = F_k^2(X_R^2(k|k)) + \eta_k^2 \tag{33}$$

With
$\eta_k^1 \mapsto \mathrm{N}([0\,0\,0]^{\mathrm{T}}, Q_1)$ and $\eta_k^2 \mapsto \mathrm{N}([0\,0\,0]^{\mathrm{T}}, Q_2)$. F_k^1 and F_k^2 correspond to odometric model of robot 1 and 2, respectively, similarly to (1).
Let $X = [X_R^1 \;\; X_R^2]^T$ be the state vector constituted of the two robot states. Then the associated variance-covariance matrix P can be rewritten as in (Martinelli et al., 2005):
$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$, with P_{ii} stands for P_R^i -variance-covariance of the state of ith robot.
The predicted variance-covariance and predicted state are evaluated as

$$P_{ii}(k+1|k) = \nabla F_k^i . P_{ij}(k|k) . \nabla F_k^{iT} + Q_i \quad \text{(i=1,2)} \tag{34}$$

$$P_{ij}(k+1|k) = \nabla F_k^i . P_{ij}(k|k) . \nabla F_k^{jT} \qquad \text{(i,j=1,2 and i} \neq \text{j)}. \tag{35}$$

$$\hat{X}(k+1|k) = [F_k^1(X_R^1(k|k)) \;\; F_k^2(X_R^2(k|k))]^T \tag{36}$$

Within the framework constituted of the single system of the two robots, assume, for instance, at a given time, the Robot 1 observes the Robot 2, this relative observation can be modelled as

$$z = h(X) + w, \text{ with } w \mapsto \mathrm{N}(0, R_w) \tag{37}$$

Where h corresponds to the model of the predicted observation, which, in case of relative bearing in terms of the two robot configurations, is given by:

$$h(X) = a\tan\left(\frac{-\sin\theta_1.(x_2 - x_1) + \cos\theta_1.(y_2 - y_1)}{\cos\theta_1.(x_2 - x_1) + \sin\theta_1.(y_2 - y_1)}\right) \tag{38}$$

Therefore, the update estimations given the (relative) observations are determined using the

standard (extended) Kalman filter equations by:

$$\hat{X}(k+1|k+1) = X(k+1|k) + P(k+1|k).\left(\nabla h.P(k+1|k).\nabla h^T\right)^{-1}\left[z - h(X)\right] \quad (39)$$

$$P(k+1|k+1) = P(k+1|k) - P.\nabla h^T\left(\nabla h.P(k+1|k).\nabla h^T\right)^{-1}\nabla h.P(k+1) \quad (40)$$

With,

$$\nabla h = [\nabla_{X_R^1} h \quad \nabla_{X_R^2} h] \quad (41)$$

$$\nabla_{X_R^1} h = \left[\frac{y_2 - y_1}{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad -\frac{x_2 - x_1}{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad -1\right] \quad (42)$$

$$\nabla_{X_R^2} h = \left[-\frac{y_2 - y_1}{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad \frac{x_2 - x_1}{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad 0\right] \quad (43)$$

The above estimation provides an estimation of the states of both robots 1 and 2 in terms of state vector estimation as well as the variance-covariance matrix associated to each one. Moreover, the estimation also provides an indication concerning the correlation among the estimations of both robots as quantified by the quantity P_{12} and P_{21} of the matrix P.

Notice that in case where the two robots are within range sensor reach, the observation also includes the relative distance, in this case, the observation model h in (37) boils down to:

$$h(X) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (44)$$

This leads to a Jacobian measurement matrix $\nabla h = [\nabla_{X_R^1} h \quad \nabla_{X_R^2} h]$ with,

$$\nabla_{X_R^1} h = \left[-\frac{x_2 - x_1}{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad -\frac{y_2 - y_1}{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad 0\right] \quad (45)$$

$$\nabla_{X_R^2} h = \left[\frac{x_2 - x_1}{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad \frac{y_2 - y_1}{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad 0\right] \quad (46)$$

The (update) estimation process is still made of expressions (39-40).

Note that the update equations can be used either via one of the measurement equation (38) or (44) or both of them depending on availability of measurements as the filter can be used recursively with respect to measurements as well.

Remark that the state vector in this situation does not include landmark states due to the fact that the process of multi-localization is not used as often as mapping. It is only employed when the process of robot collaboration is called upon, or when the quality of the estimation as quantified by the matrix P is too poor. This is because the process of looking for Robot 2 by Robot 1 using the vision turret is relatively costly.

Alternatively, one may think of using both the mapping and multi-robot simultaneously. In this case, the rational is to leave the state of landmarks with only one robot state vector, e.g., $X = [X_R^1 \ \mathrm{X_L} \ \mathrm{X_R^2}]$. However, if both robots navigate within the environment for sufficient time interval, then a rational is to maintain two different maps; that is, each robot will maintain its own map of the environment, leading to an augmented state vector. The estimation process is somehow similar to that already developed in this section, where

observations might be either relative observation with respect to a second robot where one robot identified another robot, or relative observation with respect to landmark (s).

3.8 Overview of general approach

Figure 5 summarizes the general scheme of our SLAM approach involving multi-robot collaboration and localization. The vector U mentioned in Figure 5 corresponds to the command vector in terms of incremental moving of the right and the left wheel of the robot used by the encoder.

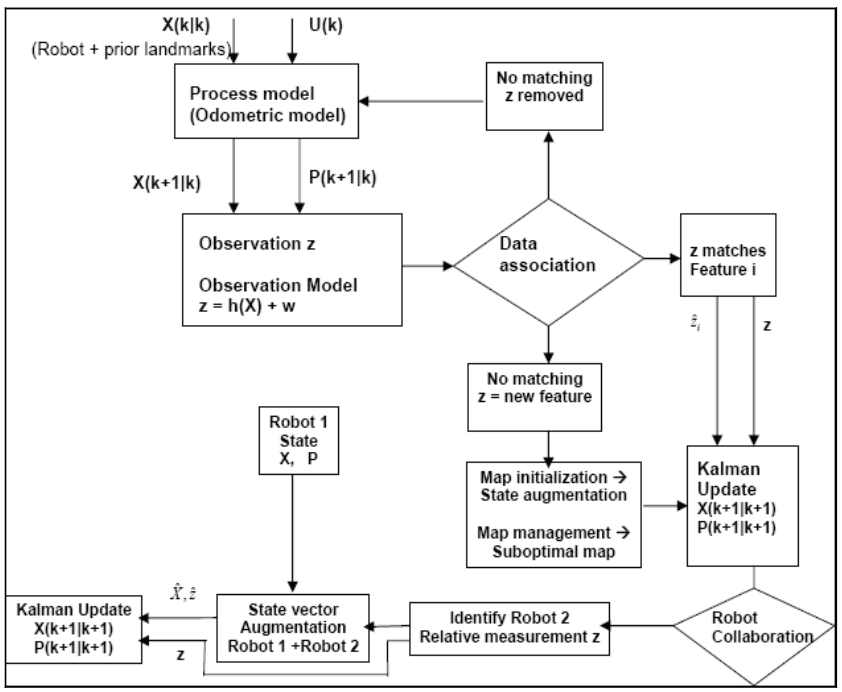


Fig. 5. General scheme of the SLAM –collaboration process.

Note that the possible movement of Robot 1 is not represented, but can obviously carry out the same reasoning as Robot 2 in terms of stages: i) prediction using encoder model; ii) measurements & measurement model; iii) data association based on Mahalanobis distance that determines the validity of the measurement and whether it corresponds to an already identified landmark or to a new landmark, which, in turn, induces either state augmentation, or just a false alarm that needs to be discarded; iv) mapping and suboptimal map construction based on the viewing field of the sensor and the timing frame; v) update with respect to (extended) Kalman filter equations.

4. Testing and results

Figure 6 shows an example of mapping the entire perimeter of the environment using one single infrared sensor. On the left hand side of Figure 6 is shown the virtual representation of the perceived environment in terms of a set of landmarks. Remark that the number of landmarks depends on the robot speed and sampling frequency of infrared sensors. Also by increasing the resolution to 'High' the walls have a more solid and continuous look.
The prior landmarks consisting of the four corners of the environment are used here for geometrical validation purpose of all represented landmarks but they are not plotted in figure 6 because they are not perceived by the robot sensors.

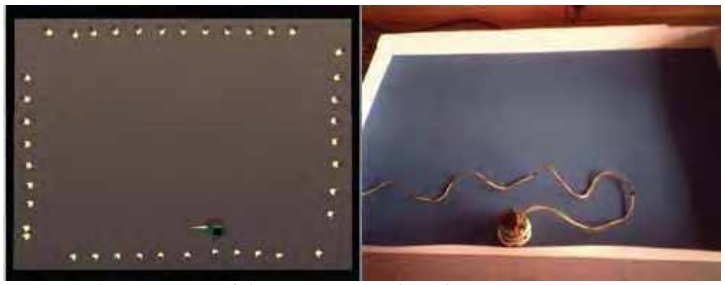


Fig. 6. Mapping the perimeter of the environment by robot.

Figure 7 shows another example of robot configuration and map building using SLAM algorithm while using curved object as a set of bounded plans in the environment. On the left hand side of Figure 7, the objects are modelled as a successive set of landmarks (virtual Cartesian points). The feature landmarks can therefore be extracted from these Cartesian points. Intuitively, one can identify at least eight segment lines together with eight corners.

Fig. 7. Example of environment mapping.

Ultimately linked to the preceding is the issue of loading and saving environment map. This is carried out by the user's interface. Indeed, to ensure the maintaining of previous maps even if the system switched off, the map is saved in file format and loaded upon request. To recreate the map from the interface, one requires: i) select "Load Map" previously saved

map in the interface; ii) click the console window to make it the active window; iii) type in the name of the map to load, in this case "completeperimeter" and press enter. The console will then display object and robot information that is being loaded into the map, the result of which will be displayed in the map window. An example is shown in Figure 8 that retrieves the perimeter map, which is kept along all experiments carried out in this study.
Now in order to quantify the quality of the estimation, Figure 9 displays the error in vehicle localization. The errors are quantified for the environment shown in Figure 7. In the latter the true positioning are measured with respect to Cartesian coordinate chosen at the left hand corner of the environment.

Fig. 8. Console output showing map objects being successfully loaded.

Figure 9 shows the actual error in estimated vehicle location in both x and y coordinates (solid line) as a function of time increments, which summarizes the vehicle movement as in Figure 7. The Figure also displays the 95% confidence limits, or two-times standard deviation around estimates (represented in dashed line), which is driven from the state covariance P by selecting the relevant component of P pertaining to $P_{R_{xx}}$ and $P_{R_{yy}}$ and taking the square root. As it can be noticed from the plot, the vehicle error is clearly bounded by the confidence limits of estimated vehicle error, which shows the consistency and convergence of the underlying SLAM algorithm. This also demonstrates that the algorithm clearly yields consistent and bounded errors.

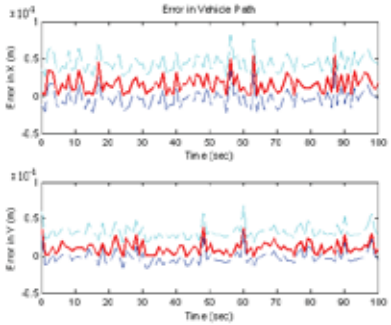


Fig. 9. Error in vehicle location estimate in x and y coordinates and the associated 95%

confidence bounds (dotted line) obtained from state estimate covariance matrix.

Figure 10 shows the evolution of the innovation of range measurement over the time. The bounded limit curves corresponding to 95% confidence limit drawn in the same plot indicate the consistency of the filter outcomes and estimation process. Note that only the range innovation is displayed here because only range measurements were used in the map initialization and management. The bearing measurements are only used for robot collaboration.

Figure 11 provides an estimation of the quality of landmark estimations in terms of standard deviation with respect to x and y coordinates as quantified by landmark variance-covariance matrices P_{L_i}. We restricted to three chosen landmarks consisting of the first landmark encountered and two others.

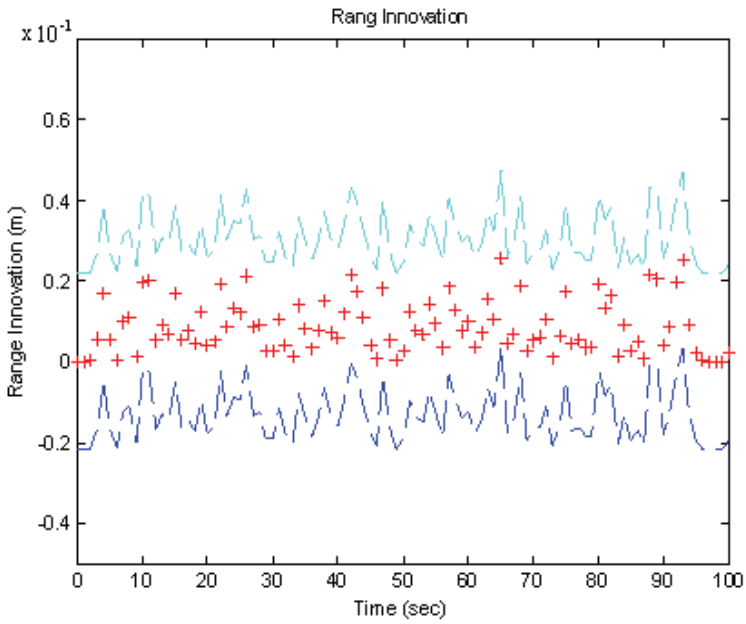


Fig. 10. Range innovation with associated 95 confidence bounds.

This explains why the estimation process does not start at time t=0 for all landmarks as landmark 2 and 3 are not identified yet at the beginning. The plots were obtained after using least square fitting in standard deviation results in order to obtain a smooth representation. The graph shows clearly a decreasing tendency in the standard deviation over time, which indicates consistent and convergence of the estimation process. On the other hand, as far as the local map is concerned, the display shows that despite the use of local map while maintaining all information with computational effective algorithm whose complexity is proportional to the number of landmarks, the performances of the filter in terms of variance-covariance matrix still behave consistently. This demonstrates that the developed cost effective SLAM algorithm does not suffer from divergence. Notice that in Figure 11, the initial

uncertainty ascribed to landmarks 2 and 3 is always less than that of landmark 1. This is because while the vehicle is moving around, its associated variance-covariance matrix tends to decrease, which, in turn, reduces the uncertainty of the identified landmarks.

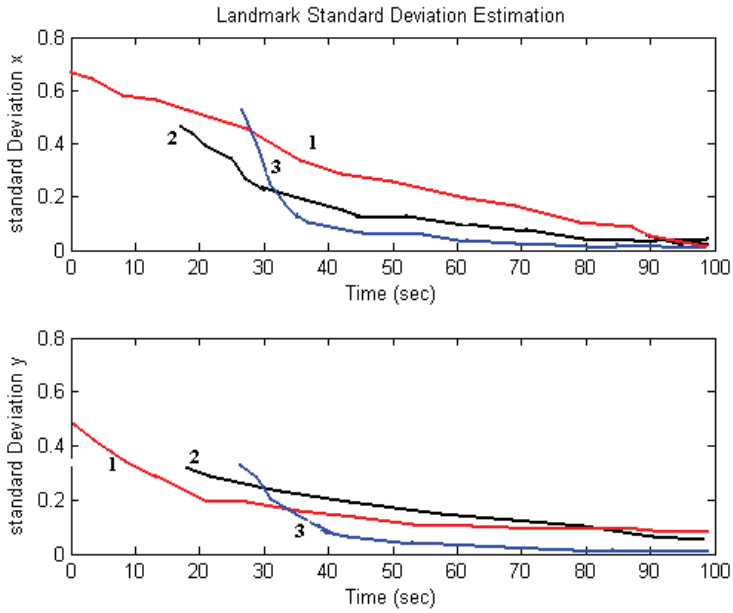


Fig. 11. Landmark Estimation In terms of Standard Deviations in x and y coordinates.

Now considering the situation in which two robots were used to accomplish a task, in order to quantify the performance of the algorithm, one considers the scenario shown in Figure 1 of this chapter. In the latter Robot 1 (vision robot) identifies an object (white box) in the environment and sends a command to Robot 2 to go to its location. The figure displays the performance of the estimation of the Robot 2 localization in terms of x and y standard deviation together with the associated 95% confidence bounds. Clearly, the decreasing behaviour together with the conservative bounds testify on the consistent and convergent estimation process along the collaboration scheme. Notice that the figure shows a local region where standard deviation does increase. This, in fact, corresponds to the region of the environment where the robot goes straight to the object and no observations were taken, so the filter only relies on prediction part of the filter, which trivially tends to increase the variance-covariance estimate. Intuitively by exchanging relative pose information, the states of the robots are updated in a centralized fashion. Note that Robot 1 once the image of the object after a single revolution, the robot becomes static. Consequently, the possibility of using Robot 2 to update the state of Robot 1 is very limited in this scenario.

Figure 13 shows the trajectory of Robot 2 while moving toward the target object and exploring the underlying object. In the same plot is displayed the true position of the vehicle in terms of x-y coordinates of the reference point in the robot platform. The estimated trajectory is represented in circles (o) while the true trajectory is drawn in start (*).

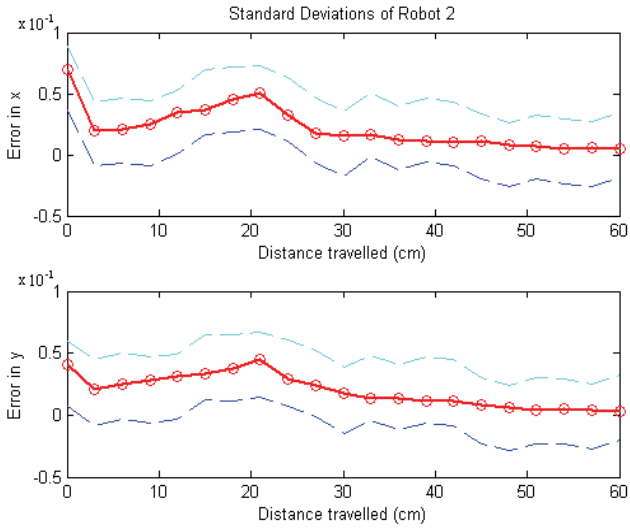


Fig. 12. Performances of Robot 2 in terms of standard deviation in x and y in case of robot-collaboration scenario.

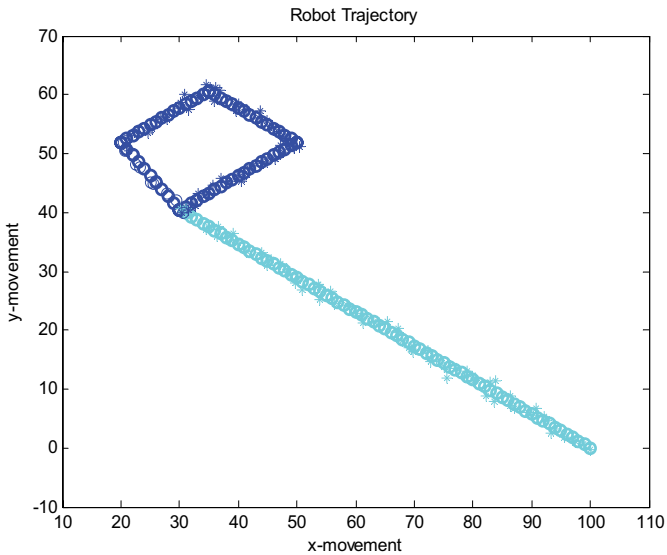


Fig. 13. Estimated Robot trajectory versus True trajectory.

5. Conclusion

This chapter investigated the state of the art of SLAM system whose aim is to simultaneously map the environment and provide accurate positioning of the vehicle where the state of both vehicle and landmarks are brought into a single estimation process. The estimation process supplies estimation of vehicle and landmarks in terms of mean and variance-covariance estimate of the state vector conditional on the whole set of measurements. The construction of the map involves appropriate initialization step in which some prior knowledge regarding the environment is reported, which allows us to ensure geometrical validation test later on, and an alignment stage in which the observations are turned into landmark Cartesian coordinates. Next, a data association stage is required to map the observations to the already identified landmark or initiate new landmarks. For this purpose, one uses Mahanalobis distance to match the observation to possible landmarks. If none of the landmarks matches the current observation and both the geometrical and statistical tests were positive, then a new landmark is initiated and the state vector is therefore augmented. Note that, in order to balance the cost effectiveness and optimality requirement, the reasoning is carried out only within a submap of the environment, where it is most likely to find the matching given the sensor limitations. At later stage, the obtained landmarks are also used to extract feature landmark consisting of segments and corners. The methodology has been validated in a platform using two Khepera robots, one of which is equipped with vision turret while both are equipped with range infrared sensors and encoders. A virtual interface showing the robot trajectory as well as environment is developed using OpenGL platform for 3D visualization. The use of both robots also allowed us to test and validate collaboration scenarios in which multi-robot localization technique is used in conjunction with SLAM algorithm. The tests carried out demonstrated the validation of the developed algorithm and the consistency of the outcomes when looking at the 95% confidence bound limits.
This open a new area of research where more advanced collaboration scenarios can be used in more complex environments where the features can be constituted of either geometrical or non-geometrical features. On the other hand, inspired by the overall intelligent behaviour of large biological insect communities, together with the rapid development of the field of distributed artificial intelligence, through, for instance, the concrete RoboCup robot soccer initiative, this offers new motivation grounds for further developments of multiple robot systems at different research communities.

6. Acknowledgment.

This work in sponsored in part by a grant from Nuffield Foundation.

7. References

J. Andrade-Cetto and A. Sanfeliu (2004). The Effect of partial observability in SLAM, in *Proceedings of IEEE International Conference on Robotics and Automation*, p. 397-402

T. Bailey and H. Durrant-White (2006). Simultaneous localization and Mapping (SLAM): Part II. State of the Art, *IEEE Robotics and Automation Magazine* 13(3), 108-117

M. Begum, G.K.I, Mann and R.G. Gosine (2005), Concurrent mapping and localization for mobile robot using soft computing techniques, in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, p. 2266-2271

V. Bonato, M.M. Fernández and E. Marques (2006). A smart camera with gesture recognition and SLAM capabilities for mobile robots, *International Journal of Electronics*, 93(6), 385-401

K. Chong and L. Kleeman (1999). Mobile robot map building from an advanced sonar array and accurate odometry. *International Journal of Robotics Research*, 18(1):20–36.

M. Deans and M. Hebert (2000). Experimental comparison of techniques for localization and mapping using a bearing-only sensor. In: *Proceedings of the International Symposium on Experimental Robotics*.

M.W.M.G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, and M. Csorba (2000). An experimental and theoretical investigation into simultaneous localisation and map building. *Experimental Robotics IV*, p. 265–274.

M.W.M.G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, and M. Csorba (2001). A solution to the simultaneous and mapping (SLAM) problem, *IEEE Transactions On Robotics and Automations*, 17(3), 229-241

M.W.M.G. Dissanayake, H.F. Durrant-Whyte, and T. Bailey (2000). A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 1009–1014.

D. Dufourd and R. Chatila (2004), Combinatorial maps for Simultaneous Localization and Map building (SLAM), In: *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p.1047-1052

H.F. Durrant-White (1988), Uncertain geometry in robotics, *IEEE Transactions on Robotics and Automation*, 4(1), 23-31.

H. Durrant White and T. Bailey, Simultaneous localization and Mapping: Part I (2006), *IEEE Robotics and Automation Magazine*, 13(2), 99-108

A.I. Eliazar and R. Parr, DP-SLAM 2.0 (2004), in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, p.1324-1320

A. Gelb (1996). *Applied Optimal Estimation*. MIT Press, 14th edition.

J. Guivant, E.M. Nebo (2001) Optimization of the simultaneous localization and map building algorithm for real time implementation, *IEEE Transaction on Robotics and Automation*, 17(3), 242-257.

T. Higuchi (1996). Monte Carlo filter using the genetic algorithm operators. *Journal of Statistical Computation and Simulation*, 59(1):1--23.

Y.L. IP and A.B. Rad (2004), Incorporation of Feature Tracking into Simultaneous Localization and Map Building via Sonar Data, *Journal of Intelligent and Robotic Systems*, 39(2), 149-172

S. J. Julier (2003), The stability of covariance inflation methods for SLAM, in: *Proceedings of the IEEE/RSJ International Conference on Robots and Systems*, 2749-2754

S. J. Julier and J. K. Uhlmann (2001), Simultaneous Localization and Map Building Using Split Covariance Intersection, In: *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Muai, Hawaii, USA, p.1257-1262

H. Kitaneo (1997), *RoboCup-97: Robot Soccer World Cup I*, *Lecturer Notes in Computer Sciences / Lecturer Notes in Artificial Intelligence* Vol. 1375, Springer-Verlag, Berlin, NY.

J. Knight, A. Davison and I. Reid (2001), Towards Constant Time SLAM using Postponement, in: *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Muai, Hawaii, USA, 406-412

N. M. Kwok and G. Dissanayake (2004). An Efficient Multiple Hypothesis Filter for Bearing-Only SLAM, in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, p.736-741

T. Lemaire, S. Lacroix and J. Sola (2005), A practical 3D Bearing-Only SLAM algorithm, in: *Proceedings of Intelligent Robotics and Systems, IEEE/RSJ International Conference*, p.2449-2454.

J. J. Leonard and H.F. Durrant-Whyte (1991). Simultaneous map building and localization for an autonomous mobile robot. In: *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol. 3, p. 1442–1447.

J. J. Leonard, R.J. Rikoski, P.M. Newman, and M.C. Bosse (2002). Mapping partially observable features from multiple uncertain vantage points. *International Journal of Robotics Research*, 21(10–11):943–975.

B. Lisien, D. Morales, D. Silver, G. Kantor, I. Rekleitis, and H. Choset (2003). Hierarchical simultaneous localization and mapping. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, p 448--453.

A. Martinelli, N. Tomatis and R. Sieqward (2003), Open Challenges in SLAM : an open solution based on shift and rotation invariants, in: *Proceedings of IEEE International Conference on Robotics and Automation*, p.1327-1332

A. Martinelli, F. Pont and R. Siegwart (2005). Mult-irobot localization using relative observations, in: *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 2808-2813

M. Matarik (1998), Coordination and learning in multi-robot systems, *IEEE Intelligent Systems*, 13(2), 6-8

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0 (2003): An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: *Proceedings of the IJCAI-03*, Morgan Kaufmann,. 1151–1156.

M. Montemerlo and S. Thrun (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 1985–1991.

R. Murphy (2000), *An introduction to AI Robotics*, MIT Press

J. Nieto, J. Guivant, and E. Nebot (2004). The hybrid metric maps (HYMMs): A novel map representation for DenseSLAM. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, p. 391–396.

J. Nieto, T. Bailey, and E. Nebot. Scan-SLAM (2005). Combining EKF SLAM and scan correlation. In: *Proceedings of the International Conference on Field and Service Robotics.*

M. Oussalah, H. Maaref and C. Barret (2003), Application of a possibilistic-based approach to mobile robotics, *Journal of Intelligent and Robotics Systems*, 38(2), 175-195.

R. Smith and P. Cheeseman (1986). On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68.

R. Smith, M. Self, and P. Cheeseman (1987). Estimating uncertain spatial relationship in robotics, in: Autonomous Robot Vehicles, I.J Cox and G.T. Wilfon (Editor), *A stochastic map for uncertain spatial relationships. Autonomous Mobile Robots: Perception, Mapping and Navigation*, 323–330.

J. Sola, A. Monin, M. Devy and T. Lemaire (2005), Undelayed Initialization in Bearing Only SLAM, in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, 2499-2504

J. Todd and D. Pomerleau (1996). Life in the fast lane: The evolution of an adaptive vehicle control system. *Artificial Intelligence Magzine* 17 (2) 11-50.

S. Thrun, D. Fox and W. Burgard (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning* 31:29–53. Also appeared in: *Autonomous Robots* 5:253–271

W. S. Wijesoma , *L. D. L.* Perera and M.D. Adams (2006). Towards Multidimensional Assignment Data Association in Robot Localization and Mapping, *IEEE*

Transactions on Robotics, 22(2), 350-365.

S.B. Wiliams (2001), *Efficient Solutions to Autonomous Mapping and Navigation Problems*, PhD thesis, University of Sidney.

Edited by Sascha Kolski

Today robots navigate autonomously in office environments as well as outdoors. They show their ability to beside mechanical and electronic barriers in building mobile platforms, perceiving the environment and deciding on how to act in a given situation are crucial problems. In this book we focused on these two areas of mobile robotics, Perception and Navigation.

ISBN 978-953-51-5804-2

Photo by dianaarturovna / iStock

IntechOpen