



IntechOpen

Mobile Robotics, Moving Intelligence

Edited by Jonas Buchli



Mobile Robots

Moving Intelligence

Edited by
Jonas Buchli

Mobile Robotics, Moving Intelligence

<http://dx.doi.org/10.5772/34>

Edited by Jonas Buchli

© The Editor(s) and the Author(s) 2006

The moral rights of the and the author(s) have been asserted.

All rights to the book as a whole are reserved by INTECH. The book as a whole (compilation) cannot be reproduced, distributed or used for commercial or non-commercial purposes without INTECH's written permission.

Enquiries concerning the use of the book should be directed to INTECH rights and permissions department (permissions@intechopen.com).

Violations are liable to prosecution under the governing Copyright Law.



Individual chapters of this publication are distributed under the terms of the Creative Commons Attribution 3.0 Unported License which permits commercial use, distribution and reproduction of the individual chapters, provided the original author(s) and source publication are appropriately acknowledged. If so indicated, certain images may not be included under the Creative Commons license. In such cases users will need to obtain permission from the license holder to reproduce the material. More details and guidelines concerning content reuse and adaptation can be found at <http://www.intechopen.com/copyright-policy.html>.

Notice

Statements and opinions expressed in the chapters are those 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, 2006 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 Robotics, Moving Intelligence

Edited by Jonas Buchli

p. cm.

Print ISBN 3-86611-284-X

eBook (PDF) ISBN 978-953-51-5802-8

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,400+

Open access books available

118,000+

International authors and editors

130M+

Downloads

151

Countries delivered to

Our authors are among the
Top 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Preface

It has been an age-old dream to build versatile machines which should assist humans for work which is dangerous or boring and dull. Today, some machines perform very well in tasks which require more strength, speed, endurance or precision than a human can possibly achieve. In this direction industrial robotics has achieved a lot. It is an established engineering field and a successful industry. These industrial robots work in special environments, carefully planned and modelled. Simply put, the environment is tailored to the need of the robot and not the robot to the need of the environment. Furthermore, their tasks are very specific; they are not flexible and have no autonomy in terms of decisions. In short they do not possess any form of what is commonly called intelligence.

But now, robots are about to step out of their carefully engineered environments to discover and share our daily world with us. They will experience an unstructured environment of which they need to make sense and plan accordingly. We would like these robots to be versatile, useful for many tasks without the need for reprogramming or even re-designing them. Ideally they should integrate seamlessly with humans in the same environment or in other places which are not specially designed a priori for robots, such as accident sites for example. This makes the modelling of the environment and the robots interaction with it much more challenging or even impossible in many cases. Difficult challenges arise in many aspects; the robots need to guarantee safety for themselves, humans and other things in their surroundings. They need to deal with, possibly moving, obstacles, have to navigate their way to goals, plan and fulfil their tasks. And last but not least they have to deal with the limited amount of energy they can carry with them, either by being energy efficient or by being able to recharge in time.

Thus, with the requirement to move about unstructured, unknown complex environments, comes the need to be more "intelligent", i.e. a larger capability of integrating information from different sources, need for longer term memory and longer term planning, more extended mapping and more performant navigation. A crucial element is flexibility and adaptation in the face of new unseen challenges and tasks. The fundamental nature of those problems is reflected in the guiding theme of the book "Mobile Robots, Moving Intelligence".

Also the senses and ways of interacting with the environment of the robots have to become more versatile. Humans, with their hands, have an amazing multi-function manipulator which is capable of performing tasks requiring strong force and can be used with amazing precision and subtlety on other tasks. Living beings have a massive numbers of sensors, which operate at low energy and they can successfully deal with and integrate these high dimensional data. Current robots have none of both, only a few very specific sensors and manipulators and very limited capability to deal with high dimensional data.

It is very interesting, that one of the most basic capabilities of many living beings, namely roam their environment, is one of the least satisfactorily solved in robotics. And it is only this capability which makes a robot truly mobile. Locomotion for robots is not only a prob-

lem of putting the right nuts and bolts together with some electronics, but there are hard and rather fundamental problems yet to be solved from the level of actuators, over mechanical issues, up to the level of control.

Many of these aspects can be found in the latest research robots and are addressed in one way or the other in the present book. So at this time when the expectation towards a mobile robot grows larger this book comes timely and collects accounts of research in many of the aforementioned directions.

It covers many aspects of the exciting research in mobile robotics. It deals with different aspects of the control problem, especially also under uncertainty and faults. Mechanical design issues are discussed along with new sensor and actuator concepts. Games like soccer are a good example which comprise many of the aforementioned challenges in a single comprehensive and in the same time entertaining framework. Thus, the book comprises contributions dealing with aspects of the Robotcup competition.

The reader will get a feel how the problems cover virtually all engineering disciplines ranging from theoretical research to very application specific work. In addition interesting problems for physics and mathematics arises out of such research.

We hope this book will be an inspiring source of knowledge and ideas, stimulating further research in this exciting field. The promises and possible benefits of such efforts are manifold, they range from new transportation systems, intelligent cars to flexible assistants in factories and construction sites, over service robot which assist and support us in daily live, all the way to the possibility for efficient help for impaired and advances in prosthetics.

Editor
Jonas Buchli

Contents

Control

1. **Supervisory Control for Turnover Prevention of a Teleoperated Mobile Agent with a Terrain-prediction Sensor Module** 001
Jae Byung Park and Beom Hee Lee
2. **Dynamics and Control for Nonholonomic Mobile Modular Manipulators** 029
Yangmin Li and Yugang Liu
3. **Combined Torque and Velocity Control of a Redundant Robot System** 053
Damir Omrcen, Leon Zlajpah and Bojan Nemeč
4. **EMOBOT: A Robot Control Architecture Based on Emotion-Like Internal Values** 075
Nils Goerke
5. **Mobile Robotics, Moving Intelligence** 095
Souma Alhaj Ali, Masoud Ghaffari, Xiaoqun Liao and Ernest Hall
6. **Decentralized Robust Tracking Control for Uncertain Robots** 117
Zongying Shi, Yisheng Zhong and Wenli Xu
7. **Gauging Intelligence of Mobile Robots** 135
Ka C Cheok
8. **A Reusable UART IP Design and It's Application in Mobile Robots** 147
Ching-Chang Wong and Yu-Han Lin
9. **Intelligent Pose Control of Mobile Robots Using an Uncalibrated Eye-in-Hand Vision System** 159
T. I. James Tsay and Y. F. Lai
10. **A Fuzzy Logic Controller for Autonomous Wheeled Vehicles** 175
Mohamed B. Trabia, Linda Z. Shi and Neil E. Hodge

-
11. **Real-Time Optimization Approach for Mobile Robot** 201
Hiroki Takeuchi
 12. **Design and Control of an Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels** 223
Jae-Bok Song and Kyung-Seok Byun
 13. **Dynamic Model, Control and Simulation of Cooperative Robots: A Case Study** 241
Jorge Gudino-Lau and Marco A. Arteaga
 14. **Biologically-plausible reactive control of mobile robots** 271
Zapata Rene and Lepinay Pascal
 15. **Transputer Neuro-Fuzzy Controlled Behaviour-Based Mobile Robotics System** 287
E. Al-Gallaf
 16. **Mechanism and Control of Anthropomorphic Biped Robots**..... 307
Hun-ok Lim and Atsuo Takanishi
 17. **Bio-mimetic Finger: Human like morphology, control and motion planning for intelligent robot and prosthesis** 325
Emanuele Lindo Secco and Giovanni Magenes
 18. **Vision Based Control of Model Helicopters** 349
Erdinc Altug, James P. Ostrowski and Camillo J. Taylor

MultiRobot Systems

19. **Multi –Agent System Concepts; Theory and application phases** 369
Adel Al-Jumaily and Mohamed Al-Jaafreh
20. **Grid technologies for intelligent autonomous robot swarms** 393
Fabio P. Bonsignorio
21. **Acromovi architecture: A framework for the development of multirobot applications** 409
Patricio Nebot and Enric Cervera
22. **Multi-Robot Systems and Distributed Intelligence: the ETHNOS approach to Heterogeneity** 423
Antonio Sgorbissa

Multi-legged robots

- 23. Force Sensing for Multi-legged Walking Robots: Theory and Experiments**
Part 1: Overview and Force Sensing 447
A. Schneider and U. Schmucker
- 24. Force Sensing for Multi-legged Walking Robots: Theory and Experiments**
Part 2: Force Control of Legged Vehicles 471
A. Schneider and U. Schmucker
- 25. Force Sensors in Hexapod Locomotion** 495
Sathya Kaliyamoorthy, Sasha N. Zill and Roger D. Quinn
- 26. Novel Robotic Applications using Adaptable Compliant Actuation. An Implementation Towards Reduction of Energy Consumption for Legged Robots** 513
Bjorn Verrelst, Bram Vanderborght,
Ronald Van Ham, Pieter Beyl and Dirk Lefeber
- 27. Acquisition of Obstacle Avoidance Actions with Free-Gait for Quadruped Robots** 535
Tomohiro Yamaguchi, Keigo Watanabe and Kiyotaka Izumi
- 28. Fault-Tolerant Gait Planning of Multi-Legged Robots** 557
Jung-Min Yang, Yong-Kuk Park and Jin-Gon Kim

Supervisory Control for Turnover Prevention of a Teleoperated Mobile Agent with a Terrain-Prediction Sensor Module

Jae Byung Park & Beom Hee Lee
Seoul National University
Seoul, Korea

1. Introduction

Teleoperated mobile agents (or vehicles) play an important role especially in hazardous environments such as inspecting underwater structures (Lin, 1997), demining (Smith, 1992), and cleaning nuclear plants (Kim, 2002). A teleoperated agent is, in principle, maneuvered by an operator at a remote site, but should be able to react autonomously to avoid dangerous situations such as collisions with obstacles and turnovers. Many studies have been conducted on collision avoidance of mobile agents (Borenstein, 1989; Borenstein, 1991a; Borenstein, 1991b; Howard, 2001; Niwa, 2004; Singh et al., 2000). In this research, however, we will focus on turnover prevention of mobile agents moving on uneven terrain because a turnover can cause more fatal damage to the agents. Here, we adopt the term 'turnover' as a concept which includes not only a rollover but also a pitchover.

Extensive studies have been conducted on motion planning problems of mobile agents traveling over sloped terrain in the robotics research community (Shiller, 1991). Shiller presented optimal motion planning for an autonomous car-like vehicle without a slip and a rollover. The terrain was represented by a *B*-spline patch and the vehicle path was represented by a *B*-spline curve, where the terrain and vehicle path were given in advance. With the models of the terrain and the path, the translational velocity limit of the vehicle was determined to avoid a slip and a rollover. Also, many studies have been conducted on rollover prevention of heavy vehicles like trucks and sports utility vehicles in the vehicular research community. Takano analyzed various dynamic outputs of large vehicles, such as the lateral acceleration, yaw rate, roll angle, and roll rate, in the frequency domain for predicting rollovers (Takano, 2001). Chen developed the time-to-rollover (TTR)-based rollover threat index in order to predict rollovers of sports utility vehicles (Chen, 1999). This intuitive measure TTR was computed from the simple model and then corrected by using an artificial neural network. Nalecz et al. suggested an energy-based function called the rollover prevention energy reserve (RPER) (Nalecz, 1987; Nalecz, 1991; Nalecz, 1993). RPER is the difference between the energy needed to bring the vehicle to its rollover position and the rotational kinetic energy, which can be transferred into the gravitational potential energy to lift the vehicle. RPER is positive for

non-rollover cases and negative for rollover cases. Acarman analyzed the rollover of commercial vehicles with tanks that are partially filled with liquid cargo (Acarman, 2003). In this case, the frequency shaped backstepping sliding mode control algorithm was designed to stabilize and attenuate the sloshing effects of the moving cargo by properly choosing the crossover frequencies of the dynamic compensators in accordance with the fundamental frequencies of the slosh dynamics.

Many studies have been conducted on turnover prevention of mobile manipulators like a fork lift. Rey described the scheme for automatic turnover prediction and prevention for a forklift (Rey, 1997). By monitoring the static and dynamic turnover stability margins of a mobile manipulator, it is possible to predict turnovers and take appropriate actions to prevent turnovers. Here, the dynamic force-angle measure of turnover stability margin proposed by Papadopoulos (Papadopoulos, 1996) is employed. Also, Sugano suggested the concepts about stability such as the stability degree and the valid stable region based on the zero-moment point (ZMP) criterion to evaluate the stability for a mobile manipulator (Sugano, 1993). In addition, the method of ZMP path planning with a stability potential field was suggested for recovering and maintaining stability (Huang, 1994). Based on the path planning method, the motion of the manipulator is planned in advance to ensure stability while the vehicle is in motion along a given trajectory. Furthermore, for stability recovery, the compensation motion of the manipulator is derived by using the redundancy of the manipulator, taking into consideration the manipulator configuration and the static system stability (Huang, 1997).

In the abovementioned researches for an autonomous mobile agent, the path and trajectory of a vehicle and a manipulator were given in advance and modified for rollover prevention. However, the path and trajectory of a teleoperated mobile agent cannot be given in advance since both of them are determined by a teleoperator at each time instant. Thus, it is impossible to analyze and prevent rollovers in advance. For a fork lift mentioned above, its path and trajectory were not known in advance since it was maneuvered by an operator. Thus, the previous researchers estimated the path and trajectory using the proprioceptive sensor data (internal sensor data) for turnover prevention. However, in the case where there is a potential risk of turnovers due to an abrupt change in the configuration of the ground, the proprioceptive sensor data is not enough to prevent turnovers. Therefore, in this research, a low-cost terrain-prediction sensor with a camera vision and a structured laser light is proposed for predicting turnovers at front terrain before the agent arrives there. With these predicted data, a turnover prevention algorithm is suggested with the quasi-static rollover analysis of a rigid vehicle (Gillespie, 1992).

A proposed turnover prevention algorithm (Park, 2006a) consists of a pitchover prevention algorithm and a rollover prevention algorithm (Park, 2006b). According to the turnover prevention algorithm, the translational and rotational velocities of the agent are restricted for avoiding turnovers. However, the turnover prevention control brings about some inconsistencies between the intended motion and the reactive motion of the agent. For compensating these inconsistencies, we propose a force reflection technique based on virtual reality. A force reflection technique has already been used in various research areas such as medical surgery (Chen, 1998; Basdogan, 2004; Nudehi, 2005), micromanipulation (Ando, 2001; Boukhnifer, 2004), and obstacle avoidance of teleoperated mobile agents (Park, 2003a; Park, 2003b; Park, 2004; Park, 2006b). In this research, a reflective force helps an operator

control the agent without a turnover, where a 2-DOF force-feedback joystick is used as a Haptic device which can not only receive an operator's command from an operator but also send back a reflective force to him.

2. Teleoperation System

2.1 Supervisory Control

In a teleoperation system, an operator, in principle, controls a mobile agent at a remote site using a force feedback joystick, but the agent needs to control itself autonomously for escaping dangerous situations like overturning. As a result of autonomous control, the reactive motion of the agent may be different from the intended motion of an operator. It is to violate the principle rule of a teleoperation system as mentioned above. So we analyze boundaries of safe motion of an agent without turnovers and allow an operator to freely control the agent within the analyzed safe boundaries. That is, the agent motion determined by an operator is restricted for turnover prevention only when the agent motion is beyond the safe boundaries. Thus, the resultant motion of the agent is determined by the closest motion to the intended motion of an operator among the turnover-free motions. In addition, we propose a force feedback technique for an operator to recognize the inconsistency between the reactive and intended motions of an agent. If the agent controlled by an operator is faced with danger, the operator feels reflective force generated by the force feedback joystick for preventing the operator from controlling the agent beyond the safe boundaries. Thus, reflective force makes it possible that the operator drives the agent without turnovers.

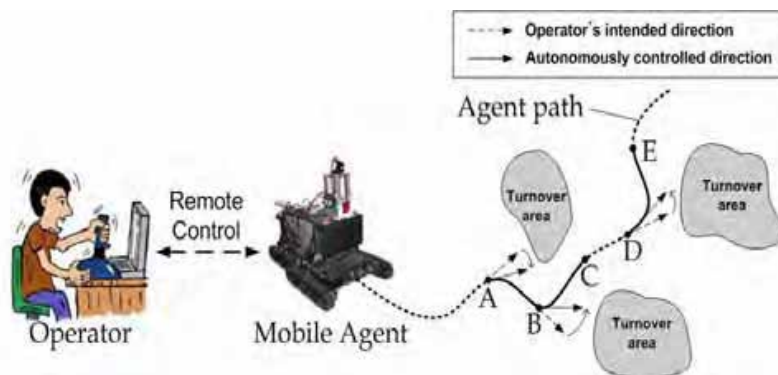


Fig. 1. Supervisory control of a teleoperation system with a mobile agent. (A-C, D-E: Autonomously controlled path segment for turnover prevention).

An example of the supervisory control is shown in Fig. 1. An agent moves to A according to an operator's command. From A to C, the agent is autonomously controlled for avoiding turnovers since it detects a potential turnover area. From C, the agent is controlled by an operator since the agent escapes from danger of turnovers. Again, the agent is autonomously controlled for turnover prevention from D to E. As described above, the operator's intended direction of the agent is modified by the autonomously controlled

direction for turnover prevention in the case that the agent detects potential turnovers. Also, whenever the agent is autonomously controlled, the operator feels reflective force and is able to recognize the modified agent motion. However, in the case that there is no danger of turnovers, the agent is controlled by an operator.

2.2 System Configuration

The teleoperation system consists of a remote control system (RCS) and a mobile agent system (MAS) as shown in Fig. 2. The RCS and the MAS communicate with each other via wireless Ethernet communication. Control signals and sensor data are denoted in Table 1. The RCS receives input force $F_o(t)$ from an operator via a force feedback joystick, and the joystick position $P_j(t)$ is determined by $F_o(t)$. Then, velocity command $V_{cmd}(t)$ of the agent is determined from $P_j(t)$ by a position-to-velocity matcher, where $V_{cmd}(t)$ consists of the translational velocity $v(t)$ and rotational velocity $\omega(t)$. Here, each velocity can be controlled independently since the agent used in this research is a differential-drive machine which has two individually motorized tracks. The operator's command $V_{cmd}(t)$ is restricted by a turnover prevention controller for avoiding potential turnovers using predicted terrain data $Tr(t)$ transmitted from the MAS. Finally, the resultant velocity command $V_d(t)$ for turnover prevention is transmitted to the MAS for actually controlling the agent without turnovers. Also, reflective force $F_R(t)$ is generated by $P_{ub}(t)$ to $P_{lb}(t)$, where $P_{ub}(t)$ and $P_{lb}(t)$ are determined by upper and lower bounds, $V_{ub}(t)$ and $V_{lb}(t)$, for turnover-free ranges of $v(t)$ and $\omega(t)$, respectively. As a result of force reflection, an operator can intuitively recognize whether the agent motion is restricted for turnover prevention or not.

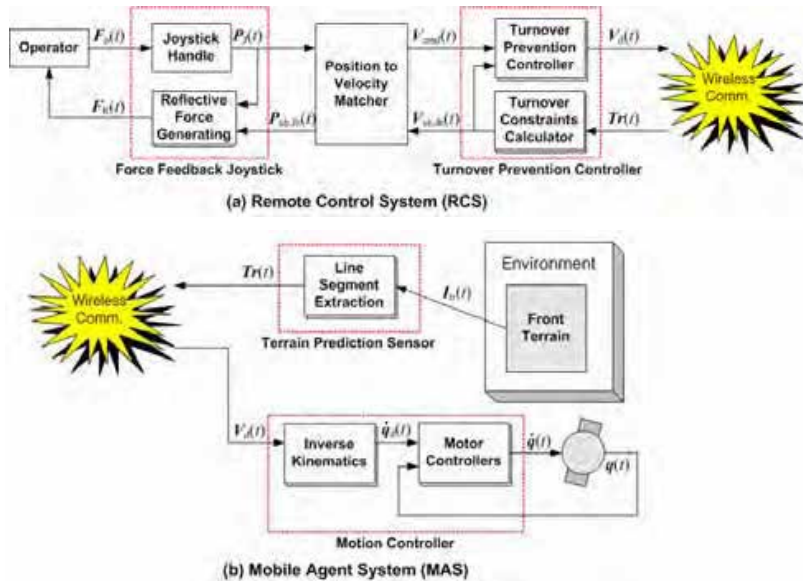


Fig. 2. Teleoperation system which is comprised of the RCS and the MAS.

Symbols	Descriptions
$F_O(t)$	Input force by an operator's command
$F_R(t)$	Reflective force generated by the force feedback joystick
$P_J(t)$	Joystick position determined by $F_O(t)$
$V_{ub}(t), V_{lb}(t)$	Upper and lower bounds of the translational and rotational velocities for avoiding turnovers
$P_{ub}(t), P_{lb}(t)$	Joystick positions determined by $V_{ub}(t)$ and $V_{lb}(t)$
$V_{cmd}(t)$	Control command of the agent determined by $P_J(t)$
$V_d(t)$	Desired velocities for turnover prevention determined by $V_{cmd}(t)$, $V_{ub}(t)$ and $V_{lb}(t)$
$I_{tr}(t)$	Terrain image data obtained by the camera vision with the structured laser light
$Tr(t)$	Terrain data obtained after image processing of $I_{tr}(t)$
$\dot{q}_d(t)$	Desired spinning speeds of the actual motors
$\dot{q}(t)$	Actual spinning speeds of the motors
$q(t)$	Encoder data of the motors

Table 1. Symbols of control signals and sensor data.

The MAS is composed of a mobile agent and a proposed terrain-prediction sensor module. The ROBHAZ-DT which is developed by the Korea Institute of Science and Technology (KIST) and Yujin Robotics Co., Ltd. is employed as an actual mobile agent. KIST and Yujin Robotics Co., Ltd. are developing various ROBHAZ-series with high mobility for conducting dangerous tasks such as rescue mission, explosive ordnance disposal (EOD), mine exclusion and scout. The ROBHAZ-DT3 conducted military missions such as reconnaissance and explosive detection with Korean troops in Iraq for six months in 2004. Also, the improved model Roscuc of the ROBHAZ took first, second and third prizes for rescue robots at the RoboCup 2004 in USA, the RoboCup 2005 in Japan and the RoboCup 2006 in Germany, respectively. For more information about the ROBHAZ-series, you can found at <http://www.robhaz.com>. The ROBHAZ-DT used in this research is an early prototype with double tracks as shown in Fig. 3. The MAS computes the desired spinning speeds $\dot{q}_d(t)$ of two drive wheels for the desired velocity command $V_d(t)$ received from the RCS, and sends them to the embedded controllers that control the actual motors to achieve the desired spinning speeds through internal feedback control loops with encoder data $q(t)$. Next, front terrain data $Tr(t)$ for turnover prevention are obtained by the terrain-prediction sensor module, which projects a structured laser light on the front terrain and detects the projected line using a web camera. In this case, the laser-line segment is extracted from terrain image data $I_{tr}(t)$ on the camera image plane for obtaining $Tr(t)$. Finally, the obtained terrain data $Tr(t)$ is transmitted to the RCS for turnover prevention.

Let time T_s be the communication period between the RCS and the MAS. Then time T_s should satisfy

$$T_s > T_a + 2T_d \quad (1)$$

where T_a is the maximum delay for sensor acquisition and T_d is the maximum delay for wireless communication. If the accessible range of IEEE 802.11b based Wireless Local Area Networks (WLANs) used in our system covers the locations of both RCS and MAS, the

round-trip time delay $2T_d$ (less than 0.29 ms) can be neglected as compared with time T_a (less than 100 ms). As the sum of communication packet sizes of both control signals and sensor data is less than 200 bytes (or 1600 bits) and the IEEE 802.11b standard promises data rates up to 11 Mbps, the time delay $2T_d$ can be bounded in 0.29 ms. Although the coverage of the WLANs is reduced in crowded areas, the coverage can be easily expanded by establishing additive wireless Access Points (APs) in the areas. Also, the motion control of the agent can be completed in time T_s since the motion control time is much less than T_a and the motion control is conducted simultaneously with sensor acquisition. Hereafter, the translational velocity $v(t)$ and the rotational velocity $\omega(t)$ will be discretely described as $v(k)$ and $\omega(k)$, based on time index k which denotes time $t = kT_s$. Of course, control signals and sensor data will also be described with k instead of t .



Fig. 3. ROBHAZ-DT (Robot for Hazardous Application-Double Track).

2.3 Basic Assumptions

Basic assumptions are introduced for terrain prediction and turnover prevention control as follows:

1. The communication period T_s between the RCS and the MAS ensures enough time to complete the terrain data acquisition and the motion control of the agent, taking into consideration the maximum time delay for wireless communication.
2. No turnover occurs between the starting position of the agent and the first detected terrain position by the terrain-prediction sensor since the agent is impossible to avoid turnovers without terrain sensor data.
3. The process for terrain data acquisition is fast enough to obtain sufficient terrain data for turnover prevention control at each time instant. At least much more than two are available within the longitudinal length of the agent while the agent moves at its normal speed.
4. The agent is represented as one lumped mass located at its center of gravity (CG) with appropriate mass and inertia properties since all components of the agent move together. The point mass at the CG, with appropriate rotational moments of inertia, is dynamically equivalent to the agent itself for all motions in which it is reasonable to assume the agent to be rigid (Gillespie, 1992).
5. The agent has a trapezoidal velocity profile. That is, the translational acceleration of the agent is determined by constant values such as a_c for accelerated motion, 0 for uniform motion and $-a_c$ for decelerated motion.

6. The motion controllers of the agent control the translational acceleration with tolerable errors according to the reference inputs such as a_c , 0 and $-a_c$. Therefore, we do not consider the variation of the acceleration depending on the various terrain types such as rocky and sandy terrain.
7. The agent is able to reduce its translational velocity from v_{max} to 0 for a distance of D_{tr} , where D_{tr} is the reference distance to the front terrain detected for turnover prevention control at each time instant. In other words, D_{tr} is defined to satisfy the condition $D_{tr} > v_{max}^2 / 2a_c$. Thus, taking the condition for D_{tr} into consideration, the configuration of the terrain-prediction sensor module such as the orientations of the camera and the laser-line generator should be determined. According to this assumption, even though the agent detects inevitable turnover terrain at a distance of D_{tr} , it can reduce the translational velocity and stop before arriving at the detected terrain.

3. Front Terrain Prediction

3.1 Terrain-prediction Sensor Module

We develop a low-cost terrain-prediction sensor module for obtaining front terrain data in advance. As shown in Fig. 4, the developed terrain-prediction sensor module consists of a web camera, a laser-line generator and an inclinometer, and is attached to the ROBHAZ-DT. The laser-line generator LM-6535ML6D developed by Lanics Co., Ltd. is used to project a line segment on front terrain. The fan angle and line width of the laser-line generator are 60° and 1 mm, respectively. The wavelength of the laser beam ranges from 645 nm to 665 nm and the optical output power is 25 mW. The complementary metal-oxide-semiconductor (CMOS) web camera ZECA-MV402 developed by Mtekvision Co., Ltd. is used to detect the line segment projected onto the front terrain. The inclinometer 3DM developed by MicroStrain Inc. is used to measure the absolute angles from 0° to 360° on both yaw and pitch axes, and from -70° to 70° on the roll axis with respect to the universal frame. The data of the inclinometer are obtained via RS232 Serial interface.

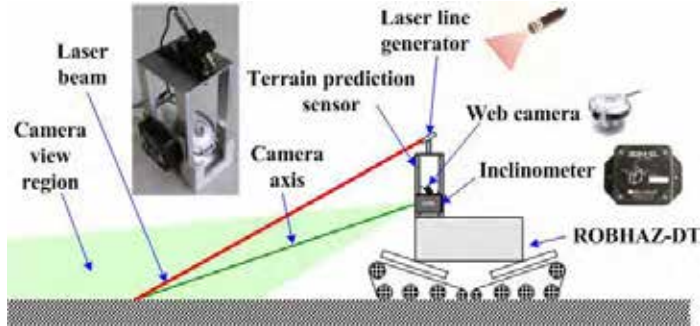


Fig. 4. Low-cost terrain prediction sensor module attached to the ROBHAZ-DT.

3.2 Acquisition of Vision Data

For terrain data acquisition, we first propose an image processing method for extracting a projected laser line from an original camera image where the image size is 320×240 pixels.

The partitioning an image into regions such as an object and the background is called segmentation (Jain, 1995). A binary image for an object and the background is obtained using an appropriate segmentation of a gray scale image. If the intensity values of an object are in an interval and the intensity values of the background pixels are outside this interval, a binary image can be obtained using a thresholding operation that sets the points in that interval to 1 and points outside that interval to 0. Thus, for binary vision, segmentation and thresholding are synonymous.

Thresholding is a method to convert a gray scale image into a binary image so that objects of interest are separated from the background. For thresholding to be effective in object-background separation, it is necessary that the objects and background have sufficient contrast and that we know the intensity levels of either the objects or the background. In a fixed thresholding scheme, these intensity characteristics determined the value of the threshold. In this research, a laser line is an object to be separated from the background. Since a laser line is lighter than the background, an original image $F(u_1, u_2)$ for $u_1=1, \dots, 320$ and $u_2=1, \dots, 240$ can be partitioned into the laser line and the background using a thresholding operation as follows:

$$F_T(u_1, u_2) = \begin{cases} 1 & \text{if } F(u_1, u_2) \geq T \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $F_T(u_1, u_2)$ is the resulting binary image and T is a threshold. By (2), $F_T(u_1, u_2)$ has 1 for the laser line and 0 for the background. The results of producing an image using different thresholds are shown in Fig. 5. Fig. 5 (b) shows the resulting image with $T=150$. The left and right sides of the projected laser line is not separated from the background since the intensity values of both sides of the line are outside the interval. For detecting both sides of the line, an image is obtained using $T=120$ as shown in Fig. 5 (c). As compared with $T=150$, more parts of the line are detected. Finally, the binary image with $T=100$ is shown in Fig. 5 (d). Although the resulting image includes more parts of the line as compared with $T=150$ and $T=120$, some parts of the background pixels are wrong detected as the line since the intensity of some background pixels are in the interval. As shown in these examples, the threshold of the fixed threshold method should be appropriately determined according to the application domain. In other words, we have to change the threshold whenever the domain is changed. Also, the threshold needs to be changed for an illumination change.

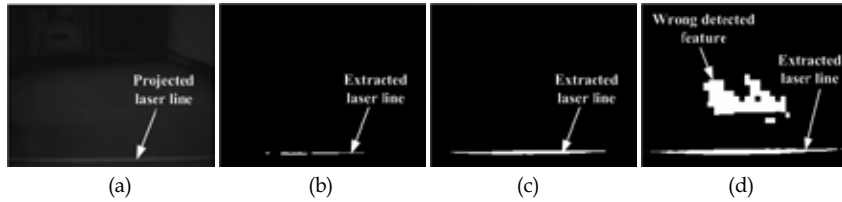


Fig. 5. Laser-line detection using a fixed threshold scheme: (a) original image, and binary images thresholded with (b) $T=150$, (c) $T=120$ and (d) $T=100$.

In this research, we propose an adaptive vertical threshold scheme in order to separate the laser line from the background regardless of an illumination change. The concept of the proposed

threshold scheme is shown in Fig. 6. Although the intensity of both sides of the line is weaker than the intensity of the center of the line, the artificial laser light is lighter than other pixels on its vertical line. Using this fact, we define a threshold for the u th vertical line as follows:

$$T_v(u) = \max_{u_2} F(u, u_2) \quad (3)$$

where $u=1, \dots, 320$. As shown in (3), the vertical threshold $T_v(u)$ is adaptively determined as the maximum intensity of the pixels on the u th vertical line even though the intensity of illumination is changed. Using $T_v(u)$, each vertical line is thresholded as follows:

$$F_{VT}(u, u_2) = \begin{cases} 1 & \text{if } F(u, u_2) \geq T_v(u) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Finally, the resulting binary image is obtained by the union of $F_{VT}(u, u_2)$ for u as follows:

$$F_{VT}(u_1, u_2) = \bigcup_{u=1}^{320} F_{VT}(u, u_2) \quad (5)$$

That is, the detected laser line is the region for $F_{VT}(u_1, u_2)=1$. The results of producing an image using the adaptive vertical threshold scheme are shown in Fig. 7. For the low intensity of illumination, the projected laser line is shown in Fig. 7 (a). In this case, the entire laser line is obtained as shown in Fig. 7 (b). For the high intensity of illumination, it is hard to distinguish the laser line from the background as shown in Fig. 7 (c). However, the entire line is also obtained by the proposed vertical threshold scheme as shown in Fig. 7 (d). That is, the adaptive vertical threshold scheme is not sensitive to an illumination change. Thus, the vertical threshold scheme can be directly applied to various application domains.

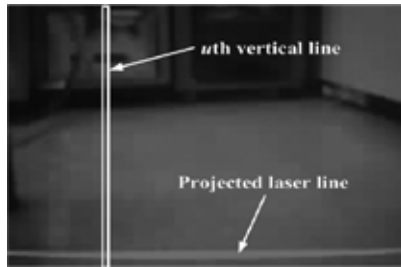


Fig. 6. Concept of an adaptive vertical threshold scheme.

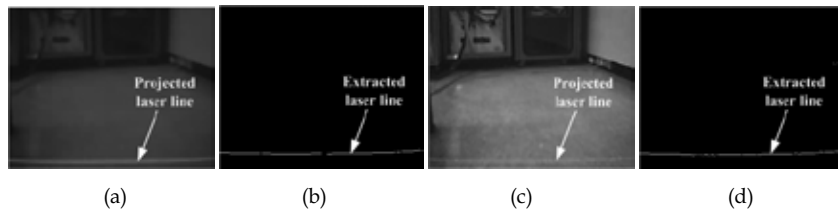


Fig. 7. Laser-line detection using an adaptive vertical threshold scheme: (a) original image and (b) its vertical thresholded image for low intensity of illumination; (c) original image and (d) its vertical thresholded image for high intensity of illumination.

3.3 Acquisition of 3D Information

In this section, we obtain 3D information from the detected laser line on the 2D camera image using the geometry of the terrain-prediction sensor module. The mobile base frame $\{B\}$ of the agent and the camera frame $\{C\}$ of the terrain prediction sensor with respect to the universal frame $\{U\}$ are depicted in Fig. 8, where the Y_c -axis is set parallel with the Y_b -axis. The X_b -axis of $\{B\}$ is parallel with the heading direction of the agent and the Z_b -axis is normal to the surface of the ground. The Y_b -axis is defined perpendicular to the X_b - Z_b plane and its direction is determined by the right-hand-rule (RHR). The origin of $\{B\}$ is the agent center position (ACP), which is the projected point of the CG on the X_b - Z_b plane. In this research, all other coordinate systems are also defined in accordance with the RHR. According to the relation between $\{B\}$ and $\{C\}$, point $P_c(x_c, y_c, z_c) \in \mathbb{R}^3$ relative to $\{C\}$ can be transformed into point $P_b(x_b, y_b, z_b) \in \mathbb{R}^3$ relative to $\{B\}$ as follows:

$$\begin{bmatrix} x_b \\ y_b \\ z_b \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_{bc} & 0 & \sin \theta_{bc} & l_{bc} \\ 0 & 1 & 0 & 0 \\ -\sin \theta_{bc} & 0 & \cos \theta_{bc} & h_{bc} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \quad (6)$$

where l_{bc} and h_{bc} are the translational distances between origins of $\{B\}$ and $\{C\}$ about the X_b -axis and the Z_b -axis, respectively, and θ_{bc} is the angle between $\{B\}$ and $\{C\}$ about the Y_b -axis.

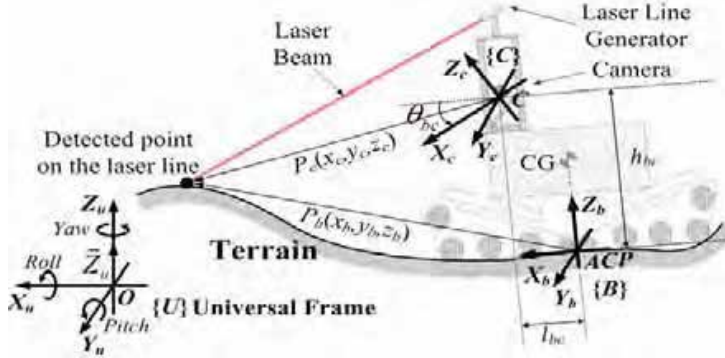


Fig. 8. Transformation of point $P_c(x_c, y_c, z_c) \in \mathbb{R}^3$ relative to $\{C\}$ into point $P_b(x_b, y_b, z_b) \in \mathbb{R}^3$ relative to $\{B\}$.

In Fig. 9, point $P_c(x_c, y_c, z_c) \in \mathbb{R}^3$ on the laser line with respect to $\{C\}$ is obtained from point $P_{img}(u_1, u_2) \in \mathbb{R}^2$ on the image plane by comparing the similar triangles $\Delta P_c M C$ and $\Delta P_{img} M' C$ as follows:

$$\begin{bmatrix} x_c & y_c & z_c \end{bmatrix} = \frac{b'}{f \cot \theta_{ip} + u_2} \begin{bmatrix} f & u_1 & u_2 \end{bmatrix} \quad (7)$$

where f is the focal length of the camera, θ_{ip} is the projection angle of the laser line on the image plane, and b' is the distance between the center of the camera lens C and the

intersection L' of the Z_c -axis and the laser beam. According to the Sine's Law, the distance b' in triangle $\Delta LCL'$ can be obtained as follows:

$$b' = \frac{b \sin(\theta_{lp} - \theta_{bc})}{\sin \theta_{lp}} \quad (8)$$

where b is the baseline distance between the center of the laser line generator L and the camera center C . By (7) and (8), point $P_{img}(u_1, u_2) \in \mathbb{R}^2$ on the image plane can be transformed into point $P_b(x_b, y_b, z_b) \in \mathbb{R}^3$ relative to $\{B\}$.

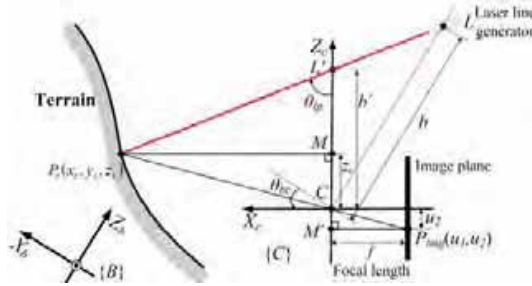


Fig. 9. Geometry between point $P_c(x_c, y_c, z_c) \in \mathbb{R}^3$ on the laser line and point $P_{img}(u_1, u_2) \in \mathbb{R}^2$ on the image plane relative to $\{C\}$.

3.4 Acquisition of Terrain Parameters

The terrain data at a distance of D_{tr} in front of the agent consist of the roll and pitch angles of the agent set on that terrain. As shown in Fig. 10, the roll angle of the front terrain relative to the current roll angle of the agent is predicted as follows:

$$\Delta \theta_{Roll}(k) = \tan^{-1} \left(\frac{z'_{bL}(k) - z'_{bR}(k)}{D_{track}} \right) \quad (9)$$

where D_{track} is a distance between the right and left tracks of the agent. $P'_{bR}(k)$ and $P'_{bL}(k)$ are the contact points of the right and left tracks with the front terrain at a distance of D_{tr} , where $P'_{bR}(k)$ and $P'_{bL}(k)$ are denoted as $(x'_{bR}(k), y'_{bR}(k), z'_{bR}(k))$ and $(x'_{bL}(k), y'_{bL}(k), z'_{bL}(k))$, respectively. $P'_{bR}(k)$ and $P'_{bL}(k)$ are obtained as following steps:

1. Store the detected points $P_{bR}(k)$ and $P_{bL}(k)$ for the right and left tracks on the laser line and the translational velocity $v(k)$ in the memory at time k .
2. For each time instant, find the minimum times Δk_1 and Δk_2 satisfying the following conditions:

$$x_{bR}(k - \Delta k_1) \cos(\theta_{3DM-Pitch}(k - \Delta k_1)) - \sum_{n=1}^{\Delta k_1} v(k-n) T_s \cos(\theta_{3DM-Pitch}(k-n)) < D_{tr} \quad (10)$$

$$x_{bL}(k - \Delta k_2) \cos(\theta_{3DM-Pitch}(k - \Delta k_2)) - \sum_{n=1}^{\Delta k_2} v(k-n) T_s \cos(\theta_{3DM-Pitch}(k-n)) < D_{tr} \quad (11)$$

where $\theta_{3DM-Pitch}(k)$ is the pitch angle of the agent obtained by the inclinometer at time k relative to $\{U\}$.

- Using Δk_1 and Δk_2 satisfying (10) and (11), obtain $P'_{bR}(k)$ and $P'_{bL}(k)$ by the linear interpolation of $P_{bR}(k-\Delta k_1+1)$ and $P_{bR}(k-\Delta k_1)$ and the linear interpolation of $P_{bL}(k-\Delta k_2+1)$ and $P_{bL}(k-\Delta k_2)$, respectively.

Finally, the roll angle relative to $\{U\}$ is obtained from the predicted roll angle $\Delta\theta_{Roll}(k)$ relative to $\{B\}$ as follows:

$$\hat{\theta}_{Roll}(k) = \theta_{3DM-Roll}(k) + \Delta\theta_{Roll}(k) \quad (12)$$

where $\theta_{3DM-Roll}(k)$ is the roll angle of the agent obtained by the inclinometer at time k relative to $\{U\}$.

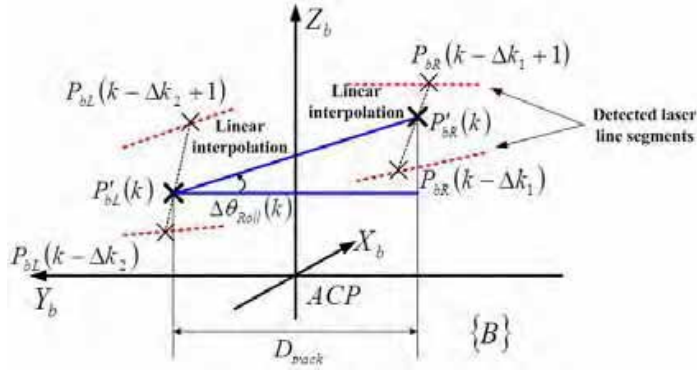


Fig. 10. Predicted roll angle $\Delta\theta_{Roll}(k)$ at a distance of D_{tr} relative to the roll angle at time k by using interpolated points $P'_{bL}(k)$ and $P'_{bR}(k)$ at time k .

As shown in Fig. 11, the pitch angle of the front terrain relative to the current pitch angle of the agent is predicted by the terrain data obtained at times k and $k-\Delta k_3$ as follows:

$$\Delta\theta_{Pitch}(k) = -\tan^{-1}\left(\frac{z'_{bF}(k) - z'_{bF}(k-\Delta k_3)}{x'_{bF}(k) - x'_{bF}(k-\Delta k_3)}\right) \quad (13)$$

where Δk_3 is the minimum time satisfying the condition $L_{fr} \leq |P'_{bF}(k)P'_{bF}(k-\Delta k_3)|$. Here, L_{fr} is the length of the agent tracks, and $|P'_{bF}(k)P'_{bF}(k-\Delta k_3)|$ is the distance between points $P'_{bF}(k)$ and $P'_{bF}(k-\Delta k_3)$. Point $P'_{bF}(k)$ is defined by points $P'_{bR}(k)$ and $P'_{bL}(k)$ as follows:

$$\begin{aligned} P'_{bF}(k) &= (x'_{bF}(k), y'_{bF}(k), z'_{bF}(k)) \\ &= \left(D_{tr}, 0, \frac{z'_{bL}(k) + z'_{bR}(k)}{2}\right) \end{aligned} \quad (14)$$

To obtain the distance $|P'_{bF}(k)P'_{bF}(k-\Delta k_3)|$, point $P_{bF}(k-\Delta k_3)$ relative to base frame $\{B(k-\Delta k_3)\}$ defined at time $k-\Delta k_3$ needs to be transformed into point $P'_{bF}(k-\Delta k_3)$ relative to $\{B(k)\}$ (or $\{B\}$) defined at time k as follows:

$$P'_{bF}(k - \Delta k_3) = P_{bF}(k - \Delta k_3) - \begin{bmatrix} \sum_{n=1}^{\Delta k_3} v(k-n)T_s \cos(\theta_{3DM-Pitch}(k-n)) \\ 0 \\ -\sum_{n=1}^{\Delta k_3} v(k-n)T_s \sin(\theta_{3DM-Pitch}(k-n)) \end{bmatrix} \quad (15)$$

The second term on the right-hand side of (15) indicates the displacement vector between $\{B(k-\Delta k_3)\}$ and $\{B(k)\}$. Finally, the pitch angle relative to $\{U\}$ is obtained by the predicted pitch angle $\Delta\theta_{Pitch}(k)$ as follows:

$$\hat{\theta}_{Pitch}(k) = \theta_{3DM-Pitch}(k) + \Delta\theta_{Pitch}(k) \quad (16)$$

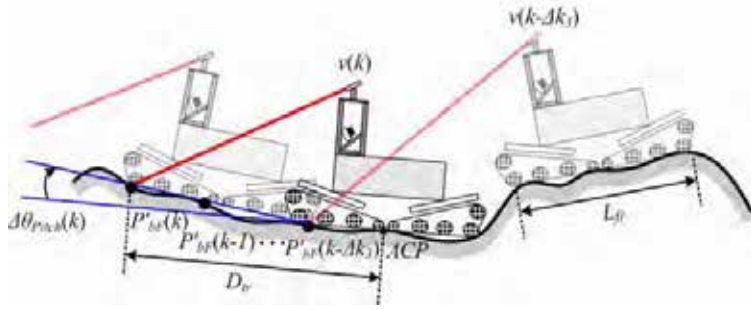


Fig. 11. Predicted pitch angle $\Delta\theta_{Pitch}(k)$ at a distance of D_{tr} relative to the pitch angle at time k by using interpolated points $P'_{bF}(k)$ and $P'_{bF}(k-\Delta k_3)$ at times k and $k-\Delta k_3$, respectively.

4. Turnover Prevention through Prediction

In this section, a turnover prevention algorithm for preventing the agent from pitching over or rolling over is discussed. The pitchover-free range of the translational acceleration and the rollover-free range of the rotational velocity are determined by using the predicted-terrain sensor data. According to both ranges, the translational and rotational velocities of the agent are controlled for pitchover and rollover prevention.

4.1 Dynamics of the Agent

In order to determine turnover constraints for the agent moving through unknown terrain, we adopt the quasi-static rollover analysis of a rigid vehicle (Gillespie, 1992). By assuming the ROBHAZ-DT as a rigid vehicle, the deflections of the suspensions and tracks need not be considered in the analysis. The external forces acting on the agent consist of the friction forces between the vehicle and ground, the normal force, and the gravity force. The total friction force F , tangent to the X_b - Y_b plane, can be defined as follows:

$$F = f_{X_b}X_b + f_{Y_b}Y_b \quad (17)$$

where f_{X_b} and f_{Y_b} are the components tangent and normal to the heading direction of the agent, respectively. By modifying the dynamic-motion equations for the car-like agent

described by Shiller (Shiller, 1991), the motion equation for a differential-drive agent moving through unknown terrain can be described in terms of the translational velocity v and the translational acceleration a as follows:

$$f_{Xb}\mathbf{X}_b + f_{Yb}\mathbf{Y}_b + N\mathbf{Z}_b - mg\mathbf{Z}_u = mv^2/r\mathbf{Y}_b + ma\mathbf{X}_b \quad (18)$$

where N is the magnitude of the normal force in the direction of \mathbf{Z}_b , m is the lumped mass of the agent, and r is the turning radius of the agent. Radius r can be represented as v/ω since the agent is a differential-drive vehicle. Parameters f_{Xb} , f_{Yb} and N can be obtained by the dot products of the unit vectors \mathbf{X}_b , \mathbf{Y}_b and \mathbf{Z}_b with (18), respectively, as follows:

$$f_{Xb} = mgk_{Xb} + ma \quad (19)$$

$$f_{Yb} = mgk_{Yb} + mv^2/r = mgk_{Yb} + mv\omega \quad (20)$$

$$N = mgk_{Zb} \quad (21)$$

where k_{Xb} , k_{Yb} and k_{Zb} are terrain parameters defined by the projections of unit vector \mathbf{Z}_u on unit vectors \mathbf{X}_b , \mathbf{Y}_b and \mathbf{Z}_b , respectively. Vector \mathbf{Z}_u is the unit vector $[0 \ 0 \ 1]^T$ in the opposite direction of the gravity $[0 \ 0 \ -g]^T$ relative to $\{U\}$. The terrain parameters are represented by the roll and pitch angles of that terrain as follows:

$$k_{Xb} = \mathbf{Z}_u \cdot \mathbf{X}_b = -\sin(\theta_{Pitch}) \quad (22)$$

$$k_{Yb} = \mathbf{Z}_u \cdot \mathbf{Y}_b = \sin(\theta_{Roll})\cos(\theta_{Pitch}) \quad (23)$$

$$k_{Zb} = \mathbf{Z}_u \cdot \mathbf{Z}_b = \cos(\theta_{Roll})\cos(\theta_{Pitch}) \quad (24)$$

where θ_{Roll} and θ_{Pitch} are determined according to the conventional method of the X - Y - Z fixed angles.

4.2 Pitchover Prevention Control

The force distribution of the agent is depicted in Figs. 12 (a) and 12 (b) when the agent pitches over CCW and CW about the Y_b -axis, respectively. At the point where the agent is about to pitch over CCW, the total normal force N and the friction force f_{Xb} of the agent are applied on the only front endpoint of the track. Thus the moment on the agent created by those forces should satisfy the condition $f_{Xb}l + NL_{fr}/2 \geq 0$ for preventing a pitchover in a CCW direction, where h is the height of the center of gravity (CG) of the agent. In the same way, the moment on the agent should satisfy the condition $f_{Xb}l - NL_{fr}/2 \leq 0$ for preventing a pitchover in a CW direction, where forces N and f_{Xb} are applied on the only rear endpoint of the track. The resultant condition for preventing a pitchover can be determined by combining the above conditions as follows:

$$-N \frac{L_{fr}}{2h} \leq f_{Xb} \leq N \frac{L_{fr}}{2h} \quad (25)$$

Substituting (19) and (21) into (25) transforms the resultant condition to an inequality equation in a as follows:

$$-g \left(\frac{L_{fr}}{2h} k_{Zb} + k_{Xb} \right) \leq a \leq g \left(\frac{L_{fr}}{2h} k_{Zb} - k_{Xb} \right) \quad (26)$$

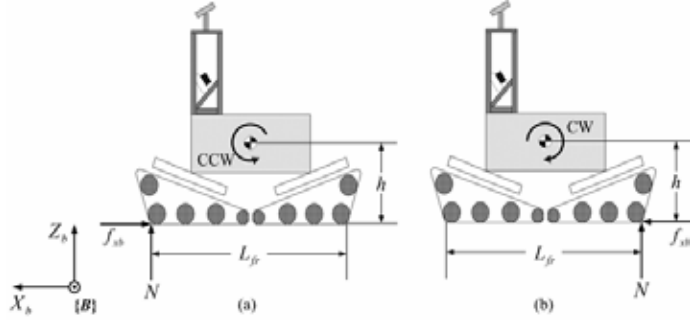


Fig. 12. Force distribution of the agent which is about to pitch over (a) CCW and (b) CW.

Hereafter, the upper and lower bounds of a in (26) are denoted as a_{ub} and a_{lb} , respectively. Bounds a_{ub} and a_{lb} are represented as surfaces in $\theta_{Roll}-\theta_{Pitch}-a$ space as shown in Fig. 13, where θ_{Roll} and θ_{Pitch} replace k_{Xb} and k_{Zb} in (26). That is, the inner region between the upper and lower surfaces indicates a safe region of the translational acceleration for preventing a pitchover. In this case, the permitted accelerations of the agent for accelerated, uniform and decelerated motions are represented as three planes $a=a_c$, $a=0$ and $a=-a_c$ in Fig. 13.

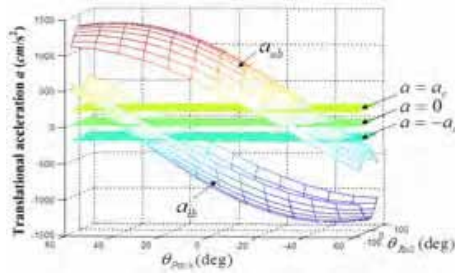


Fig. 13. Graphical analysis of the condition for preventing a pitchover (a_c : normal acceleration of the agent).

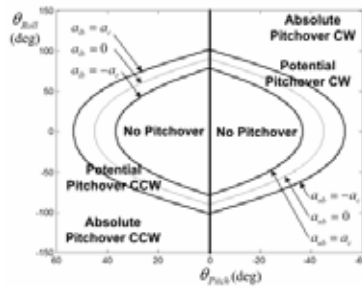


Fig. 14. Five cases for pitchover possibility of the agent according to the roll and pitch angles.

According to the relation of the two surfaces and the three planes, five possible cases of a pitchover are defined as shown in Fig. 14. Each case is determined by the intersection curves of two surfaces with three planes. According to the five cases, the control strategies of the translational velocity for pitchover prevention are described in Table 2. For pitchover prevention control, the pitchover possibility is determined by the front terrain data which are predicted by the terrain-prediction sensor. When the agent detects the terrain for the absolute pitchover CW or CCW case, the agent must decelerate to zero because all permitted accelerations of the agent are beyond the boundary of the safe region of the translational acceleration and thus the agent will unconditionally pitch over at the detected terrain. As a result of deceleration, the agent can stop before arriving at the dangerous terrain. For the potential pitchover CW case, the agent must maintain its velocity or decelerate since it is allowed to only move in uniform and decelerated motions to avoid the CW pitchover. Especially, if the agent detects the terrain where it must decelerate in order to prevent from pitching over CW, it will decelerate and stop before it reaches that terrain. That is, the agent does not enter that pitchover region since it already stops at around the vicinity of the region. On the other hand, in the case of the potential pitchover CCW case, the agent must maintain its velocity or accelerate to avoid the CCW pitchover. In this case, the agent can not accelerate further after its translational velocity reaches the maximum velocity. At this point of view, the agent must decelerate and stop before it arrives at that terrain. The potential pitchover CW case is similar to the potential pitchover CCW case explained before. Finally, in the no pitchover case, the agent is allowed to move in accelerated, uniform and decelerated motions. In other words, the agent need not be controlled for pitchover prevention.

Cases	Permissible acc. ranges	Possible motions	Control strategies
No Pitchover	$-a_c \leq a \leq a_c$ ($a_{ub} > a_c$ and $a_{lb} < -a_c$)	Accelerated Uniform Decelerated	($a = a_c$) ($a = 0$) ($a = -a_c$) Needless
Potential Pitchover CW	$-a_c \leq a \leq 0$ ($0 \leq a_{ub} < a_c$ and $a_{lb} < -a_c$) $-a_c \leq a \leq -a_c$ ($-a_c \leq a_{ub} < 0$ and $a_{lb} < -a_c$)	Uniform Decelerated Decelerated	($a = 0$) ($a = -a_c$) ($a = -a_c$) Maintain the translational velocity/ Decelerate to zero
Potential Pitchover CCW	$0 \leq a \leq a_c$ ($a_{ub} > a_c$ and $-a_c < a_{lb} \leq 0$) $a_c \leq a \leq a_c$ ($a_{ub} > a_c$ and $0 < a_{lb} \leq a_c$)	Uniform Accelerated Accelerated	($a = 0$) ($a = a_c$) ($a = a_c$) Decelerate to zero
Absolute Pitchover (CCW or CW)	None ($a_{ub} < -a_c$ or $a_{lb} > a_c$)	None	Decelerate to zero

Table 2. Control strategies for preventing the pitchover of the agent.

4.3 Rollover Prevention Control

The force distribution of the agent is depicted in Figs. 15 (a) and 15 (b) where the agent rolls over CCW and CW, respectively. In the case where the agent is about to roll over CCW, the total normal force N and the friction force f_{yb} of the agent are applied on the only left track. Thus, the moment on the agent created by those forces should satisfy the condition $f_{yb}l + NW_b/2 \geq 0$ for preventing a rollover in a CCW direction. In the same way, the moment on the agent should satisfy the condition $f_{yb}l - NW_b/2 \leq 0$ for preventing a rollover in a CW

direction where the forces N and f_{y_b} are applied on the only right track as shown in Fig. 15 (b). Therefore, the resultant condition to prevent a rollover can be determined by combining the above conditions as follows:

$$-N \frac{W_b/2}{h} \leq f_{y_b} \leq N \frac{W_b/2}{h} \quad (27)$$

Substituting (20) and (21) into (27) transforms the resultant condition to an inequality equation in v and ω as follows:

$$-g \left(k_{y_b} + \frac{W_b/2}{h} k_{z_b} \right) \leq v\omega \leq -g \left(k_{y_b} - \frac{W_b/2}{h} k_{z_b} \right) \quad (28)$$

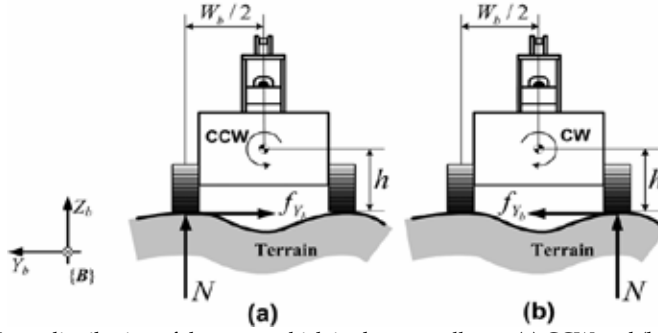


Fig. 15. Force distribution of the agent which is about to roll over (a) CCW and (b) CW.

Hereafter, the upper and lower bounds of $v\omega$ in (28) are denoted as $(v\omega)_{ub}$ and $(v\omega)_{lb}$, respectively. In this case, the translational velocity v is determined by the operator's command and the condition of pitchover prevention. Thus, for the given v , the inequality equation (28) can be represented in terms of ω as follows:

$$\frac{(v\omega)_{lb}}{\min(v + \Delta v, v_{\max})} \leq \omega \leq \frac{(v\omega)_{ub}}{\min(v + \Delta v, v_{\max})} \quad (29)$$

where Δv is the maximum increase of the translational velocity while the agent is moving the distance of D_{tr} : $\Delta v = -v + (v^2 + 2a_c D_{tr})^{1/2}$. Due to the motor torque constraints, translational velocity $v + \Delta v$ is restricted by v_{\max} . Here, the upper and lower bounds in (29) are denoted as ω_{ub} and ω_{lb} , respectively. The rollover-free region of the rotational velocity is defined as the inner region between surfaces ω_{ub} and ω_{lb} in $\theta_{Roll} - \theta_{Pitch} - \omega$ space as shown in Fig. 16. In this figure, three planes $\omega = \omega_{\max}$, $\omega = 0$ and $\omega = -\omega_{\max}$ for the rotational velocity are also depicted with the surfaces, where ω_{\max} is the maximum rotational velocity of the agent. According to the relation of the two surfaces and the three planes, the control regions for rollover prevention are defined as shown in Fig. 17. The boundaries b_{ui} and b_{lj} for $i, j = 1, 2, 3$ are determined by the intersection curves of surfaces ω_{ub} and ω_{lb} with the three planes, respectively. According to the five control regions, the control strategies of the translational and rotational velocities for rollover prevention are described in Table 3. For the free moving region A, the rotational velocity of the agent can be determined for the

entire permissible range from $-\omega_{\max}$ to ω_{\max} . That is, the operator can control the agent with no restriction for the rotational velocity. On the contrary, for the restricted regions B1 and B2, the rotational velocity must be restricted for preventing a rollover. If the detected terrain is in B1, the rotational velocity is truncated to range from $-\omega_{\max}$ to ω_{ub} since $\omega_{ub} < \omega_{\max}$. Especially, for the region between b_{u2} and b_{u3} , the agent is allowed to only turn right since $\omega_{ub} < 0$. In other words, the agent cannot turn left and go straight. For the region B2, the rotational velocity is truncated to range from ω_{lb} to ω_{\max} since $-\omega_{\max} < \omega_{lb}$. Similarly to the case of B1, for the region between b_{l2} and b_{l3} , the agent is allowed to only turn left since $\omega_{lb} < 0$. Finally, if the detected terrain is in the uncontrollable regions C1 and C2, the agent must stop before arriving at that terrain because the whole range from $-\omega_{\max}$ to ω_{\max} is beyond the safe range from ω_{ub} to ω_{lb} and the agent will unconditionally roll over at that terrain.

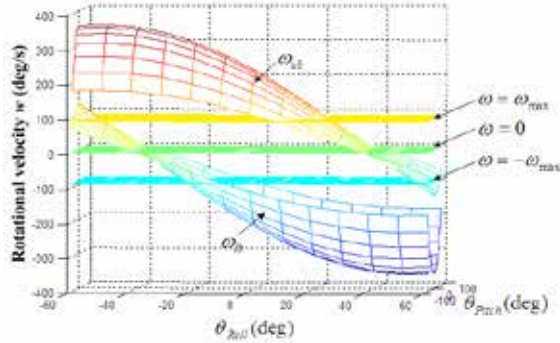


Fig. 16. Graphical analysis of the condition for preventing a rollover (ω_{\max} : maximum rotational velocity of the agent).

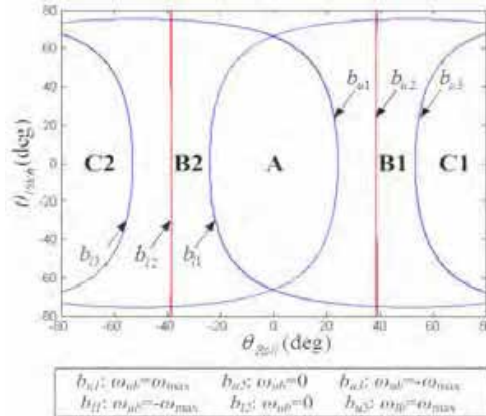


Fig. 17. Control regions for rollover prevention according to roll and pitch angles (A: free moving region; B1, B2: restricted regions; C1, C2: uncontrollable regions).

Control regions	Rollover-free ranges of the rotational vel.	Control strategies
Free moving region A	$-\omega_{\max} \leq \omega \leq \omega_{\max}$ ($\omega_{ub} \geq \omega_{\max}$ and $\omega_{lb} \leq -\omega_{\max}$)	Needless
Restricted region B1	$-\omega_{\max} \leq \omega \leq \omega_{ub}$ ($\omega_{ub} < \omega_{\max}$ and $\omega_{lb} \leq -\omega_{\max}$)	Restrict the rotational velocity by ω_{ub}
Restricted region B2	$\omega_{lb} \leq \omega \leq \omega_{\max}$ ($\omega_{ub} \geq \omega_{\max}$ and $\omega_{lb} > -\omega_{\max}$)	Restrict the rotational velocity by ω_{lb}
Uncontrollable region C1	None ($\omega_{ub} < -\omega_{\max}$)	Reduce the translational velocity to zero/Stop
Uncontrollable region C2	None ($\omega_{lb} > \omega_{\max}$)	Reduce the translational velocity to zero/Stop

Table 3. Control strategies for preventing the rollover of the agent.

5. Reflective Force Generation

5.1 Force Reflection System

It is possible that turnover prevention control can cause inconsistencies between the driving command of the operator and the reactive motion of the agent. Thus, a reflective force is generated to compensate the inconsistencies. The experimental setup for force reflection is depicted in Fig. 18. The WingMan Force Pro joystick of Logitech is employed as a 2 DOF force feedback joystick which not only receives a command of an operator but also generates a reflective force. The joystick interface is developed by using the Microsoft DirectX 8.0 Software Development Kit (SDK). The positions about the X-axis and the Y-axis of the joystick coordinates determine the rotational and translational velocities of the agent, respectively.

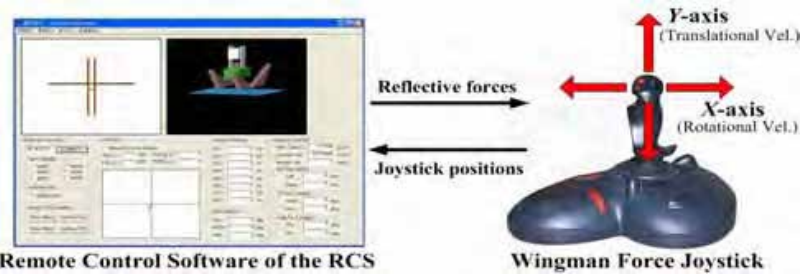


Fig. 18. Experimental setup for force reflection with the Logitech Wingman Force Pro joystick.

5.2 Position-based Reflective Force for Turnover Prevention

The position-based force \mathbf{F}_R is depicted in Fig. 19. The force \mathbf{F}_R is determined by the position \mathbf{q} about the axis of the joystick coordinates as follows:

$$\mathbf{F}_R = \begin{cases} -\mathbf{k}_{NC} \cdot [\mathbf{q} - (\mathbf{q}_{\text{offset}} - \mathbf{W}_{DB})], & \mathbf{q} < (\mathbf{q}_{\text{offset}} - \mathbf{W}_{DB}) \\ -\mathbf{k}_{PC} \cdot [\mathbf{q} - (\mathbf{q}_{\text{offset}} + \mathbf{W}_{DB})], & \mathbf{q} > (\mathbf{q}_{\text{offset}} + \mathbf{W}_{DB}) \end{cases} \quad (30)$$

where the parameters of the position-based force are described in Table 4. If \mathbf{q} is apart from $\mathbf{q}_{\text{offset}}$, the reflective force is generated for pushing the joystick to $\mathbf{q}_{\text{offset}}$. In other words, the position-based force makes it difficult for the operator to push the joystick far from $\mathbf{q}_{\text{offset}}$. The force parameters F_{PS} and k_{PC} for $\mathbf{q} > \mathbf{q}_{\text{offset}}$ and the parameters F_{NS} and k_{NC} for $\mathbf{q} < \mathbf{q}_{\text{offset}}$ can be determined independently. In addition, as the dead-band for the reflective force can be defined by W_{DB} around $\mathbf{q}_{\text{offset}}$, no reflective force is generated if \mathbf{q} is located between $(\mathbf{q}_{\text{offset}} - W_{\text{DB}})$ and $(\mathbf{q}_{\text{offset}} + W_{\text{DB}})$. Thus, the sensitivity to a slight displacement of \mathbf{q} around $\mathbf{q}_{\text{offset}}$ can be reduced.

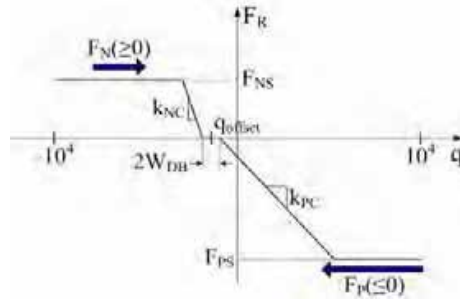


Fig. 19. Parameters of the position-based force F_R for the joystick position \mathbf{q} .

Parameters	Descriptions	Ranges	
		(from)	(to)
$\mathbf{q}_{\text{offset}}$	Reference position of the position-based force	-10^4	10^4
W_{DB}	Dead-band defined relative to $\mathbf{q}_{\text{offset}}$, where no force is generated	0	10^4
k_{NC}	Proportional factor of the force on the negative side of $\mathbf{q}_{\text{offset}}$ when $\mathbf{q} < (\mathbf{q}_{\text{offset}} - W_{\text{DB}})$	-10^4	10^4
k_{PC}	Proportional factor of the force on the positive side of $\mathbf{q}_{\text{offset}}$ when $\mathbf{q} > (\mathbf{q}_{\text{offset}} - W_{\text{DB}})$	-10^4	10^4
F_{NS}	The maximum force on the negative side of $\mathbf{q}_{\text{offset}}$	-10^4	10^4
F_{PS}	The maximum force on the positive side of $\mathbf{q}_{\text{offset}}$	-10^4	10^4

Table 4. Parameters of the position-based reflective force.

For pitchover prevention, the reflective force about the Y-axis of the joystick coordinates is generated as shown in Fig. 20 (a). As described in Section 4.2, if the agent detects pitchovers at front terrain, it must keep its translational velocity or decelerate to zero to avoid a pitchover. That is, the desired translational velocity v_d for pitchover prevention is set as the current translational velocity of the agent or decreased continuously. Through the reflective force, the operator recognizes that the translational velocity is restricted by v_d . If the operator pushes the joystick in the positive direction above the joystick position for v_d , he/she will feel a repulsive force in the negative direction. On the other hand, if the operator pulls the joystick in the negative direction below the joystick position for v_d , he/she will feel no reflective force. Therefore, the operator can recognize the upper limit v_d for pitchover prevention by the repulsive force. The parameters of the reflective force are determined as $\mathbf{q}_{\text{offset}} = f_1(v_d)$, $W_{\text{DB}} = 10^2$, $F_{\text{NS}} = 0$, $F_{\text{PS}} = 10^4$, $k_{\text{NS}} = 0$ and $k_{\text{PS}} = 10^4$, where $f_1(\cdot)$ is a mapping function

of the desired translational velocity onto the joystick position. In this case, the only $\mathbf{q}_{\text{offset}}$ is changed according to the desired translational velocity v_d for pitchover prevention.

For rollover prevention, a reflective force about the X-axis is generated as shown in Fig. 20 (b). If the agent detects a possible rollover at the front terrain, the safety range of its rotational velocity is determined to avoid rollovers as discussed in Section 4.3. The operator can detect the safety range through the reflective force while driving the agent. If the operator maneuvers the agent within this safety range of the rotational velocity, no reflective force is generated. Thus, the operator can drive the agent without any restriction. However, if the operator pushes the joystick beyond the safety region, he will feel a reflective force which pushes the joystick in the direction of the safety region. That is, if the operator pushes the joystick above the joystick position for the upper bound of the safety region, the reflective force in the negative direction is generated to prevent from being pushed in the positive direction. Also, in the case where the operator pushes the joystick below the joystick position for the lower bound of the safety region, the reflective force in the positive direction is generated to prevent from being pushed in the negative direction. The parameters $\mathbf{q}_{\text{offset}}$ and \mathbf{W}_{DB} of the reflective force about the X-axis are determined according to the safety region of the rotational velocity as follows:

$$\begin{aligned} \mathbf{q}_{\text{offset}} &= f_2(\omega_{lb}, \omega_{ub}) \\ &= 10^4 \cdot \frac{1}{2} \cdot \frac{[\max(\omega_{lb}, -\omega_{\max}) + \min(\omega_{ub}, \omega_{\max})]}{\omega_{\max}} \end{aligned} \quad (31)$$

$$\begin{aligned} \mathbf{W}_{\text{DB}} &= f_3(\omega_{lb}, \omega_{ub}) \\ &= 10^4 \cdot \frac{1}{2} \cdot \frac{[\min(\omega_{ub}, \omega_{\max}) - \max(\omega_{lb}, -\omega_{\max})]}{\omega_{\max}} \end{aligned} \quad (32)$$

where $f_2(\cdot)$ is a mapping function of the center of the safety region onto the joystick position and $f_3(\cdot)$ is a mapping function of the width of the safety region onto the dead-band of the reflective force. The other parameters are determined as $\mathbf{F}_{\text{NS}}=10^4$, $\mathbf{F}_{\text{PS}}=10^4$, $\mathbf{k}_{\text{NS}}=10^4$ and $\mathbf{k}_{\text{PS}}=10^4$. In this case, the parameters $\mathbf{q}_{\text{offset}}$ and \mathbf{W}_{DB} are changed according to the safety region for rollover prevention. As a result of reflective force generation, the operator can intuitively determine how to drive the agent for turnover prevention.

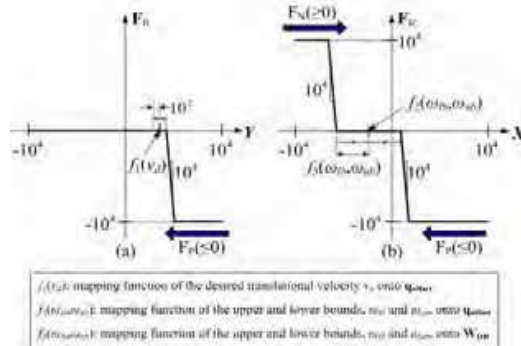


Fig. 20. Reflective forces for recognizing (a) the desired translational velocity for pitchover prevention and (b) the safe region of the rotational velocity for rollover prevention.

6. Experimental Results

Two experiments were carried out with the ROBHAZ-DT in order to verify the feasibility of the proposed turnover prevention algorithm. The sampling time T_s was set as 100 ms. Two resultant paths of the agent moving on the sloped terrain are depicted in Fig. 21. The system parameters for the experiments are described in Table 5.

Parameters	Descriptions
$v_{\max}=860$	The maximum translational velocity [mm/s]
$\omega_{\max}=90$	The maximum rotational velocity [$^{\circ}$ /s]
$a_c=1500$	The normal acceleration [mm/s ²]
$W_b=48$	The width between two tracks of the agent [cm]
$h_1=25, h_2=70$	The height of the center of gravity of the agent [cm]
$D_{tr}=68$	The reference distance for the turnover prevention [cm]

Table 5. System parameters for the experiments about turnover prevention.

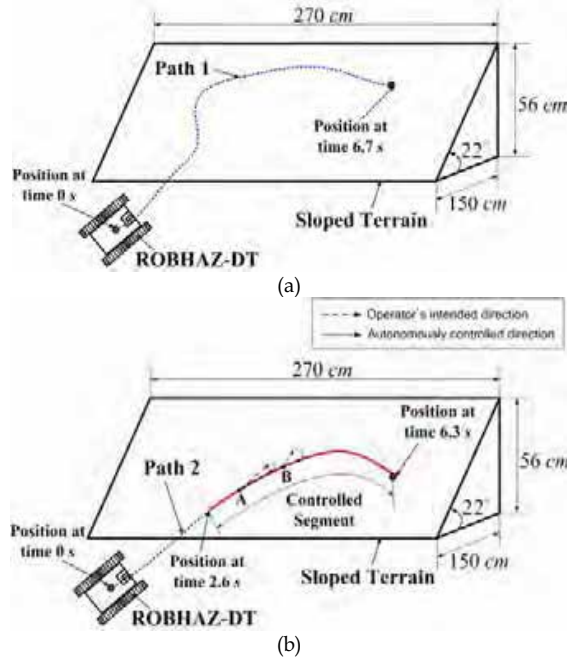
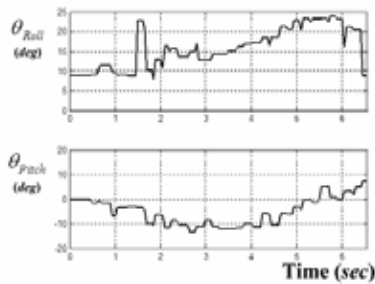


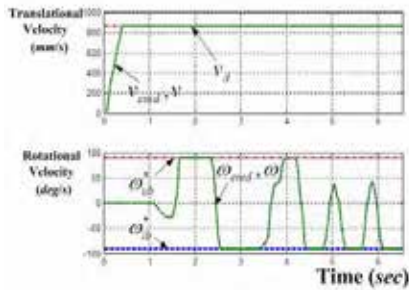
Fig. 21. Resultant paths of the ROBHAZ-DT moving on the sloped terrain: (a) Path1 for $h_1=25$ cm, and (b) Path2 for $h_2=70$ cm where the solid line segment indicates the controlled path for turnover prevention.

The first experiment was carried out with an only mobile base of the ROBHAZ-DT, where $h_1=25$ cm. The agent moved for 6.7 s as shown in Path1 of Fig. 21 (a). The terrain data at a distance of D_{tr} in front of the agent are depicted in Fig. 22 (a). These terrain data were

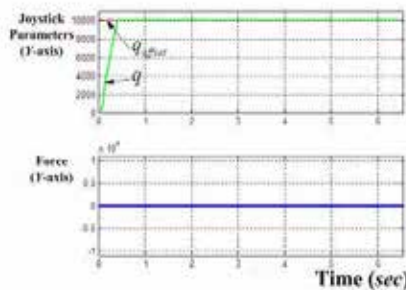
predicted by the terrain-prediction sensor and used for turnover prevention. In this experiment, no turnover was detected in the front terrain and thus the translational and rotational velocities of the agent need not be controlled for turnover prevention. That is, the desired translational velocity v_d for pitchover prevention was set as the maximum translational velocity v_{max} and the safety region of the rotational velocity covered the whole range of the rotational velocity of the agent as shown in Fig. 22 (b). Also, reflective force for turnover prevention was not generated and thus the operator could freely control the agent as shown in Figs.22 (c) and 22 (d).



(a)



(b)



(c)

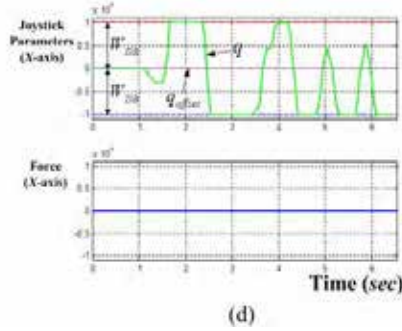
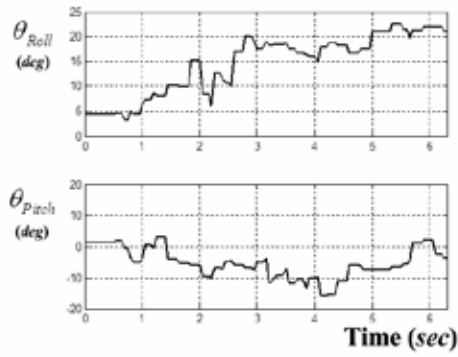


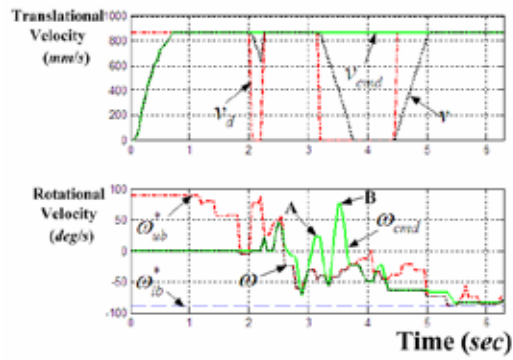
Fig. 22. Experimental results about turnover prevention ($W_b=48$ cm and $h_1=25$ cm): (a) Terrain parameters at a distance of D_{tr} in front of the agent, (b) rotational and translational velocities, (c) joystick parameters about the Y-axis, and (d) joystick parameters about the X-axis.

The second experiment was carried out using the mobile base with a manipulator. In this experiment, we assumed that the configuration of the manipulator was fixed while the agent was in motion since the action of the manipulator might bring about a change of the center of gravity (CG) of the agent. In this case, although the CG of the agent was not changed, the height of the CG rose up to $h_2=70$ cm due to the mass of the manipulator attached to the mobile base. In the second experiment, the agent moved for 6.3 s as shown in Path2 of Fig. 21 (b). The solid line segment of Path2 indicates that the agent was autonomously controlled for turnover prevention. Especially, at A and B of Path2, the intended direction of the operator is modified for turnover prevention. If the agent is still controlled by the operator at A and B, it will overturn soon.

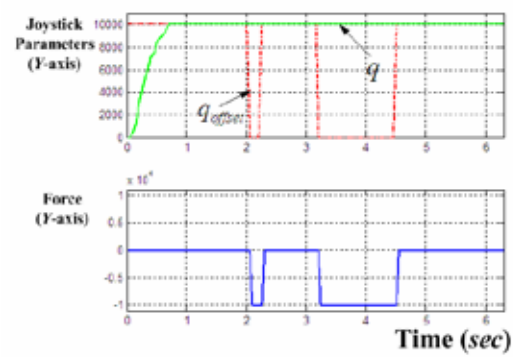
The terrain data at a distance of D_{tr} in front of the agent are depicted in Fig. 23 (a). In this case, the agent detected turnovers in the front terrain and thus the translational and rotational velocities of the agent were controlled as shown in Fig. 23 (b). For the given terrain data at each time instant, the desired translational velocity v_d was autonomously controlled for pitchover prevention and the translational velocity v_{cmd} of the operator's command was restricted by v_d . As shown in Fig. 23 (b), the resultant translational velocity v was decelerated by $-a_c$ and accelerated by a_c to follow the desired velocity v_d . Also, the upper bound ω_{ub} of the safety region of the rotational velocity was determined for rollover prevention and the rotational velocity ω_{cmd} of the operator's command was restricted by ω_{ub} . At A and B of Fig. 23 (b), the resultant rotational velocity ω was restricted by ω_{ub} , since ω_{cmd} exceeded ω_{ub} . Here, A and B of Fig. 23 (b) correspond to A and B of Fig. 21 (b), respectively. As shown in Fig. 23 (c), when the joystick position for v_{cmd} exceeded v_d , the reflective force about the Y-axis was generated in the negative direction. As a result, the operator felt a repulsive force preventing him/her from pushing above the position for v_d , and hence recognized that the translational velocity of the agent was restricted by v_d for pitchover prevention. Also, when the joystick position for ω_{cmd} exceeded the upper bound of the safety region, the reflective force about the X-axis was generated in the negative direction and vice versa. Thus, through the reflective force, the operator could intuitively recognize the safety region of the rotational velocity for rollover prevention and thus be guided to control the rotational velocity within the safety range.



(a)



(b)



(c)

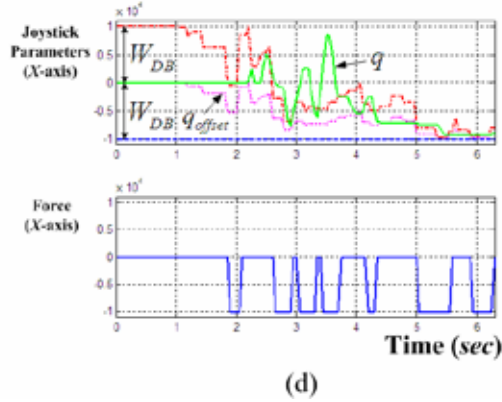


Fig. 23. Experimental results about turnover prevention ($W_b=48$ cm and $h_2=70$ cm): (a) Terrain parameters at a distance of D_{tr} in front of the agent, (b) rotational and translational velocities, (c) joystick parameters about the Y-axis, and (d) joystick parameters about the X-axis.

7. Conclusions

The turnover prevention control algorithm of a teleoperated mobile agent was presented. For online prediction of front terrain, a low-cost terrain prediction sensor composed of a camera vision, a laser line generator, and an inclinometer was developed. The terrain parameters were obtained by finding structured laser line projected onto the front terrain and used for turnover prevention control through the quasi-static rollover analysis. As a result of turnover prevention control, the translational and rotational velocities of the agent were restricted. However, the velocity restriction for turnover prevention may bring about the inconsistencies between the intended motion and the reactive motion of the agent. Thus, the force reflection technique was proposed in order to compensate the inconsistencies. Through the position-based reflective force, the operator could intuitively recognize how the agent should be controlled to avoid turnovers. Finally, based on the experimental results, we found that the agent can even avoid turnovers in unknown sloped terrain.

8. Future Works

In future works, the proposed algorithm for a mobile manipulator with a moving manipulator will be studied. As the manipulator motion brings about a change of center of gravity, a change of the center of gravity of the agent needs to be considered simultaneously.

9. Acknowledgement

This work was supported in part by the Korea Institute of Science and Technology, in part by the Science Research Center/Engineering Research Center program of Ministry of Science and Technology/Korea Science and Engineering Foundation under Grant R11-1999-008, and in part by the Automation and Systems Research Institute.

10. References

- Acarman, T. & Özgüner, Ü. (2003). Rollover prevention for heavy trucks using frequency shaped sliding mode control. *IEEE International Conference on Control Application*, pp.7-12, Istanbul, Turkey, June, 2003
- Ando, N.; Korondi, P. & Hashimoto, H. (2001). Development of micromanipulator and haptic interface for networked micromanipulation. *IEEE Transactions on Mechatronics*, Vol.6, No.4, 2001, pp.417-427
- Basdogan, C.; De, S. & Kim, J. (2004). Haptics in minimally invasive surgical simulation and training. *IEEE Computer Graphics and Applications*, Vol.24, No.2, 2004, pp.56-64
- Borenstein, J. & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.19, No. 15, September/October, 1989, pp. 1179-1187
- Borenstein, J. & Koren, Y. (1991a). The vector field histogram-Fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, Vol.7, No.3, June, 1991, pp.278-288
- Borenstein, J. & Koren, Y. (1991b). Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, Vol.7, No.4, August, 1991, pp.535-539
- Boukhifir, M.; Ferreira, A. & Fontaine, J. (2004). Scaled teleoperation controller design for micromanipulation over internet. *IEEE International Conference on Robotics and Automation*, pp.4577-4583, New Orleans, LA, USA, April, 2004
- Chen, B. & Peng, H. (1999). A real-time rollover threat index for sports utility vehicles. *American Control Conference*, pp.1233-1237, San Diego, CA, June, 1999
- Chen, E. & Marcus, B. (1998). Force feedback for surgical simulation. *Invited paper, Proceedings of IEEE*, Vol.86, No.3, pp.524-530, March, 1998
- Gillespie, T. (1992). *Fundamentals of Vehicle Dynamics*. Warrendale, PA: Society of Automotive Engineers, Inc., 1992
- Howard, A. & Seraji, H. (2001). An intelligent terrain-based navigation system for planetary rovers. *IEEE Robotics and Automation Magazine*, Vol.8, No.4, 2001, pp.9-17
- Huang, Q.; Sugano, S. & Tanie, K. (1994). Stability control for a mobile manipulator using a potential method. *IEEE International Conference on Intelligent Robots and Systems*, pp.839-846, Munich, Germany, September, 1994
- Huang, Q.; Sugano, S. & Kato, I. (1994). Stability compensation of a mobile manipulator by manipulator motion: Feasibility and planning. *IEEE International Conference on Intelligent Robots and Systems*, pp.1285-1292, Grenoble, France, September, 1997
- Jain, B.; Kasturi, R. & Schunck, B. (1995). *Machine Vision*, New York: McGraw-Hill
- Kim, J.; Lee, H. & Yang, M. (2002). Robotic contamination cleaning system. *IEEE International Conference on Intelligent Robots and Systems*, pp.1874-1879, Lausanne, Switzerland, October, 2002
- Lin, Q. & Kuo, C. (1997). Virtual tele-operation of underwater robots, *IEEE International Conference on Robotics and Automation*, pp.1022-1027, Albuquerque, NM, USA, 1997
- Nalecz, A. & Bindenmann, A. (1987). Sensitivity analysis of vehicle design attributes that affect vehicle response in critical accident situations-Part I: User's manual. *DOT-HS-807-229*, Washington, D.C.: Department of Transportation, 1987
- Nalecz, A. (1991). Intermediate maneuver induced rollover simulation (IMIRS)-A sensitivity analysis. *DOT-HS-807-672*, Washington, D.C.: Department of Transportation, February, 1991

- Nalecz, A.; Lu, A. & d'Entremont, K. (1993). An investigation into dynamic measures of vehicle rollover propensity. *SAE Technical Paper 930831*, Detroit, MI: Society of Automotive Engineers, 1993
- Niwa, Y.; Yukita, S. & Hanaizumi, H. (2004). Depthmap-based obstacle avoidance on rough terrain. *IEEE International Conference on Intelligent Robotics and Systems*, pp.1612-1617, Sendai, Japan, September, 2004
- Nudehi, S.; Mukherjee, R. & Ghodoussi, M. (2005). A shared-control approach to haptic interface design for minimally invasive telesurgical training. *IEEE Transactions on Control Systems Technology*, Vol.13, No.4, 2005, pp.588-592
- Papadopoulos, G. & Rey, D. (1996). A new measure of tipover stability margin for mobile manipulators. *IEEE International Conference on Robotics and Automation*, pp.3111-3116, Minneapolis, MN, April, 1996
- Park, J.; Lee, B. & Chung, W. (2003a). Reflective force navigation control for a mobile robot using a state transition diagram. *IEEE International Conference on Advanced Intelligent Mechatronics*, pp.52-57, Kobe, Japan, July, 2003
- Park, J.; Lee, B. & Kim, M. (2003b). Remote control of a mobile robot using distance-based reflective force. *IEEE International Conference on Robotics and Automation*, pp.600-605, Taipei, Taiwan, R.O.C., September, 2003
- Park, J. & Lee, B. (2004). Reflective force integration method for nonautonomous mobile robot control. *IEEE International Conference on Robotics and Automation*, pp.3950-3955, New Orleans, LA, USA, April, 2004
- Park, J.; Lee, J. & Lee, B. (2006a). Online turnover-free control for a mobile agent with a terrain prediction sensor. *Journal of Field Robotics*, Vol.23, Issue 1, January, 2006, pp.59-77
- Park, J.; Lee, J. & Lee, B. (2006b). Rollover-free navigation for a mobile agent in an unstructured environment. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, Vol.36, No.4, August, 2006, pp.835-848
- Rey, A. & Papadopoulos, E. (1997). Online automatic tipover prevention for mobile manipulators. *IEEE International Conference on Intelligent Robots and Systems*, pp.1273-1278, Grenoble, France, September, 1997
- Shiller, Z. & Gwo, Y. (1991). Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, Vol.7, No.2, April, 1991, pp.241-249
- Singh, S.; Simmons, R.; Smith, T.; Stentz, A.; Verma, V.; Yahja, A. & Schwehr, K. (2000). Recent progress in local and global traversability for planetary rovers. *IEEE International Conference on Robotics and Automation*, pp.1194-1200, San Francisco, CA, April, 2000
- Smith, F.; Backman, D. & Jacobsen, S. (1992). Telerobotic manipulator for hazardous environments. *Journal of Robotic Systems*, Vol.9, No.2, 1992, pp.251-260
- Sugano, S.; Huang, Q. & Kato, I. (1993). Stability criteria in controlling mobile robotic systems. *IEEE International Conference on Intelligent Robots and Systems*, pp.832-838, Yokohama, Japan, July 1993
- Takano, S. & Nagai, M. (2001). Dynamics control of large vehicles for rollover prevention. *IEEE International Conference on Vehicle Electronics*, pp.85-89, Tottori, Japan, September, 2001

Dynamics and Control for Nonholonomic Mobile Modular Manipulators

Yangmin Li & Yugang Liu
University of Macau
Macao S. A .R., P. R. China

1. Introduction

The development of a robot requires that it be able to adopt as many configurations as possible using limited modules, so as to allow the construction of new types of robots without redesign and remanufacturing. Traditionally, modular manipulators are mounted on a fixed base whose mobility is constrained. However, with the development of industry and technology, such modular manipulators as mounted on fixed bases can not meet some practical requirements any more. An intelligent and autonomous mobile manipulator, which can fulfil some operations without human interference, has become an active research topic recently since it has many potential applications such as in modern factories for transporting materials, in dangerous fields for dismantling bombs or moving nuclear infected objects, in modern families for doing housework, as well as in the public places for city maintenance.

In this chapter, a nonholonomic mobile platform is attached to the modular manipulator in order to increase workspace of the entire robot. Building up the dynamic model for a nonholonomic mobile modular manipulator is a challenging task due to the interactive motions between the modular manipulator and the mobile platform, as well as the nonholonomic constraints of the mobile platform. Also a trajectory following task becomes even more complex and difficult to achieve.

Such conventional control strategies as computed-torque control require precise apriori knowledge of the dynamic parameters for the controlled system. However, in practical applications, it is almost impossible to obtain exact dynamic parameters for a mobile modular manipulator because of such uncertainties as complex nonlinear frictions, flexibilities of the joints and links, payload variations, and terrain irregularities. Robust control techniques provide a natural rejection to external disturbances, which are provided by a high-frequency commuted control action that constrains the error trajectories to stay on the sliding surface. Classical sliding mode control law adopts sign functions and the caused chattering may do harm to the robots. Adaptive control technique does not rely on precise apriori knowledge of dynamic parameters and it can suppress such errors as caused by parameter uncertainties by online adjusting dynamic parameters. Furthermore, adaptive control can counteract the negative influence of high-frequency switching caused by robust control because its action has naturally smooth time behaviour.

In related work on modular robots, the modular robot concept could be traced back to the 1970's (Will & Grossman, 1975). In early modular robot research, the emphasis was put on the structure design of self-organizing, self-reconfigurable, self-assembling, and self-repairable modular robots (Fukuda et al., 1989; Tomita et al., 1999). Kinematic and dynamic analysis as well as trajectory planning became another active topic in the past decades (Chen & Yang, 1998; Fei et al., 2001). In recent years, the scholars had turned their attentions to trajectory following control for modular manipulators (Melek & Goldenberg, 2003; Shen et al., 2002; Stoy et al., 2002). Parameter identification and vibration control for a 9-DOF reconfigurable modular manipulator were investigated by authors in (Li et al., 2004a).

Regarding to literatures on mobile manipulators, mobile manipulators were exploited to install and remove aircraft warning spheres (Campos et al., 2002), to polish aircraft canopy (Jamisola et al., 2002), to organize furniture in a room (Rus et al., 1995), and to collectively transport a single palletized load (Stilwell & Bay, 1993). A great deal of research activities can be found on motion planning of mobile manipulators (Carriker et al., 1991; Chitta & Ostrowski, 2002; Nagatani et al., 2002). Several kinematic and dynamic modelling methods were presented for mobile manipulators in the past decade, such as the Kane's method (Tanner & Kyriakopoulos, 2001), the Newton-Euler method (Chung & Velinsky, 1999) and the Lagrange method (Li & Liu, 2004b; Liu & Li, 2005a; Yu & Chen, 2002). Tip-over analysis and prevention attracted numerous scholars and several tip-over stability criteria were defined, such as the potential energy stability level (Ghasempoor & Sepehri, 1995), the force-angle stability measure (Papadopoulos & Rey, 1996), the zero moment point criterion (Furuno et al., 2003), and the criterion based on supporting forces (Li & Liu, 2005b). Extensive literatures can be found on control of mobile manipulators. Dynamic characteristics between the mobile platform and the onboard manipulator were investigated (Yamamoto & Yun, 1996). A robust control method was developed to eliminate the harmful effect of the wheel slip on the tracking performance of a spatial mobile manipulator (Chung & Velinsky, 1999). A homogeneous kinematic stabilization strategy and an adaptive control scheme were combined for mobile manipulator control without any knowledge of the system dynamic model (Colbaugh, 1998). Neural network and fuzzy logic control for mobile manipulators were also studied by authors (Li & Liu, 2005c, 2005d, 2006a, 2006b).

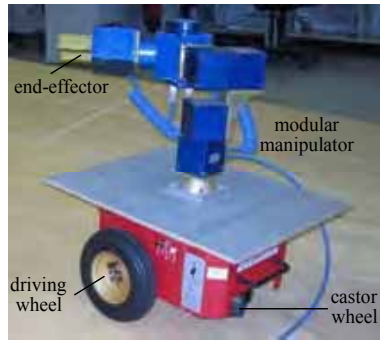
In the previous research work, modeling for the mobile platform and for the manipulator was usually carried out separately and control for nonholonomic mobile robots was mostly limited to kinematic velocity control, while few work on dynamic torque control. The interactive motions between the manipulator and the mobile platform made the models established inaccurate, which then affected the control result. Most of the present controllers were designed in joint space, but few in task space (Ge et al., 1997). However, in practical applications, the end-effector of a robot is usually specified to fulfil some operations.

This chapter is organized as follows: an integrated modelling method is proposed considering nonholonomic constraints and interactive motions in Section 2. In Section 3, a robust adaptive controller is designed in task space to control the end-effector to follow desired spatial trajectories. Simulations are conducted on a real mobile modular manipulator, and a comparison is made with the conventional model-based controller in Section 4. Section 5 gives some concluding remarks. Some suggested ideas for future research work are presented in the last section.

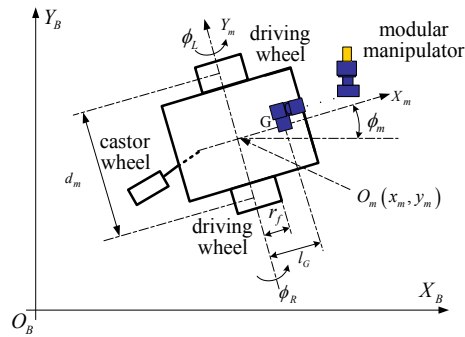
2. An Integrated Modelling Method

2.1 Kinematics analysis

A mobile modular manipulator is normally composed of an m -wheeled holonomic or nonholonomic mobile platform and an n -DOF onboard modular manipulator, as shown in Fig. 1(a). In this chapter, we analyze the 3-wheeled nonholonomic mobile platform, which has two driving wheels and one castor wheel. The two driving wheels are coaxial and mounted in front of the platform and the castor wheel in the rear is orientable with respect to the cart. The robot is assumed to move on a horizontal plane; then the motion of the mobile platform can be illustrated as shown in Fig. 1(b). An arbitrary inertial base frame $O_B X_B Y_B Z_B$ is fixed on the motion plane, while a frame $O_m X_m Y_m Z_m$ is attached to the mobile platform. In frame $O_m X_m Y_m Z_m$, $O_m(x_m, y_m)$ is selected as the midpoint of the line segment connecting the two driving-wheel centres; $O_m Y_m$ is along the coaxial of the two driving wheels; $O_m X_m$ is perpendicular to $O_m Y_m$ and passes through O_m . The heading angle ϕ_m determines the posture of the mobile platform.



(a) Prototype



(b) coordinate system definition

Fig. 1. Prototype and coordinate system definition for a mobile modular manipulator.

Since the modular manipulator is mounted on the mobile platform, it can be assumed that the relative position of frame $O_m X_m Y_m Z_m$ and the frame $O_0 X_0 Y_0 Z_0$ is steady; here $O_0 X_0 Y_0 Z_0$ is the frame attached to the 1st module of the modular manipulator. Therefore the mobile platform can be treated as a special module added to the bottom of the modular manipulator, which can both move on the motion plane and rotate about the vertical axis. From Fig. 1(b), transformation matrices between the frames $O_B X_B Y_B Z_B$, $O_m X_m Y_m Z_m$ and $O_0 X_0 Y_0 Z_0$ are given by

$${}^B_m T = \begin{bmatrix} \cos \phi_m & -\sin \phi_m & 0 & x_m \\ \sin \phi_m & \cos \phi_m & 0 & y_m \\ 0 & 0 & 1 & r_f \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^m_0 T = \begin{bmatrix} 1 & 0 & 0 & l_G \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h_G \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where r_f is the radius for the driving wheels; l_G and h_G denote offsets of the 1st module with respect to $O_m X_m Y_m Z_m$ along $O_m X_m$, $O_m Y_m$ respectively.

By defining the link four parameters according to the Denavit-Hartenberg (D.H.) notation (de Wit et al., 1996), the transformation matrix between two adjacent links for the modular manipulator can be derived as follows

$${}^{i-1}_i T = \begin{bmatrix} \cos q_i & -\sin q_i & 0 & l_i \\ \sin q_i \cos \alpha_i & \cos q_i \cos \alpha_i & -\sin \alpha_i & -d_i \sin \alpha_i \\ \sin q_i \sin \alpha_i & \cos q_i \sin \alpha_i & \cos \alpha_i & d_i \cos \alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

where l_i, α_i, d_i, q_i are D.H. parameters.

Therefore the transformation matrix of the end effector with respect to the inertial base frame can be derived by

$${}^B_n T = {}^B_0 T {}^0_1 T {}^1_2 T \cdots {}^{n-1}_n T \quad (3)$$

Hence, the position vector \vec{p}_e and the posture vectors $\vec{n}, \vec{o}, \vec{a}$ of the end-effector with respect to the inertial base frame can be derived, which can be observed from the fourth column and the first three columns of the matrix ${}^B_n T$ correspondingly.

However, it is inconvenient to describe the posture of the end-effector using these posture vectors with nine parameters. The posture can be determined by only three independent parameters using Z-Y-Z Euler angles ϕ, θ and ψ . Relationship between the posture vectors and the Euler angles can be described as follows

$$\begin{aligned} \phi &= a \tan 2(a_y, a_x), \\ \theta &= a \tan 2(o_z, -n_z), \\ \psi &= a \tan 2(a_x \cos \phi + a_y \sin \phi, a_z) \end{aligned} \quad (4)$$

2.2 Nonholonomic constraints and Jacobian

The motion of a mobile modular manipulator during a very short time interval $[t^i \ t^{i+1}]$ is shown in Fig. 2. l_r , d_r and d_m represent the distances from the fixed bar to $O_m Y_m$, from the fixed bar to the horizontal axis of the castor wheel, and that between the two driving wheels respectively; r_i and θ_i are steering radius and yaw angle during the time interval; ϕ_L , ϕ_R , ϕ_r and β_r denote rotating angles of the left driving wheel, the right driving wheel, the castor wheel around its horizontal axis, and that around the fixed bar respectively; ΔS_L , ΔS_R and ΔS_m represent advances of the left driving wheel, the right driving wheel, and the mobile platform during the time interval correspondingly.

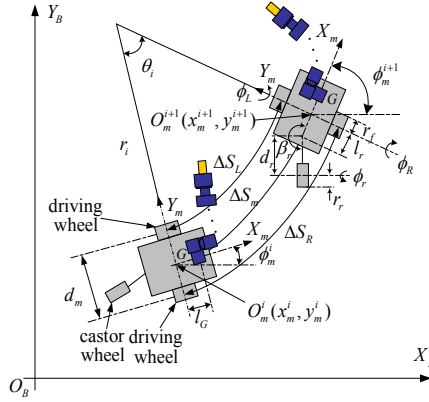


Fig. 2. Nonholonomic constraints derivation.

Let the time interval $\Delta t = t^{i+1} - t^i \rightarrow 0$, on the assumption of low speeds, velocities during this time interval can be treated as constants. Then,

$$\begin{cases}
 \Delta x_m^i = |O_m^i O_m^{i+1}| \cdot \cos \phi_m^i & (a) \\
 \Delta y_m^i = |O_m^i O_m^{i+1}| \cdot \sin \phi_m^i & (b) \\
 \Delta S_L = \theta_i \cdot \left(r_i - \frac{d_m}{2} \right) & (c) \\
 \Delta S_R = \theta_i \cdot \left(r_i + \frac{d_m}{2} \right) & (d) \\
 \Delta S_m \approx |O_m^i O_m^{i+1}| & (e) \\
 \theta_i \approx \Delta \phi_m^i & (f) \\
 r_i = \frac{\Delta S_m}{\theta_i} & (g)
 \end{cases} \quad (5)$$

where $\Delta x_m^i = x_m^{i+1} - x_m^i$, $\Delta y_m^i = y_m^{i+1} - y_m^i$ and $\Delta \phi_m^i = \phi_m^{i+1} - \phi_m^i$.

Substituting Eq. 5(f, g) into Eq. 5(c, d) yields

$$\begin{cases} \Delta S_L = \Delta S_m - \frac{d_m}{2} \cdot \Delta \phi_m^i & (a) \\ \Delta S_R = \Delta S_m + \frac{d_m}{2} \cdot \Delta \phi_m^i & (b) \end{cases} \quad (6)$$

Adding Eq. 6(a) and Eq. 6(b) yields

$$\Delta S_m = \frac{\Delta S_L + \Delta S_R}{2} \quad (7)$$

Subtracting Eq. 6(a) for Eq. 6(b) yields

$$\Delta \phi_m = - \frac{\Delta S_L - \Delta S_R}{d_m} \quad (8)$$

Substituting Eq. 7 and Eq. 5(e) into Eq. 5(a,b) yields

$$\begin{cases} \Delta x_m^i = \frac{\cos \phi_m^i}{2} \cdot (\Delta S_L + \Delta S_R) \\ \Delta y_m^i = \frac{\sin \phi_m^i}{2} \cdot (\Delta S_L + \Delta S_R) \end{cases} \quad (9)$$

From Eq. 8 and Eq. 9,

$$\begin{cases} \dot{x}_m^i = \lim_{\Delta t \rightarrow 0} \left(\frac{\Delta x_m^i}{\Delta t} \right) = \frac{\cos \phi_m^i}{2} \cdot \left[\lim_{\Delta t \rightarrow 0} \left(\frac{\Delta S_L}{\Delta t} \right) + \lim_{\Delta t \rightarrow 0} \left(\frac{\Delta S_R}{\Delta t} \right) \right] = \frac{\cos \phi_m^i}{2} \cdot (\dot{S}_L + \dot{S}_R) \\ \dot{y}_m^i = \lim_{\Delta t \rightarrow 0} \left(\frac{\Delta y_m^i}{\Delta t} \right) = \frac{\sin \phi_m^i}{2} \cdot \left[\lim_{\Delta t \rightarrow 0} \left(\frac{\Delta S_L}{\Delta t} \right) + \lim_{\Delta t \rightarrow 0} \left(\frac{\Delta S_R}{\Delta t} \right) \right] = \frac{\sin \phi_m^i}{2} \cdot (\dot{S}_L + \dot{S}_R) \\ \dot{\phi}_m^i = \lim_{\Delta t \rightarrow 0} \left(\frac{\Delta \phi_m^i}{\Delta t} \right) = -\frac{1}{d_m} \cdot \left[\lim_{\Delta t \rightarrow 0} \left(\frac{\Delta S_L}{\Delta t} \right) - \lim_{\Delta t \rightarrow 0} \left(\frac{\Delta S_R}{\Delta t} \right) \right] = -\frac{1}{d_m} \cdot (\dot{S}_L - \dot{S}_R) \end{cases} \quad (10)$$

Equation 10 always holds during the course of plane motion, so the superscript "i" can be omitted. Considering that $\dot{S}_L = r_f \cdot \dot{\phi}_L$ and $\dot{S}_R = r_f \cdot \dot{\phi}_R$, we can obtain

$$\begin{cases} \dot{x}_m = \frac{r_f \cdot \cos \phi_m}{2} \cdot \dot{\phi}_L + \frac{r_f \cdot \cos \phi_m}{2} \cdot \dot{\phi}_R & (a) \\ \dot{y}_m = \frac{r_f \cdot \sin \phi_m}{2} \cdot \dot{\phi}_L + \frac{r_f \cdot \sin \phi_m}{2} \cdot \dot{\phi}_R & (b) \\ \dot{\phi}_m = -\frac{r_f}{d_m} \cdot \dot{\phi}_L + \frac{r_f}{d_m} \cdot \dot{\phi}_R & (c) \end{cases} \quad (11)$$

In the same way, we can obtain

$$\begin{cases} \dot{\phi}_r = \frac{\cos(\beta_r + \phi_m)}{r_r} \cdot \dot{x}_m + \frac{\sin(\beta_r + \phi_m)}{r_r} \cdot \dot{y}_m - \frac{l_r \sin \beta_r}{r_r} \cdot \dot{\phi}_m \\ \dot{\beta}_r = \frac{\sin(\beta_r + \phi_m)}{d_r} \cdot \dot{x}_m - \frac{\cos(\beta_r + \phi_m)}{d_r} \cdot \dot{y}_m - \frac{d_r - l_r \cos \beta_r}{d_r} \cdot \dot{\phi}_m \end{cases} \quad (12)$$

Adding Eq. 11(a) multiplied by $\cos \phi_m$ to Eq. 11(b) multiplied by $\sin \phi_m$ yields

$$\cos \phi_m \cdot \dot{x}_m + \sin \phi_m \cdot \dot{y}_m = \frac{r_f}{2} \cdot \dot{\phi}_L + \frac{r_f}{2} \cdot \dot{\phi}_R \quad (13)$$

Adding Eq. 11(c) multiplied by $\frac{d_m}{2}$ to Eq. 13 yields

$$\cos \phi_m \cdot \dot{x}_m + \sin \phi_m \cdot \dot{y}_m + \frac{d_m}{2} \cdot \dot{\phi}_m - r_f \cdot \dot{\phi}_R = 0 \quad (14)$$

Subtracting Eq. 11(c) multiplied by $\frac{d_m}{2}$ from Eq. 13 yields

$$\cos \phi_m \cdot \dot{x}_m + \sin \phi_m \cdot \dot{y}_m - \frac{d_m}{2} \cdot \dot{\phi}_m - r_f \cdot \dot{\phi}_L = 0 \quad (15)$$

Subtracting Eq. 11(b) multiplied by $\cos \phi_m$ from Eq. 11(a) multiplied by $\sin \phi_m$ yields

$$\sin \phi_m \cdot \dot{x}_m - \cos \phi_m \cdot \dot{y}_m = 0 \quad (16)$$

From Eq. 12 and Eq. 14-16, we have

$$\begin{bmatrix} \cos \phi_m & \sin \phi_m & -\frac{d_m}{2} & 0 & 0 & -r_f & 0 \\ \cos \phi_m & \sin \phi_m & \frac{d_m}{2} & 0 & 0 & 0 & -r_f \\ \sin \phi_m & -\cos \phi_m & 0 & 0 & 0 & 0 & 0 \\ \cos(\beta_r + \phi_m) & \sin(\beta_r + \phi_m) & -l_r \sin \beta_r & -r_r & 0 & 0 & 0 \\ \sin(\beta_r + \phi_m) & -\cos(\beta_r + \phi_m) & l_r \cos \beta_r - d_r & 0 & -d_r & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\phi}_m \\ \dot{\phi}_r \\ \dot{\beta}_r \\ \dot{\phi}_L \\ \dot{\phi}_R \end{bmatrix} = 0 \quad (17)$$

in short $A(\xi) \cdot \dot{\xi} = 0$; here $\xi = [x_m \ y_m \ \phi_m \ \phi_r \ \beta_r \ \phi_L \ \phi_R]^T$.

From Eq. 11 and Eq. 12, we can easily obtain

$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\phi}_m \\ \dot{\phi}_r \\ \dot{\beta}_r \\ \dot{\phi}_L \\ \dot{\phi}_R \end{bmatrix} = \begin{bmatrix} \frac{r_f \cdot \cos \phi_m}{2} & \frac{r_f \cdot \cos \phi_m}{2} \\ \frac{r_f \cdot \sin \phi_m}{2} & \frac{r_f \cdot \sin \phi_m}{2} \\ -\frac{r_f}{d_m} & \frac{r_f}{d_m} \\ \frac{r_f (d_m \cos \beta_r + 2l_r \sin \beta_r)}{2d_m r_r} & \frac{r_f (d_m \cos \beta_r - 2l_r \sin \beta_r)}{2d_m r_r} \\ \frac{r_f (d_m \sin \beta_r + 2d_r - 2l_r \cos \beta_r)}{2d_m d_r} & \frac{r_f (d_m \sin \beta_r - 2d_r + 2l_r \cos \beta_r)}{2d_m d_r} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi}_L \\ \dot{\phi}_R \end{bmatrix} \quad (18)$$

in short $\dot{\xi} = S(\xi) \cdot [\dot{\phi}_L \ \dot{\phi}_R]^T$.

Substituting Eq. 18 into Eq. 17 yields

$$A(\xi) \cdot S(\xi) = 0 \quad (19)$$

Let $x = [p_x \ p_y \ p_z \ \phi \ \theta \ \psi]^T$ and $q = [\phi_L \ \phi_R \ q_1 \ q_2 \ \dots \ q_n]^T$ be coordinates in task space and joint space respectively; and define $\varsigma = [\xi^T \ q_1 \ \dots \ q_n]^T$ as extended coordinates, and then the Jacobian matrix can be derived by

$$J = \frac{\partial x}{\partial q} = \frac{\partial x}{\partial \varsigma} \cdot \begin{bmatrix} S(\xi) & 0_{7 \times n} \\ 0_{n \times 2} & I_{n \times n} \end{bmatrix} \quad (20)$$

2.3 Dynamic modelling

With the assumption of moving on a horizontal plane, the mobile platform has a constant potential energy U_m . To calculate the kinetic energy, the mobile platform can be divided into four parts, the cart (including all the driving units in the box), the left front wheel, the right front wheel and the rear wheel, that is

$$T_m = T_c + T_{lf} + T_{rf} + T_r \quad (21)$$

where T_m is the entire kinetic energy for the mobile platform; T_c , T_{lf} , T_{rf} and T_r denote respectively kinematic energy of the cart, the left and right front wheel and the rear wheel. Kinetic energy for the cart can be calculated by

$$T_c = \frac{1}{2} \cdot m_c \cdot \dot{x}_c^2 + \frac{1}{2} \cdot m_c \cdot \dot{y}_c^2 + \frac{1}{2} \cdot I_c \cdot \dot{\phi}_m^2 \quad (22)$$

where m_c is the mass of the cart; $x_c = x_m + l_c \cos \phi_m$, $y_c = y_m + l_c \sin \phi_m$ denote coordinates for the mass centre of the cart, with l_c being distance between the mass centre and O_m ; I_c represents the moment of inertia for the cart around the axis of $O_m Z_m$.

With the assumption of rolling without slipping, the contact points of the wheels with the motion plane can be treated as instant rotating centres. Then, kinematic energy for the front driving wheels can be given by

$$T_{lf} = \frac{3}{4} \cdot m_f \cdot r_f^2 \cdot \dot{\phi}_L^2, \quad T_{rf} = \frac{3}{4} \cdot m_f \cdot r_f^2 \cdot \dot{\phi}_R^2 \quad (23)$$

where m_f is the mass for the front driving wheels.

The castor wheel can rotate about both its axis and the fixed bar that does not pass through its centre, as shown in Fig. 3. So, the kinetic energy of the rear castor wheel includes two parts as given by Eq. 24.

$$T_r = \frac{1}{2} \cdot I_{r1} \cdot \dot{\phi}_r^2 + \frac{1}{2} \cdot I_{r2} \cdot \dot{\beta}_r^2 \quad (24)$$

where the moments of inertia I_{r1} and I_{r2} can be determined by

$$I_{r1} = \frac{1}{2} \cdot m_r \cdot r_r^2 + m_r \cdot r_r^2 = \frac{3}{2} \cdot m_r \cdot r_r^2 \quad (25)$$

$$I_{r2} = \frac{1}{4} \cdot m_r \cdot r_r^2 + m_r \cdot d_r^2 = m_r \cdot \left(\frac{r_r^2}{4} + d_r^2 \right)$$

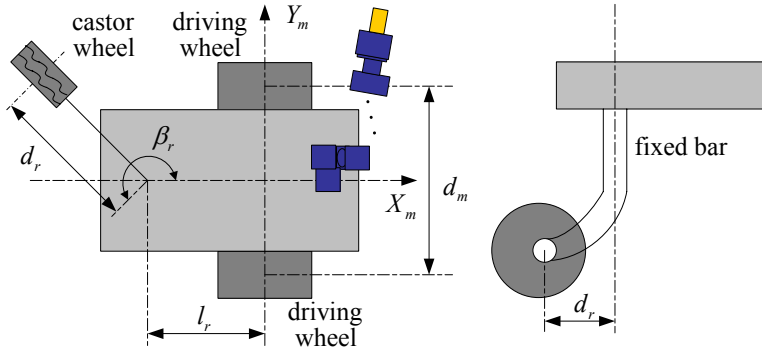


Fig. 3. Structure for the castor wheel.

Kinematic energy for the onboard modular manipulator can be calculated by

$$T_M = \sum_{i=0}^n \left(\frac{1}{2} \cdot v_{ci}^T \cdot m_i \cdot v_{ci} + \frac{1}{2} \cdot \omega_i^T \cdot I_i \cdot \omega_i \right) \quad (26)$$

where T_M denotes kinetic energy for the modular manipulator; m_i and I_i are mass and inertial moment matrix for the i^{th} module respectively; v_{ci} and ω_i represent the linear and angular velocities for the mass centre of the i^{th} module.

Potential energy for the onboard modular manipulator can be calculated by

$$U_M = \sum_{i=0}^n \{m_i \cdot g^T \cdot p_{ci}\} \quad (27)$$

where $g = [0 \ 0 \ 9.81]^T$ and $p_{ci} = [x_i^B \ y_i^B \ z_i^B]^T$.

Then, the Lagrange function can be given by

$$L = T_m - U_m + T_M - U_M \quad (28)$$

The constraint dynamics for the nonholonomic mobile modular manipulator can be determined by

$$\begin{cases} \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}_m} \right) - \frac{\partial L}{\partial x_m} = \lambda_1 \cos \phi_m + \lambda_2 \cos \phi_m + \lambda_3 \sin \phi_m + \lambda_4 \cos(\beta_r + \phi_m) + \lambda_5 \sin(\beta_r + \phi_m) \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{y}_m} \right) - \frac{\partial L}{\partial y_m} = \lambda_1 \sin \phi_m + \lambda_2 \sin \phi_m - \lambda_3 \cos \phi_m + \lambda_4 \sin(\beta_r + \phi_m) - \lambda_5 \cos(\beta_r + \phi_m) \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\phi}_m} \right) - \frac{\partial L}{\partial \phi_m} = -\lambda_1 \frac{d_m}{2} + \lambda_2 \frac{d_m}{2} - \lambda_4 l_r \sin(\beta_r) + \lambda_5 (l_r \cos \beta_r - d_r) \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\phi}_r} \right) - \frac{\partial L}{\partial \phi_r} = -\lambda_4 r_r \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\beta}_r} \right) - \frac{\partial L}{\partial \beta_r} = -\lambda_5 d_r \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\phi}_L} \right) - \frac{\partial L}{\partial \phi_L} = -\lambda_4 r_f + \tau_L + \frac{\partial x^T}{\partial \phi_L} F_{ext} \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\phi}_R} \right) - \frac{\partial L}{\partial \phi_R} = -\lambda_2 r_f + \tau_R + \frac{\partial x^T}{\partial \phi_R} F_{ext} \\ \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = \tau_i + \frac{\partial x^T}{\partial q_i} F_{ext}, \quad (i=1, \dots, n) \end{cases} \quad (29)$$

where $\lambda_i (i=1, \dots, 5)$ are Lagrange multipliers associated with the nonholonomic constraints, and $F_{ext} \in \mathfrak{R}^6$ is a vector for external forces; τ_L, τ_R, τ_i represent driving torques of the left wheel, the right wheel and the i^{th} joint for the onboard modular manipulator respectively.

Equation 29 can be written into the matrix form

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\zeta}} \right)^T - \left(\frac{\partial L}{\partial \zeta} \right)^T = B(\zeta) \cdot (\tau + J^T \cdot F_{ext}) + C(\zeta) \cdot \lambda \quad (30)$$

where $B(\zeta) = \begin{bmatrix} 0_{(n+2) \times 5} & I_{(n+2) \times (n+2)} \end{bmatrix}^T$, $C(\zeta) = \begin{bmatrix} A(\xi) & 0_{5 \times n} \end{bmatrix}^T$, $\tau = [\tau_L \quad \tau_R \quad \tau_1 \quad \cdots \quad \tau_n]^T$, $\lambda = [\lambda_1 \quad \cdots \quad \lambda_5]^T$.

Equation 30 can be rewritten into the following standard form

$$M(\zeta) \cdot \ddot{\zeta} + V(\zeta, \dot{\zeta}) \cdot \dot{\zeta} + G(\zeta) = B(\zeta) \cdot (\tau + J^T \cdot F_{ext}) + C(\zeta) \cdot \lambda \quad (31)$$

where M is the inertial matrix; V represents the centrifugal and coriolis matrix; and G denotes the gravitational force vector.

Remark 1: Matrix M is symmetric and positive semi-definite, i.e. $M^T = M \geq 0$.

Remark 2: Matrix $\dot{M} - 2V$ is skew-symmetric, i.e., for any $r \in \mathfrak{R}^{n+7}$, $r^T \cdot (\dot{M} - 2V) \cdot r = 0$.
From Eq. 20,

$$\dot{\zeta} = \begin{bmatrix} S(\xi) & 0_{7 \times n} \\ 0_{n \times 2} & I_{n \times n} \end{bmatrix} \cdot \dot{q} \quad (32)$$

Substituting Eq. 32 and its derivative into Eq. 31 yields

$$\bar{M}(\zeta) \cdot \ddot{q} + \bar{V}(\zeta, \dot{q}) \cdot \dot{q} + \bar{G}(\zeta) = \bar{\tau} \quad (33)$$

where $\bar{M} = \bar{S}^T \cdot M \cdot \bar{S}$, $\bar{V} = \bar{S}^T \cdot (V \cdot \bar{S} + M \cdot \dot{\bar{S}})$, $\bar{G} = \bar{S}^T \cdot G$, $\bar{\tau} = \tau + J^T \cdot F_{ext}$; the terms $\bar{S}^T \cdot B = I_{(n+2) \times (n+2)}$ and $\bar{S}^T \cdot C \cdot \lambda = 0_{n+2}$ are eliminated.

If $n < 4$, there are some dead zones that the end-effector can not reach; on the other hand, if $n > 4$ the entire mobile modular manipulator will be a redundant one; this two cases are beyond the discussion of this chapter. Assume the Jacobian matrix is full rank, solving the differential kinematics $\dot{x} = J \cdot \dot{q}$ and its derivative, yields

$$\dot{q} = J^{-1} \cdot \dot{x}, \quad \ddot{q} = J^{-1} \cdot \ddot{x} - J^{-1} \cdot \dot{J} \cdot J^{-1} \cdot \dot{x} \quad (34)$$

Substituting Eq. 34 into Eq. 33 and left multiplying J^{-T} yields

$$\tilde{M}(\zeta) \cdot \ddot{x} + \tilde{V}(\zeta, \dot{x}) \cdot \dot{x} + \tilde{G}(\zeta) = \tilde{\tau} \quad (35)$$

where $\tilde{M} = J^{-T} \cdot \bar{M} \cdot J^{-1}$, $\tilde{V} = J^{-T} \cdot (\bar{V} - \bar{M} \cdot J^{-1} \cdot \dot{J}) \cdot J^{-1}$, $\tilde{G} = J^{-T} \cdot \bar{G}$ and $\tilde{\tau} = J^{-T} \cdot \bar{\tau}$.

Remark 3: Matrix \tilde{M} is symmetric and positive definite, i.e., $\tilde{M}^T = \tilde{M} > 0$.

Proof: From Eq. 33 and Eq. 35,

$$\tilde{M} = J^{-T} \cdot \bar{S}^T \cdot M \cdot \bar{S} \cdot J^{-1} \quad (36)$$

According to **Remark 1**,

$$\begin{aligned} \tilde{M}^T &= J^{-T} \cdot \bar{S}^T \cdot M \cdot \bar{S} \cdot J^{-1} = J^{-T} \cdot \bar{S}^T \cdot M \cdot \bar{S} \cdot J^{-1} = \tilde{M} \\ \tilde{M} &= J^{-T} \cdot \bar{S}^T \cdot M \cdot \bar{S} \cdot J^{-1} = (\bar{S} \cdot J^{-1})^T \cdot M \cdot (\bar{S} \cdot J^{-1}) > 0 \end{aligned} \quad (37)$$

Remark 4: Matrix $\dot{\tilde{M}} - 2\tilde{V}$ is skew-symmetric, i.e., for any $r \in \mathfrak{R}^{n+7}$, $r^T \cdot (\dot{\tilde{M}} - 2\tilde{V}) \cdot r = 0$.

Proof: From Eq. 33 and Eq. 35,

$$J^T \cdot \tilde{M} \cdot J = \bar{S}^T \cdot M \cdot \bar{S} \quad (38)$$

Differentiating Eq. 38 yields

$$J^T \cdot \dot{\tilde{M}} \cdot J + 2J^T \cdot \tilde{M} \cdot \dot{J} = \bar{S}^T \cdot \dot{M} \cdot \bar{S} + 2\bar{S}^T \cdot M \cdot \dot{\bar{S}} \quad (39)$$

Then

$$\dot{\tilde{M}} = J^{-T} \cdot \bar{S}^T \cdot \dot{M} \cdot \bar{S} \cdot J^{-1} + 2J^{-T} \cdot \bar{S}^T \cdot M \cdot \dot{\bar{S}} \cdot J^{-1} - 2\tilde{M} \cdot \dot{J} \cdot J^{-1} \quad (40)$$

From Eq. 33 and Eq. 35

$$\tilde{V} = J^{-T} \cdot \bar{S}^T \cdot V \cdot \bar{S} \cdot J^{-1} + J^{-T} \cdot \bar{S}^T \cdot M \cdot \dot{\bar{S}} \cdot J^{-1} - \tilde{M} \cdot \dot{J} \cdot J^{-1} \quad (41)$$

Subtracting Eq. 41 multiplied by 2 from Eq. 40 yields

$$\dot{\tilde{M}} - 2\tilde{V} = J^{-T} \cdot \bar{S}^T \cdot (\dot{M} - 2V) \cdot \bar{S} \cdot J^{-1} \quad (42)$$

According to Eq. 41

$$r^T \cdot (\dot{\tilde{M}} - 2\tilde{V}) \cdot r = (\bar{S}^T \cdot J^{-1} \cdot r)^T \cdot (\dot{M} - 2V) \cdot (\bar{S} \cdot J^{-1} \cdot r) = 0 \quad (43)$$

Remark 5: Matrices \tilde{M} , \tilde{V} and \tilde{G} are all bounded as long as J keeps nonsingular.

Remark 6: The dynamic equation in Eq. 35 is linearity in parameters, i.e., for any $\chi \in \mathfrak{R}^6$ independent of the structure parameters Φ , $\tilde{M}\ddot{\chi} + \tilde{V}\dot{\chi} + \tilde{G} = Y(\varsigma, \dot{q}, \ddot{q}, \dot{\chi})\Phi$. Here $Y(\varsigma, \dot{q}, \ddot{q}, \dot{\chi})$ is also independent of Φ .

3. Robust Adaptive Controller Design

3.1 A model-based controller

Assume x_d , \dot{x}_d and \ddot{x}_d are the desired end-effector positions, velocities and accelerations. If precise dynamic parameters can be obtained in advance, the model-based controller can be exploited, as shown in Fig. 4 and Eq. 44.

$$\tau = J^T \cdot \left\{ \tilde{M} \cdot [\ddot{x}_d + K_D \cdot (\dot{x}_d - \dot{x}) + K_P \cdot (x_d - x)] + \tilde{V} \cdot \dot{x} + \tilde{G} - F_{ext} \right\} \quad (44)$$

where $K_P > 0$ and $K_D^T = K_D > 0$ are proportional and differential gain matrices.

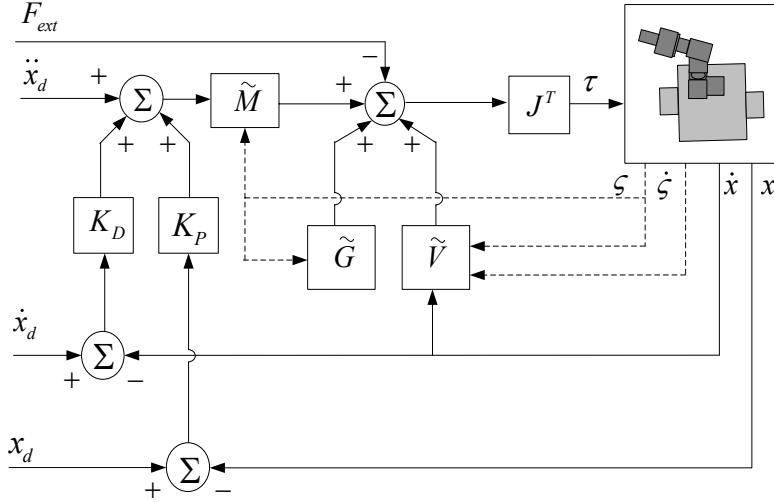


Fig. 4. A model-based controller.

Substituting Eq. 44 into Eq. 35 yields

$$\ddot{e} + K_D \cdot \dot{e} + K_P \cdot e = 0 \quad (45)$$

where $e = x - x_d$ is the error vector.

Theorem 1: If $K_p > 0$ and $K_D^T = K_D > 0$, the system described in Eq. 45 is asymptotically stable, i.e., $e \rightarrow 0$ as $t \rightarrow \infty$.

Proof: Considering the nonnegative Lyapunov candidate as follows

$$V_s = \frac{1}{2} \cdot \dot{e}^T \cdot \dot{e} + \frac{1}{2} \cdot e^T \cdot K_p \cdot e \geq 0 \quad (46)$$

The time derivative of Lyapunov candidate is

$$\dot{V}_s = \dot{e}^T \cdot \ddot{e} + \dot{e}^T \cdot K_p \cdot e \quad (47)$$

Substituting Eq. 45 into Eq. 47 yields

$$\dot{V}_s = -\dot{e}^T \cdot K_D \cdot \dot{e} \leq 0 \quad (48)$$

From Eq. 46 and Eq. 48, V_s is a Lyapunov function. Notice that when and only when $\dot{e} = 0$, $\dot{V}_s = 0$ may hold, so $\dot{e} \rightarrow 0$ as $t \rightarrow +\infty$. To analyze the steady-state error of the system, substituting $\dot{e} = 0$ and $\ddot{e} = 0$ into Eq. 45 yields $K_p \cdot e = 0$, which is followed by $e = 0$. That is to say, $e \rightarrow 0$ as $t \rightarrow +\infty$. That is all for the proof.

3.2 A robust adaptive controller

To avoid measuring accelerations, the error system can be defined as follows

$$\begin{aligned} e(t) &= x(t) - x_d(t) \\ \dot{x}_s(t) &= \dot{x}_d(t) - \Lambda \cdot e(t) \\ s(t) &= \dot{x}(t) - \dot{x}_s(t) \end{aligned} \quad (49)$$

where $s(t)$ is the tracking error measure, and $\Lambda \in \mathfrak{R}^{6 \times 6}$ is a positive definite matrix.

From Eq. 49,

$$\begin{aligned} \dot{x}(t) &= s(t) + \dot{x}_s(t) \\ \ddot{x}_s(t) &= \ddot{x}_d(t) - \Lambda \cdot \dot{e}(t) \\ \ddot{x}(t) &= \dot{s}(t) + \ddot{x}_s(t) \end{aligned} \quad (50)$$

Substituting Eq. 50 into Eq. 35 yields

$$\tilde{M} \cdot \dot{s}(t) + \tilde{V} \cdot s(t) + \tilde{M} \cdot \ddot{x}_s(t) + \tilde{V} \cdot \dot{x}_s(t) + \tilde{G} = \tilde{\tau} \quad (51)$$

From **Remark 6**,

$$\tilde{M} \cdot \ddot{x}_s(t) + \tilde{V} \cdot \dot{x}_s(t) + \tilde{G} = Y(\zeta, \dot{q}_s, \ddot{x}_s, \dot{x}_s) \cdot \Phi \quad (52)$$

Substituting Eq. 52 into Eq. 51 yields

$$\tilde{M} \cdot \dot{s}(t) + \tilde{V} \cdot s(t) + Y(\zeta, \dot{q}_s, \ddot{x}_s, \dot{x}_s) \cdot \Phi = \tilde{\tau} \quad (53)$$

Let $\hat{\Phi}$ be the estimate of the structure parameter Φ , the robust adaptive controller presented in this chapter is given by Eq. 54, and a control system diagram is shown in Fig. 5.

$$\tau = J^T \cdot \left\{ Y(\zeta, \dot{q}_s, \ddot{x}_s, \dot{x}_s) \cdot \hat{\Phi} - K_p \cdot s(t) - K_I \cdot \int_0^t s(t) dt - \tau_r(s) - F_{ext} \right\} \quad (54)$$

where the 1st term forms an adaptive controller; the 2nd and 3rd terms form PI controller; $\tau_r(s)$ is the robust sliding mode term, with its element determined by

$$\tau_{r_i}(s_i) = k_{r_i} \cdot e^{\frac{1}{|s_i|}} \cdot \text{sgn}(s_i) \quad (55)$$

where $k_{r_i} > 0$ is a constant.

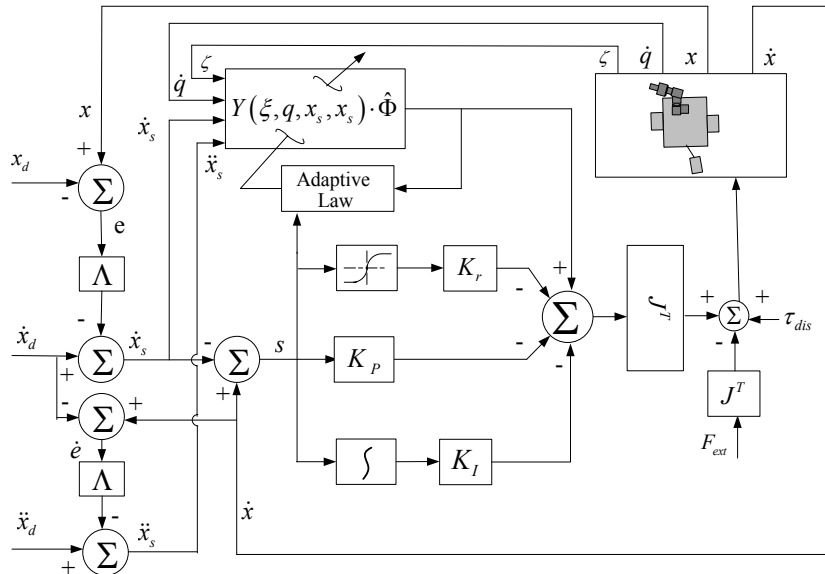


Fig. 5. A robot adaptive controller

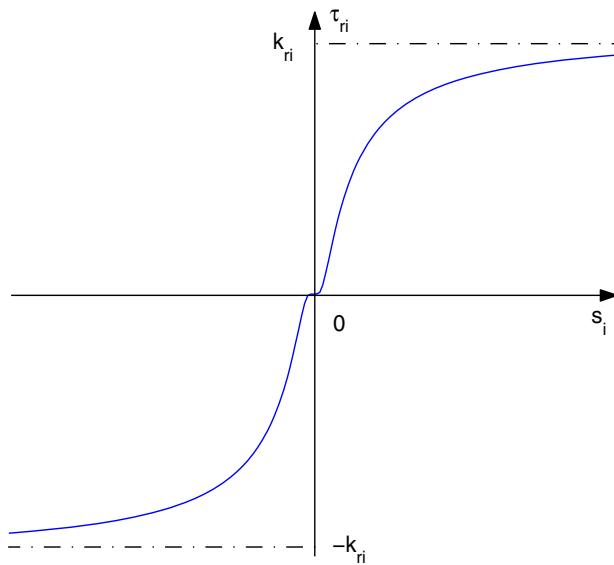


Fig. 6. A new sliding mode function.

The sliding mode control law is not only continuous but also infinitely differentiable, which can eliminate the chattering caused by the classical sign function effectively. The output varies according to $|\mathbf{S}|$. The function is plotted in Fig. 6.

Substituting Eq. 54 into Eq. 53 and considering Eq. 35 at the same time yields

$$\tilde{\mathbf{M}} \cdot \dot{\mathbf{s}}(t) + \tilde{\mathbf{V}} \cdot \mathbf{s}(t) + \mathbf{K}_p \cdot \mathbf{s}(t) + \mathbf{K}_i \cdot \int_0^t \mathbf{s}(t) dt + \mathbf{Y}(\zeta, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_s, \dot{\mathbf{x}}_s) \cdot (\Phi - \hat{\Phi}) + \tau_r = 0 \quad (56)$$

Theorem 2: The closed-loop system in Eq. 56 is asymptotically stable under the adaptation law by Eq. 57. The error signals are convergent with time, i.e., $\mathbf{e}(t), \dot{\mathbf{e}}(t) \rightarrow \mathbf{0}$ as $t \rightarrow \infty$. Furthermore, the signals in the system are all bounded.

$$\dot{\hat{\Phi}} = -\Gamma \cdot \mathbf{Y}^T(\zeta, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_s, \dot{\mathbf{x}}_s) \cdot \mathbf{s} \quad (57)$$

where $\Gamma^T = \Gamma > \mathbf{0}$ is a constant matrix.

Proof: Considering the nonnegative Lyapunov candidate as follows

$$V_s = \frac{1}{2} \cdot \left[\int_0^t \mathbf{s}(t) dt \right]^T \cdot \mathbf{K}_i \cdot \left[\int_0^t \mathbf{s}(t) dt \right] + \frac{1}{2} \cdot \mathbf{s}^T(t) \cdot \tilde{\mathbf{M}} \cdot \mathbf{s}(t) + \frac{1}{2} \cdot \tilde{\Phi}^T \cdot \Gamma^{-1} \cdot \tilde{\Phi} \quad (58)$$

where $\tilde{\Phi} = \hat{\Phi} - \Phi$.

Time derivative of the Lyapunov candidate is

$$\dot{V}_s = \mathbf{s}^T \cdot \left[\tilde{\mathbf{M}} \cdot \dot{\mathbf{s}} + \mathbf{K}_i \cdot \int_0^t \mathbf{s}(t) dt \right] + \frac{1}{2} \cdot \mathbf{s}^T \cdot \dot{\tilde{\mathbf{M}}} \cdot \mathbf{s} + \dot{\tilde{\Phi}}^T \cdot \Gamma^{-1} \cdot \tilde{\Phi} \quad (59)$$

From Eq. 56, we can obtain

$$\mathbf{s}^T \left[\tilde{\mathbf{M}} \cdot \dot{\mathbf{s}} + \mathbf{K}_i \cdot \int_0^t \mathbf{s}(t) dt \right] = -\mathbf{s}^T \cdot \tilde{\mathbf{V}} \cdot \mathbf{s} - \mathbf{s}^T \cdot \mathbf{K}_p \cdot \mathbf{s} + \mathbf{s}^T \cdot \mathbf{Y}(\zeta, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_s, \dot{\mathbf{x}}_s) \tilde{\Phi} - \mathbf{s}^T \tau_r \quad (60)$$

Substituting Eq. 60 into Eq. 59 and considering **Remark 4** at the same time yields

$$\dot{V}_s = -\mathbf{s}^T \cdot \mathbf{K}_p \cdot \mathbf{s} + \mathbf{s}^T \cdot \mathbf{Y}(\zeta, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_s, \dot{\mathbf{x}}_s) \cdot \tilde{\Phi} - \mathbf{s}^T \cdot \tau_r + \dot{\tilde{\Phi}}^T \cdot \Gamma^{-1} \cdot \tilde{\Phi} \quad (61)$$

Considering that $\dot{\tilde{\Phi}} = \dot{\hat{\Phi}}$, substituting Eq. 57 into Eq. 61 yields

$$\dot{V}_s = -\mathbf{s}^T \cdot \mathbf{K}_p \cdot \mathbf{s} - \mathbf{s}^T \cdot \tau_r \quad (62)$$

Substituting Eq. 55 into Eq. 62 yields

$$\dot{V}_s = -\mathbf{s}^T \cdot \mathbf{K}_p \cdot \mathbf{s} - \sum_{i=1}^6 \left\{ k_{r_i} \cdot e^{-\frac{1}{|\mathbf{s}_i|}} \cdot \mathbf{s}_i \cdot \text{sgn}(\mathbf{s}_i) \right\} \leq 0 \quad (63)$$

According to LaSalle's theorem, we can conclude that the system is asymptotically stable and $\mathbf{s} \rightarrow \mathbf{0}$ as $t \rightarrow +\infty$. Let ℓ_p is the p -norm space, from Eq. 58 and Eq. 63, $\mathbf{s}(t) \in \ell_2$. From

Eq. 50, $\mathbf{e}(t) \in \ell_2 \cap \ell_\infty$, $\dot{\mathbf{e}}(t) \in \ell_2$ and $\mathbf{e}(t) \rightarrow 0$ as $t \rightarrow +\infty$. Because $V_s \geq 0$ and $\dot{V}_s \leq 0$, $\tilde{\Phi} \in \ell_\infty$ can be obtained; from **Remark 5**, $\Phi \in \ell_\infty$, so $\hat{\Phi} \in \ell_\infty$; therefore from the error equation, we can conclude that $\mathbf{s}(t) \in \ell_\infty$. Since $\mathbf{s}(t) \in \ell_2$ and $\mathbf{s}(t) \in \ell_\infty$, $\mathbf{s} \rightarrow 0$ as $t \rightarrow +\infty$, which is followed by $\dot{\mathbf{e}}(t) \rightarrow 0$. This is the end of the proof.

4. Simulation Results

The mobile modular manipulator exploited in this simulation is made up of a 4-DOF onboard modular manipulator and 3-wheeled nonholonomic mobile platform, as shown in Fig. 1(a). The modular manipulator consists of 4 rotation modules named *PowerCube* produced by the *AmtecGmbH* Corporation of Germany. The *Pioneer3-DXe* produced by *ActiveMedia* Corporation of USA is used as the mobile platform.

According to the Denavit-Hartenberg notion, the simplified model of the mobile modular manipulator is drawn in Fig. 7.

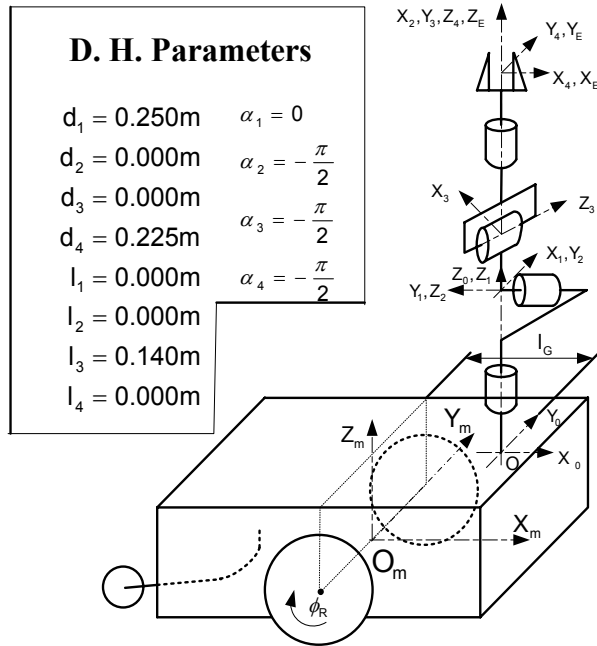


Fig. 7. A simplified model for the mobile modular manipulator.

Because of page limitations, the kinematics and dynamics model will not be detailed. To ensure the controllers valid, the desired end-effector trajectory should be selected as far away from singularities. The Jacobian matrix is derived from Eq. 20 as follows

$$J = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & 0 \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & 0 \\ 0 & 0 & 0 & -l_3 \cos q_2 + d_4 \cos q_2 \sin q_3 & d_4 \sin q_2 \cos q_3 & 0 \\ -\frac{r_f}{d_m} & \frac{r_f}{d_m} & 1 & \frac{\sin q_2 \sin q_3 \cos q_3}{\cos^2 q_3 + \cos^2 q_2 \sin^2 q_3} & \frac{-\cos q_2}{\cos^2 q_3 + \cos^2 q_2 \sin^2 q_3} & 0 \\ 0 & 0 & 0 & \frac{-\cos q_2 \sin q_3}{\sqrt{\cos^2 q_3 + \cos^2 q_2 \sin^2 q_3}} & \frac{-\sin q_2 \cos q_3}{\sqrt{\cos^2 q_3 + \cos^2 q_2 \sin^2 q_3}} & 0 \\ 0 & 0 & 0 & \frac{-\cos q_3}{\cos^2 q_2 + \sin^2 q_2 \cos^2 q_3} & \frac{\sin q_2 \cos q_2 \sin q_3}{\cos^2 q_2 + \sin^2 q_2 \cos^2 q_3} & 1 \end{bmatrix} \quad (64)$$

where

$$\begin{aligned} J_{11} &= \frac{r_f}{2} \cdot \cos \phi_m - \frac{r_f}{d_m} \cdot (J_{13} - l_G \cdot \sin \phi_m), & J_{21} &= \frac{r_f}{2} \cdot \sin \phi_m - \frac{r_f}{d_m} \cdot (J_{23} + l_G \cdot \cos \phi_m), \\ J_{12} &= \frac{r_f}{2} \cdot \cos \phi_m + \frac{r_f}{d_m} \cdot (J_{13} - l_G \cdot \sin \phi_m), & J_{22} &= \frac{r_f}{2} \cdot \sin \phi_m + \frac{r_f}{d_m} \cdot (J_{23} + l_G \cdot \cos \phi_m), \\ J_{13} &= -l_3 \sin(q_1 + \phi_m) \cos q_2 + d_4 \sin(q_1 + \phi_m) \cos q_2 \sin q_3 + d_4 \cos(q_1 + \phi_m) \cos q_3, \\ J_{23} &= +l_3 \cos(q_1 + \phi_m) \cos q_2 - d_4 \cos(q_1 + \phi_m) \cos q_2 \sin q_3 + d_4 \sin(q_1 + \phi_m) \cos q_3, \\ J_{14} &= -l_3 \cdot \cos(q_1 + \phi_m) \cdot \sin q_2 + d_4 \cdot \cos(q_1 + \phi_m) \cdot \sin q_2 \cdot \sin q_3, \\ J_{24} &= -l_3 \cdot \sin(q_1 + \phi_m) \cdot \sin q_2 + d_4 \cdot \sin(q_1 + \phi_m) \cdot \sin q_2 \cdot \sin q_3, \\ J_{15} &= -d_4 \cdot \cos(q_1 + \phi_m) \cdot \cos q_2 \cdot \cos q_3 - d_4 \cdot \sin(q_1 + \phi_m) \cdot \sin q_3, \\ J_{25} &= -d_4 \cdot \sin(q_1 + \phi_m) \cdot \cos q_2 \cdot \cos q_3 + d_4 \cdot \cos(q_1 + \phi_m) \cdot \sin q_3. \end{aligned} \quad (65)$$

The determinant of Jacobian matrix is

$$\det(J) = \frac{r_f^2 \cdot l_G \cdot l_3 \cdot \sin q_2 \cdot \cos q_2 \cdot \cos q_3}{d_m \cdot \sqrt{\cos^2 q_3 + \cos^2 q_2 \cdot \sin^2 q_3}} \quad (66)$$

So, to avoid singularities, the boundary of joint angles in the joint space can be selected as follows

$$\Omega = \left\{ \phi_L, \phi_R, q_1, q_2, q_3, q_4 \mid 0 < q_2 < \frac{\pi}{2}, -\frac{\pi}{2} < q_3 < \frac{\pi}{2} \right\} \quad (67)$$

Simulations are conducted for two control schemes, i.e., the model-based controller (MBC) and the robust adaptive controller (RAC). To examine the disturbance suppression characteristics, a series of disturbance torques are introduced. Both the MBC and the RAC are required to control the end-effector to follow a sine like spatial trajectory as shown in Fig. 8, which has been planned to ensure the system far away from singularities. All the joints and velocities are initialized to be zeros, except that $q_2 = \pi/3$. Simulation time interval is selected as 10 seconds. The gain matrices for the RAC are selected as follows

$$\begin{aligned} K_p &= \text{diag}\{100\}, & K_1 &= \text{diag}\{10.0\}, & K_r &= \text{diag}\{10.0\}, \\ \Lambda &= \text{diag}\{2.0\}, & \Gamma &= \text{diag}\{0.1\}. \end{aligned} \quad (68)$$

For the MBC, nominal dynamic parameters are adopted, which are assumed to be deviated from the real values by 10%; the gain matrices are selected as $K_p = \text{diag}\{100\}, K_D = \text{diag}\{10.0\}$. Simulation results are presented by Fig. 8-14. The desired and controlled locus for the MBC and the RAC are shown in Fig. 8(a) and Fig. 8(b); Figure 9(a) and Fig. 9(b) present the tracking position errors; tracking Euler angular errors are shown in Fig. 10(a) and Fig. 10(b); Figure 11(a) and Fig. 11(b) give the tracking linear velocity errors; tracking Euler angular velocity errors are given by Fig. 12(a) and Fig. 12(b); time-varying control torques for the MBC and the RAC are shown in Fig. 13 and Fig. 14.

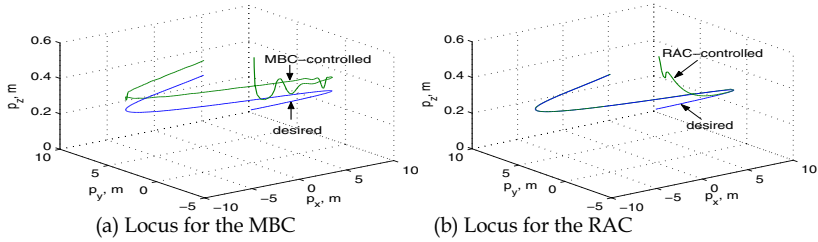


Fig. 8. Desired and controlled locus for the MBC and RAC.

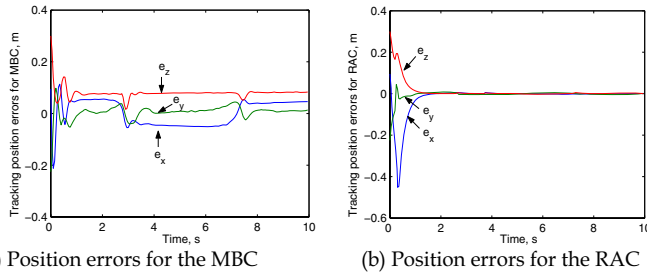


Fig. 9. Tracking position errors for the MBC and RAC.

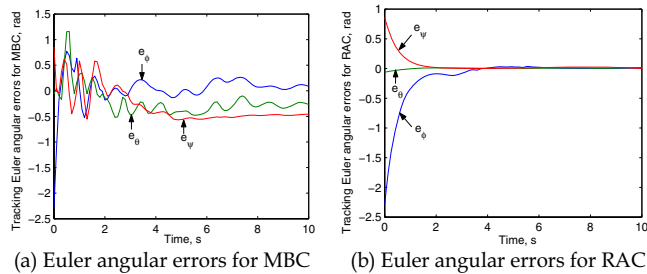


Fig. 10. Tracking Euler angular errors for the MBC and RAC.

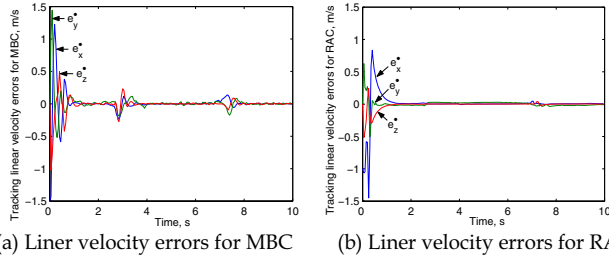


Fig. 11. Tracking linear velocity errors for the MBC and RAC.

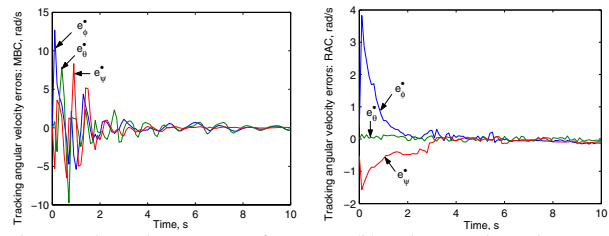


Fig. 12. Tracking Euler angular velocity errors for the MBC and RAC.

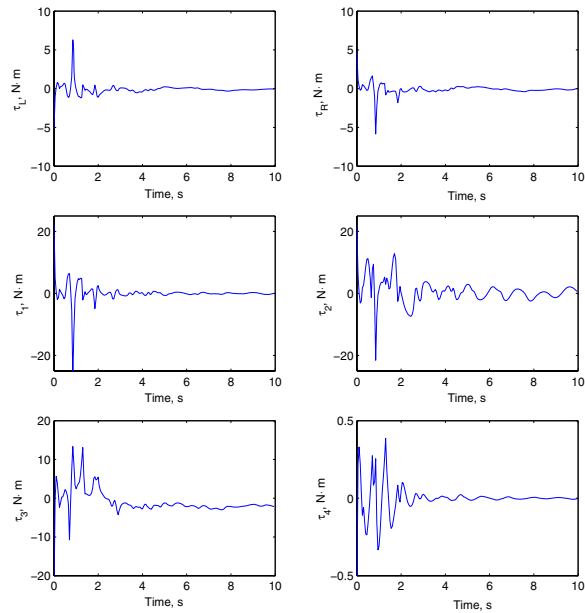


Fig. 13. Time-varying control torques for the MBC.

Remark 7: From the simulation results, we can conclude that the proposed algorithms are effective to control the end-effector to follow some definite spatial trajectories. The robust adaptive controller behaves better than the model based controller, especially when parameter uncertainties and external disturbances exist.

5. Conclusions

In this chapter, 3-wheeled nonholonomic mobile modular manipulators are investigated. First, an integrated modelling method is proposed in consideration of the nonholonomic constraints and the interactive motions. Second, a model based controller and a robust adaptive controller are presented in task space directly. Third, simulations are carried out on a mobile modular manipulator composed of a 3-wheeled mobile platform and a 4-DOF onboard modular manipulator, and the simulation results demonstrate the effectiveness of the proposed algorithms. The proposed algorithms can be easily extended to some other mobile manipulators as well.

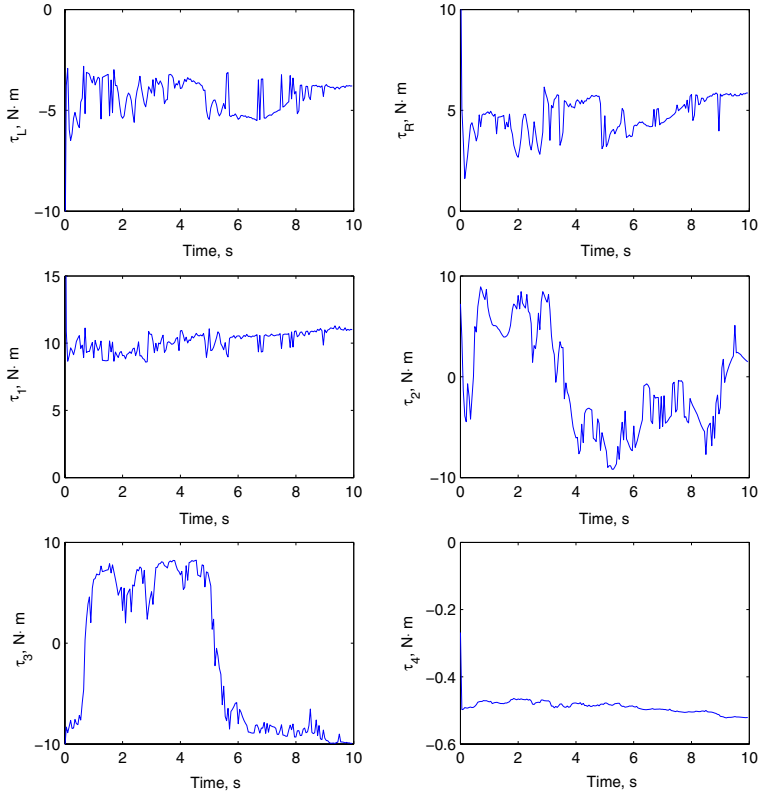


Fig. 14. Time-varying control torques for the RAC.

6. Future Research Work

In this chapter, although the number of joints for the onboard modular manipulator is just assumed to be 4, the proposed method can be applied to m-DOF manipulator atop n-DOF mobile platform system in principle. The modelling and control for a redundant mobile modular manipulator is more challenging since self-motions have to be taken into consideration, (Li & Liu, 2006b; Maciejewski & Klein, 1989). The proposed algorithms are just verified by simulations, there is still a lot of work to do to verify the algorithms by real experiments. The task space information in this simulation is calculated via forward kinematics and differential kinematics, how to acquire task-space information directly and precisely is also a complex work, which needs deep research work on sensory technology and method.

7. Acknowledgement

The authors appreciate the fund support from the research committee of University of Macau under grant no.: RG082/04-05S/LYM/FST.

8. References

- Campos, M.; Pereira, G.; Vale, S.; Bracarense, A.; Pinheiro, G. & Oliveira, M. (2002). A mobile manipulator for installation and removal of aircraft warning spheres on aerial power transmission lines. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3559-3564, Washington, DC, USA.
- Carriker, W.; Khosla, P. & Krogh, B. (1991). Path planning for mobile manipulators for multiple task execution. *IEEE Trans. on Robotics and Automation*, Vol. 7, pp. 403-408.
- Chen, I. & Yang, G. (1998). Inverse kinematics for modular reconfigurable robots. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1647-1652, Leuven, Belgium.
- Chitta, S. & Ostrowski, J. (2002). Motion planning for heterogeneous modular mobile systems. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 4077-4082, Washington, DC, USA.
- Chung, J. & Velinsky, S. (1999). Robust control of a mobile manipulator-dynamic modelling approach. *Proceedings of the American Control Conference*, pp. 2435-2439, San Diego, California, USA.
- Colbaugh, R. (1998). Adaptive stabilization of mobile manipulators. *Proceedings of the American Control Conference*, pp. 1-5, Philadelphia, Pennsylvania, USA.
- de Wit, C.; Siciliano, B. & Bastin, G. (1996). *Theory of Robot Control*, Springer-Verlag.
- Fei, Y.; Zhao, X. & Song, L. (2001). A method for modular robots generating dynamics automatically. *Robotica*, Vol. 19, pp. 59-66.
- Fukuda, T.; Nakagawa, S.; Kawauchi, Y. & Buss, M. (1989). Structure decision method for self organizing robots based on cell structures-cebot. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 695-700, Scottsdale, AZ, USA.
- Furuno, S.; Yamamoto, M. & Mohri, A. (2003). Trajectory planning of mobile manipulator with stability considerations. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3403-3408, Taipei, Taiwan, China.

- Ge, S.; Hang, C. & Woon, L. (1997). Adaptive neural network control of robot manipulators in task space. *IEEE Trans. on Industrial Electronics*, Vol. 44, No. 6, pp. 746-752.
- Ghasemipoor, A. & Sepehri, N. (1995). A measure of machine stability for moving base manipulators. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2249-2254, Nagoya, Aichi, Japan.
- Jamisola, R.; Ang, M.; Oetomo, D.; Khatib, O.; Lim, T. & Lim, S. (2002). The operational space formulation implementation to aircraft canopy polishing using a mobile manipulator. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 400-405, Washington, DC, USA.
- Li, Y.; Liu, Y.; Liu, X. & Peng, Z. (2004a). Parameter identification and vibration control in modular manipulators. *IEEE/ASME Trans. on Mechatronics*, Vol. 9, No. 4, pp. 700-705.
- Li, Y. & Liu, Y. (2004b). Control of a mobile modular manipulator moving on a slope. *Proceedings of IEEE International Conference on Mechatronics*, pp. 135-140, Istanbul, Turkey.
- Liu, Y. & Li, Y. (2005a). Dynamics and model-based control for mobile modular manipulators. *Robotica*, Vol. 23, No. 6, pp. 795-797.
- Li, Y. & Liu, Y. (2005b). Kinematics and tip-over stability analysis for the mobile modular manipulator. *Proceedings of the Institution of Mechanical Engineers Part C: Journal of Mechanical Engineering Science*, Vol. 219, No. 3, pp. 331-343.
- Li, Y.; Liu, Y. & Yan, S. (2005c). Adaptive Neural-Network Control for Redundant Nonholonomic Mobile Modular Manipulators, *Advances in Neural Networks*, Eds. by J. Wang, X. Liao and Z. Yi, Springer, LNCS 3498, Part III, pp.271-276.
- Li, Y. & Liu, Y. (2005d). Obstacle Avoidance for Redundant Nonholonomic Mobile Modular Manipulators via Neural Fuzzy Approaches. *Advances in Natural Computation*, Eds by L. Wang, K. Chen and Y.S. Ong, Springer, LNCS 3612, pp.1109-1118.
- Li, Y. & Liu, Y. (2006a). Online fuzzy logic control for tip-over avoidance of autonomous redundant mobile manipulators. *International Journal of Vehicle Autonomous Systems*, Vol.4, No.1, pp.24-43, 2006.
- Li, Y. & Liu, Y. (2006b). Real-time tip-over prevention and path following control for redundant nonholonomic mobile modular manipulators via fuzzy and neural-fuzzy approaches. *ASME Trans. of Dynamic Systems, Measurement, and Control*, (to appear).
- Maciejewski, A. A. & Klein, C. (1989). The Singular Value Decomposition: Computation and Application to Robotics. *International Journal of Robotics Research*, Vol. 8, No. 6, pp. 63-79.
- Melek, W. & Goldenberg, A. (2003). Neurofuzzy control of modular and reconfigurable robots. *IEEE/ASME Trans. on Mechatronics*, Vol. 8, No. 3, pp. 381-389.
- Nagatani, K.; Hirayama, T.; Gofuku, A. & Tanaka, Y. (2002). Motion planning for mobile manipulator with keeping manipulability. *Proceedings of IEEE/RSJ International Conference on Intelligent Robotics and Systems*, pp. 1663-1668, EPFL, Switzerland.
- Papadopoulos, E. & Rey, D. (1996). A new measure of tipover stability margin for mobile manipulators. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3111-3116, Minneapolis, MN, USA.
- Rus, D.; Donald, B. & Jennings, J. (1995). Moving furniture with teams of autonomous robots. *Proceedings of IEEE/RSJ International Conference on Intelligent Robotics and Systems*, pp. 235-242, Pittsburgh, USA.

- Shen, W.; Salemi, B. & Will, P. (2002). Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots. *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 5, pp. 700-712.
- Stilwell, D. & Bay, J. (1993). Toward the development of a material transport system using swarms of ant-like robots. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 766-771, Atlanta, Georgia, USA.
- Stoy, K.; Shen, W. & Will, P. (2002). Using role-based control to produce locomotion in chain-type self-reconfigurable robots. *IEEE/ASME Trans. on Mechatronics*, Vol. 7, No. 4, pp. 410-417.
- Tanner, H. & Kyriakopoulos, K. (2001). Mobile manipulator modelling with Kane's approach. *Robotica*, Vol. 19, pp. 675-690.
- Tomita, T.; Murata, S.; Kurokawa, H.; Yoshida, E. & Kokaji, S. (1999). Self-assembly self-repair method for a distributed mechanical system. *IEEE Trans. on Robotics and Automation*, Vol. 15, No. 6, pp. 1035-1045.
- Will, P. & Grossman, D. (1975). An experimental system for computer controlled mechanical assembly. *IEEE Trans. on Computer*, Vol. 24, pp. 879-888.
- Yamamoto, Y. & Yun, X. (1996). Effect of the dynamic interaction on coordinated control of mobile manipulators. *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 5, pp. 816-824.
- Yu, Q. & Chen, I. (2002). A general approach to the dynamics of nonholonomic mobile manipulator systems. *ASME Trans. of Dynamic Systems, Measurement, and Control*, Vol. 124, No. 4, pp. 512-521.

Combined Torque and Velocity Control of a Redundant Robot System

Damir Omrčen, Leon Žlajpah, Bojan Nemec
*Jožef Stefan Institute
Slovenia*

1. Introduction

One of the most important features of the next generation of robots is the mobility and the ability to work in an unstructured environment. Conventional manipulator arms have bounded workspace and are thus appropriate only for some limited tasks. Hence, they need additional devices such as conveyor belts and handling devices in order to perform a task requiring large workspace. A new promising concept is to place a manipulator arm on a mobile platform. Such system is usually called mobile manipulator. This adds extra degrees of freedom to the system, which makes the mobile manipulator more dexterous and it may become redundant. Redundancy is an important feature of the new generation of robots making them more versatile. For example, a redundant robot can avoid obstacles in the workspace while executing given task or it can optimize joint torques without modifying primary task, which is usually position and/or force tracking of the robot tool mounted on the top of the robot arm.

Mobile manipulators typically consist of a robot manipulator mounted on a mobile platform. Typically, the workspace of the fixed base manipulators is limited but they have good accuracy and fast dynamics. On the contrary, mobile platforms have an "infinite" workspace but they are slow and inaccurate, they can not perform any task by itself. Integration of a fast and accurate manipulator and a platform results in a mobile manipulator that should integrate good properties of both subsystems. Hence, the mobile manipulator has large workspace and high dynamics. Its accuracy is comparable to the fixed manipulator accuracy using appropriate sensors and appropriate control algorithm.

Due to the remarkable properties mobile manipulators are often used as service robots that help humans in everyday jobs in everyday environments. When working in unknown environment robot autonomy is crucial. The key feature to assure autonomy is the obstacle avoidance. Therefore, appropriate sensors are needed.

Compliant behavior is essential when a robot is in contact with the environment (Asada and Slotine [1986]), e.g. in robot assembling, grinding, driving a screw etc. The usual approaches to compliant motion control are the impedance control (Hogan [1985], Salisbury [1980]), the dynamic hybrid control (Raibert and Craig [1981]) and the (resolved) acceleration control (Luh et al. [1980]). The impedance control does not control forces or positions directly, but it controls the desired dynamics of the end-effector or its stiffness. When the dynamic hybrid control is used, the position is controlled along selected directions, while forces are

controlled along the others directions. Using the resolved acceleration control the force is not controlled directly although we can achieve compliant behavior. The goal of our research is to develop appropriate control algorithm that enables compliant motion of a redundant mobile manipulator in unstructured environments.

A lot of research has been done recently on the control of mobile manipulators (Khatib [1999], Yamamoto and Yun [1996], Altafini [2001], Petersson et al. [2000], Oropeza and Devy [1999], Omrčen et. al. [2004]). Regarding control signals there are two basic types of robot control: velocity and torque control. In the case of a velocity control the control signals are velocities in the robot joints and in the other case control signals are joint torques. Generally, from the performance point of view the torque control has a lot of advantages over the velocity control. However, the torque control can be used only if the robot is equipped with a corresponding motor controller. In fact, most commercially available mobile platforms are velocity controlled and on the other hand manipulators usually enable torque or velocity control. To solve the control problem usually the velocity control is used for both subsystems (a manipulator and a platform). This choice could result in decreased performances of the overall system. To overcome this problem a modification of the motor controller hardware is possible (Holmberg and Khatib [2000]). This is expensive and requires a lot of development. To preserve the advantages of the torque control, while using the existing velocity motor controller, we propose to use a *combined control* ((Omrčen et al. [2004])) where the velocity and the torque control are combined in a single control system.

We have examined the combined control in details analytically and experimentally on a real and a simulated system. The analysis includes tracking the trajectory reference in the task and null space is shown. Furthermore, the influence of disturbing external forces in the task and null space on the motion of the robot system is also analyzed. Similar analysis for a redundant torque controlled system has been done by Žlajpah (Žlajpah [1999]). He examined in details the influence of the external forces in task and null space in different configurations.

Khatib (Khatib [1995]) proved that a redundant robotic system is dynamically consistent in case of inertia weighted generalized inverse. This is true only for complete torque controlled systems. In case of combined controlled system this is not a case. We will show later, that such system can also be dynamically consistent while using other weight. In this case the system is partly decomposed, which is not desired.

We will show on a real system that using appropriate controller gains we can achieve high compliance of the robot system in the null space and high stiffness in the task space. This enables obstacle avoidance without use of any sensors and successful trajectory tracking. Since the robot is redundant, the obstacle avoidance can be done simultaneously with the primary task. Compliance is also an important property when the robot works in the contact with the environment. We have shown that using combined control we could maintain most of good properties of complete torque control without modification of the controller in the velocity controlled subsystem.

2 Methods

2.1 Mobile manipulator

The mobile manipulator that we used (Fig. 1), consist of a torque controlled planar manipulator with four DOF mounted on a velocity controlled holonomic platform Nomad XR4000 with three DOF. The complete system has seven DOF ($n = 7$). The task was defined as a positioning in (x - y) plane. So, the task has two DOF ($m = 2$) and consequently, the

degree of redundancy is five ($r = 5$). The planar manipulator is back-drivable and it was primary designed to be back-drivable and high speed and we sacrificed the accuracy. In general, advanced force and trajectory control relies on controlling motors torques directly. Unfortunately, mobile platform which we used enables only position and velocity control. Having in mind this limitation, we propose a combination of velocity controlled platform and torque controlled manipulator.



Fig. 1. Mobile manipulator.

First we have to set the mathematical model of the mobile manipulator. Using the Lagrangean formulation, we can express the dynamic model of the complete system in the form

$$H(q)\ddot{q} + C(\dot{q}, q)\dot{q} + g(q) = \tau, \quad (1)$$

here, H , C in g denote the inertia matrix, the vector of Coriolis and centrifugal forces and the vector of gravity forces, respectively. τ is the vector of joint torques and q is the vector of joint positions. As one part of the system is controlled by torque and the other part by velocity, the dynamic model (1) is decomposed into

$$\begin{bmatrix} H_v(q) & H_{vt}(q) \\ H_{tv}(q) & H_t(q) \end{bmatrix} \begin{bmatrix} \ddot{q}_v \\ \ddot{q}_t \end{bmatrix} + \begin{bmatrix} C_v(\dot{q}, q) & C_{vt}(\dot{q}, q) \\ C_{tv}(\dot{q}, q) & C_t(\dot{q}, q) \end{bmatrix} \begin{bmatrix} \dot{q}_v \\ \dot{q}_t \end{bmatrix} + \begin{bmatrix} g_v(q) \\ g_t(q) \end{bmatrix} = \begin{bmatrix} \tau_v \\ \tau_t \end{bmatrix}, \quad (2)$$

where $(\cdot)_v$ indicates the velocity controlled subsystem (platform) and $(\cdot)_t$ the torque controlled subsystem (manipulator). We assume that the velocity controlled subsystem is lower in kinematic chain and the torque controlled subsystem is higher. This is true for most of the combined controlled system e.g. most mobile manipulators. In other cases the controller can be easily modified.

Since the subsystems are physically connected, the dynamic interaction between both subsystem exist and it is included in H_{tv} , H_{vt} , H_v , C_v ,...

2.2 Combined control of the mobile manipulator

The combined control is based on acceleration control (Luh et al. [1980], Hsu et al. [1989]):

$$\tau_c = \hat{H}(q)\ddot{q}_c + \hat{C}(\dot{q}, q)\dot{q} + \hat{g}(q), \quad (3)$$

$$\ddot{\mathbf{q}}_c = \mathbf{J}^\#(\ddot{\mathbf{x}}_c - \dot{\mathbf{J}}\dot{\mathbf{q}}) + \mathbf{N}(\Phi - \dot{\mathbf{N}}\dot{\mathbf{q}}), \quad (4)$$

here the motion in the task space and in the null space can be controlled independently. τ_c is the control torque and $\ddot{\mathbf{q}}_c$ is the control acceleration. (\approx) indicates the approximation of dynamic model matrices (1). $\mathbf{J}^\#$ and \mathbf{N}^\top are the generalized inverse of the end-effector Jacobian matrix \mathbf{J} and the projection into the null space of \mathbf{J}^\top , respectively, and are defined by

$$\mathbf{J}^\# = \mathbf{W}^{-1}\mathbf{J}^\top(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^\top)^{-1}, \quad (5)$$

$$\mathbf{N} = \mathbf{I} - \mathbf{J}^\#\mathbf{J}. \quad (6)$$

Here \mathbf{W} is the generalized inverse weight. $\ddot{\mathbf{x}}_c$ and Φ in Eq. (4) represent the task space and null space control law, respectively, and are defined by:

$$\ddot{\mathbf{x}}_c = \ddot{\mathbf{r}} + \mathbf{K}_d\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e}, \quad \mathbf{e} = \mathbf{r} - \mathbf{x}, \quad (7)$$

$$\Phi = \mathbf{N}\ddot{\phi} + \dot{\mathbf{N}}\dot{\phi} + \mathbf{K}_n\dot{\mathbf{e}}_n, \quad \dot{\mathbf{e}}_n = \mathbf{N}(\dot{\phi} - \dot{\mathbf{q}}), \quad (8)$$

where \mathbf{e} is the task space tracking error, \mathbf{x} and \mathbf{r} are the actual and the desired task space position, respectively. ϕ is the desired null space velocity. \mathbf{K}_d and \mathbf{K}_p are constant gain matrices that define the task space impedance and \mathbf{K}_n is a constant gain matrix that defines the null space damping.

We assume that the torques of the first subsystem in kinematic chain can not be controlled, since the subsystem is controlled by a velocity controller. Therefore, we can not apply the controller (3) directly. Instead of modifying the velocity motor controller in the platform we propose the following controller, which is shown in Fig. 2:

$$\tau_{v_c} = \mathbf{f}(\dot{\mathbf{q}}_{v_c}), \quad (9)$$

$$\tau_{t_c} = [\hat{\mathbf{H}}_{v_c}(\mathbf{q}) \quad \hat{\mathbf{H}}_{t_c}(\mathbf{q})] \begin{bmatrix} \ddot{\mathbf{q}}_{v_c} \\ \ddot{\mathbf{q}}_{t_c} \end{bmatrix} + [\hat{\mathbf{C}}_{v_c}(\mathbf{q}, \dot{\mathbf{q}}) \quad \hat{\mathbf{C}}_{t_c}(\mathbf{q}, \dot{\mathbf{q}})] \begin{bmatrix} \dot{\mathbf{q}}_{v_c} \\ \dot{\mathbf{q}}_{t_c} \end{bmatrix} + \hat{\mathbf{g}}_t(\mathbf{q}). \quad (10)$$

The control accelerations $\ddot{\mathbf{q}}_c$ are divided into two parts: velocity and torque part (see Fig. 2). To control a velocity controlled subsystem we have integrated control accelerations and obtain control velocities. These control velocities define inner torques in the velocity controlled subsystem τ_{v_c} by an unknown nonlinear function $\mathbf{f}(\dot{\mathbf{q}}_{v_c})$. This controller considers only the reference velocity $\dot{\mathbf{q}}_{v_c}$. We can presume that the velocity controlled system ensure tracking the velocity reference.

On the other hand the torques of the torque controlled subsystem τ_{t_c} are defined by the user (10). They are obtained using only corresponding part of the controller (3) and includes only a corresponding part of the dynamic model (2). We can see that the motion of the velocity controlled part is completely compensated.

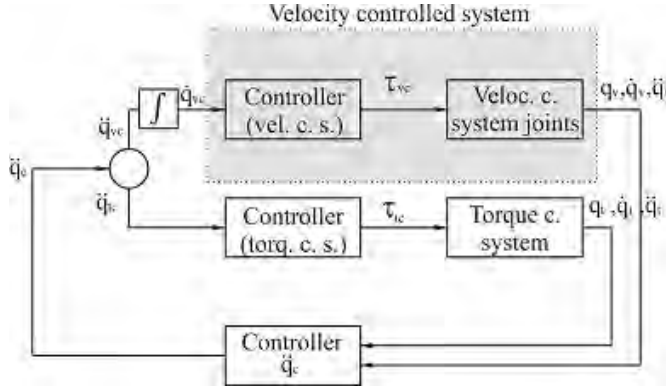


Fig. 2. Combined control scheme.

3. Analysis of the combined controller

This section describes the analysis of the combined control. We examined the trajectory tracking and the influence of the external force.

3.1 Trajectory tracking

One of the basic requirements of the robot system is the ability of tracking the predefined trajectory. First we analyze the trajectory tracking in the presence of external forces/torques, which are considered as disturbance. For the torque controlled part we use the controller defined by Eq. (10). Combining Eqs. (10) and (2) together and considering external torques τ_{ext_t} acting on the torque controlled joints yields:

$$\begin{aligned} & [\hat{H}_v(q) \quad \hat{H}_t(q)] \begin{bmatrix} \ddot{q}_v \\ \ddot{q}_t \end{bmatrix} + [\hat{C}_v(\dot{q}, q) \quad \hat{C}_t(\dot{q}, q)] \begin{bmatrix} \dot{q}_v \\ \dot{q}_t \end{bmatrix} + \hat{g}_t(q) = \\ & = [H_v(q) \quad H_t(q)] \begin{bmatrix} \ddot{q}_v \\ \ddot{q}_t \end{bmatrix} + [C_v(\dot{q}, q) \quad C_t(\dot{q}, q)] \begin{bmatrix} \dot{q}_v \\ \dot{q}_t \end{bmatrix} + g_t(q) + \tau_{ext}. \end{aligned}$$

Assuming that there is no error in the dynamic model of the system i.e. $(\hat{\cdot}) = (\cdot)$ and that the velocity controlled subsystem assures good tracking of the trajectory reference $\ddot{q}_v = \ddot{q}_v$ yields

$$H_t \ddot{q}_t = H_t \ddot{q}_t + \tau_{ext}. \quad (11)$$

Task space analysis

Starting from Eq. (11) both sides of equation are premultiplied by H_t^{-1} and by J_t . Let J_t be the right part of the Jacobian matrix J associated with the torque controlled subsystem and J_v the left part of Jacobian matrix associated with the velocity controlled subsystem, $J = [J_v | J_t]$. Then, by premultiplying (11) by $J_t H_t^{-1}$ yields

$$J_t \ddot{q}_t = J_t \ddot{q}_t + J_t H_t^{-1} \tau_{ext}. \quad (12)$$

Then, by add in the term $J_v \ddot{q}_v$ and after simplification yields

$$J \ddot{q}_c = J \ddot{q} + J_t H_t^{-1} \tau_{ext}. \quad (13)$$

Substituting Eq. (4) for \ddot{q}_c and using $JJ^\# = I$, $JN = 0$ and $\ddot{x} = J\ddot{q} + \dot{J}\dot{q}$ and then using Eq. (7) we derive the final equation that describes dynamics of the error in the task space:

$$\ddot{e} + K_d \dot{e} + K_p e = J_t H_t^{-1} \tau_{ext}. \quad (14)$$

The dynamics of the error is also a function of the robot configuration q , since J_t and H_t are functions of the configuration.

3.1.1 Null space analysis

Starting from the same equation as in the previous case (11) and premultiplying it by H_t^{-1} and N_t . N_t is right part of the matrix N that is associated with the torque controlled subsystem and N_v is left part of the matrix N that is associated with the velocity controlled subsystem $N = [N_v | N_t]$. It yields

$$N_t \ddot{q}_t = N_t \ddot{q}_t + N_t H_t^{-1} \tau_{ext},$$

then we add the term $N_v \ddot{q}_v$ and simplify to

$$N \ddot{q}_c = N \ddot{q} + N_t H_t^{-1} \tau_{ext}. \quad (15)$$

Again substituting Eq. (4) for \ddot{q}_c and using $NJ^\# = 0$, $NN = N$ and $\ddot{q} = J^\#(\ddot{x} - \dot{J}\dot{q}) + N\ddot{q}$ and using Eq. (8) we derive the final equation that describes dynamics of the error in the null space:

$$N(\ddot{e}_n + K_n \dot{e}_n) = N_t H_t^{-1} \tau_{ext}. \quad (16)$$

Using Lyapunov stability theory we can define stability region of the controller parameters. Let us select Lyapunov function

$$v = \frac{1}{2} \dot{e}_n^T W \dot{e}_n$$

and it follows

$$\begin{aligned} \dot{v} &= \dot{e}_n^T W \dot{e}_n + \frac{1}{2} \dot{e}_n^T \dot{W} \dot{e}_n = \\ &= -\dot{e}_n^T W N K_n \dot{e}_n + \dot{e}_n^T W N_t H_t^{-1} \tau_{ext} + \frac{1}{2} \dot{e}_n^T \dot{W} \dot{e}_n = \\ &= -\dot{e}_n^T (W K_n - \frac{1}{2} \dot{W}) \dot{e}_n + \dot{e}_n^T W N_t H_t^{-1} \tau_{ext}, \end{aligned} \quad (17)$$

since $\dot{\mathbf{e}}_n^T = \dot{\mathbf{e}}_n^T \mathbf{N}^T$ and $\mathbf{N}^T \mathbf{W} = \mathbf{W} \mathbf{N}$. If the above requirement ensures negative definiteness of $\dot{\mathbf{V}}$ the system is asymptotically stable. Therefore, the system without the external forces ($\tau_{\text{ext}} = \mathbf{0}$) is asymptotically stable if the matrix is $\mathbf{W} \mathbf{K}_n - \frac{1}{2} \dot{\mathbf{W}}$ is positive definite. This means that to low value of the parameter \mathbf{K}_n destabilize the system (Nemec [1997]).

3.2 External force analysis

3.2.1 Forces acting on the robot end-effector

The external force that acts on the robot end-effector \mathbf{F}_0 produces the torques in all robot joints. Since the velocity controller is stiff to the external forces, the forces do not affect the motion of the velocity controlled subsystem. On the other hand the torques in the torque controlled subsystem affect the motion and have to be analyzed. Torques in the joints of the torque controlled subsystem are defined as:

$$\tau_{\text{ext}} = \mathbf{J}_t^T \mathbf{F}_0. \quad (18)$$

Using Eq. (14) the following nonhomogenous differential equation is derived

$$\ddot{\mathbf{e}} + \mathbf{K}_d \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e} = \mathbf{J}_t \mathbf{H}_t^{-1} \mathbf{J}_t^T \mathbf{F}_0. \quad (19)$$

The equation describes the dynamics of the error in the task space in presence of the external force acting on the robot end-effector.

Žlajpah (Žlajpah [1997]) has shown that the system is stable for bounded external force \mathbf{F}_0 and that error $\dot{\mathbf{e}}$ converges to some limited region.

When setting the controller parameters \mathbf{K}_p and \mathbf{K}_d we have to consider task space error dynamics (14) and also a desired response to an external force (19). Higher value of the parameter \mathbf{K}_p makes the the system stiffer to external force and faster when tracking the trajectory. On the other hand to large value of the \mathbf{K}_p causes oscillation of the system and can cause instability of the real system, because real system often has time delays and dead zones in the controller loop. Higher values of the \mathbf{K}_d increases damping to the external force and slows down the response, but improves the stability of the system.

3.2.2 Dynamic consistency

The system is dynamically consistent when torques in the null space do not produce any accelerations in the task space (Khatib [1995]) and when forces in the task space do not produce any accelerations in the null space (Nemec and Žlajpah [2002]). According to Khatib (Khatib [1995]) the system is dynamically consistent only when inertia weighted generalized inverse is used. Using inertia matrix as weight the kinetic energy of the system is also minimized (Hollerbach [1987]). On the other hand the generalized inverse weight can be any symmetric positive definite matrix of size $\mathbf{n} \times \mathbf{n}$ (Nakamura [1991]), where \mathbf{n} is the number of DOF of the system. Depending on the weight different criterion can be fulfilled. For example, joint velocities of the

system can be minimized or the error of the secondary task can be decreased (Lenarčič [2000]) etc.

Using the inertia weighted inverse the dynamic consistency can be achieved only for the torque controlled robot systems. In contrary, in case of combined controlled systems the dynamic consistency can not be achieved using inertia weighted generalized inverse. In this section we will analyze the influence of the external force on the null space motion and the possibility of the dynamic consistency.

The external force produces torques in robot joints which can affect the null space motion. Using the equations (16) and (18) yields

$$\mathbf{N}(\ddot{\mathbf{e}}_n + \mathbf{K}_n \dot{\mathbf{e}}_n) = \mathbf{N}_t \mathbf{H}_t^{-1} \mathbf{J}_t^T \mathbf{F}_0.$$

The system is dynamically consistent if the external force \mathbf{F}_0 does not affect the null space motion, yielding

$$\mathbf{N}_t \mathbf{H}_t^{-1} \mathbf{J}_t^T = \mathbf{0}. \quad (20)$$

Let us define a selection matrix \mathbf{I}_t

$$\mathbf{I}_t = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}.$$

The lower part of the matrix is the identity matrix \mathbf{I} of the size equal to the number of DOF of the torque controlled system and upper part is the corresponding zero matrix. If we choose the weighting matrix as:

$$\mathbf{W}^{-1} = \mathbf{I}_t \mathbf{H}_t^{-1} \mathbf{I}_t^T \quad (21)$$

and using relations $\mathbf{J}_t = \mathbf{J}_t \mathbf{I}_t$ and $\mathbf{N}_t = \mathbf{N}_t \mathbf{I}_t$ it can be easily shown that using selected weight Eq. (20) is fulfilled. So, using the weight (21) the external force \mathbf{F}_0 does not produce any null space accelerations, and consequently, the system is dynamically consistent.

We could easily see that using selected weight both subsystems are separated, which is not desired.

3.2.3 Forces acting on the robot segments

An external force acting on the robot segments causes torques directly in the null space independent on the generalized inverse weight. The force acting on the torque controlled part moves the segments away from the force, i.e. a robot is compliant in the null space. Consequently, the retreat produces the null space tracking error (16).

The controller gain \mathbf{K}_n in Eq. (8) defines the compliance of the robot to the external force. However, the compliance depends also on the robot configuration and its inertia matrix (16). Higher value of \mathbf{K}_n enables better trajectory tracking in the null space and the robot is stiffer to the external force in the null space. In contrast, lower value of \mathbf{K}_n makes the robot more compliant in the null space and the trajectory tracking is worsened.

We will show later that high compliance in the null space assures obstacle avoidance. For that, low values of \mathbf{K}_n are necessary, which can make the system unstable, i.e. the derivative of the Lyapunov function (17) becomes positive definite.

Note that the external force acting on the robot segments influences also the task space motion and causes the task space error as shown in Eq. (14). Of course, when the external force vanishes the task space error converges to zero.

3.2.4 The force acting on the velocity controlled subsystem

On the other hand, the robot is not compliant to an external force acting on the velocity controlled part. The external force produces only torques in the velocity controlled joints, since the velocity controlled system is first in the kinematic chain. Because the velocity controller is stiff and assures tracking of the trajectory, the external force does not affect the motion of the system.

4. Verification of combined control by simulation

In this section we compare three different approaches to the robot control: a velocity control, a torque control and the proposed combined control. We have made the comparison on a model of a real robot system composed of a torque controlled planar manipulator mounted on a velocity controlled holonomic platform (Fig. 1).

Note that the comparison is only a raw comparison of different approaches, since there exists numerous variants of velocity and torque control. Although, the comparison is good and fair enough to see some basic properties of combined control.

4.1 Controller definitions

First controller is a velocity controller, where the entire system is controlled by a velocity. Each joint is controlled by a PID controller with optimal parameters that assures good joint trajectory tracking. We used the following velocity controller:

$$\dot{q}_c = J^\# (\dot{x}_r + K_p e) + N\dot{\phi}. \quad (23)$$

Second case is a torque controller, where the entire system is torque controlled. We used a resolved acceleration control:

$$\tau_c = \hat{H}(q)\ddot{q}_c + \hat{C}(q, \dot{q})\dot{q} + \hat{g}(q). \quad (24)$$

The third compared controller is the proposed combined controller.

Controller parameters of all three cases are optimal and were defined in such a way that all three systems have similar responses to some reference trajectory (see next section).

4.2 Trajectory tracking

First we analyze the tracking of the end-effector trajectory. The desired trajectory is a circle with radius 1 m. Two different periods was used. For slower motion the period was 1 s and for the faster motion it was 0.1 s.

Fig. 3 and Fig. 4 show the tracking error of all three controllers. For slower motion the tracking errors are approximately the same, because the controller parameters were set for such motion. Here, the motion is slow. Therefore, dynamic interactions between different joints are small and all three controllers assure good trajectory tracking.

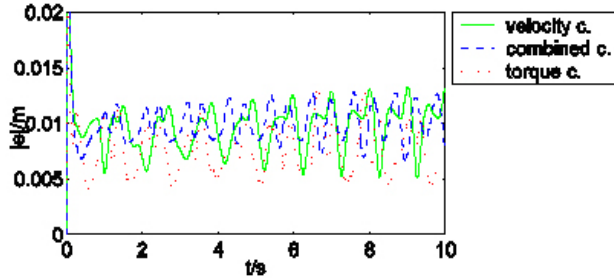


Fig. 3. Task space error in case of slower circular trajectory

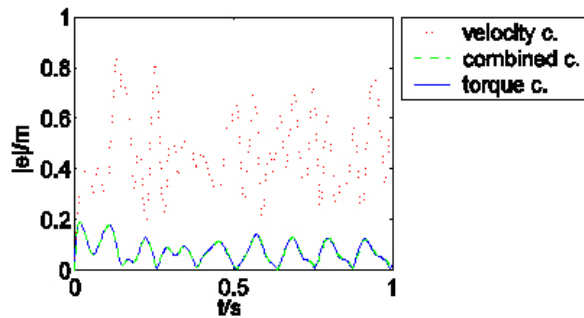


Fig. 4. Task space error in case of faster circular trajectory.

On the other hand, for the faster motion (Fig. 4) the dynamic interactions between joints are larger. Therefore, the tracking error of the velocity controller is much higher than in the case of other two controllers, since the velocity controller does not consider any dynamic interactions. The torque controller considers all dynamic interactions and the combined controller considers all the influence on the manipulator and none on the platform (10). In our case the weight of the platform is significantly higher as the weight of the manipulator and the influence on the platform is low.

4.3 Force acting on the robot end-effector

When the robot end-effector is in contact with the environment a contact force appears. None of the three controllers controls the force directly and none considers this force in the control algorithm. Therefore, the external force in all three controllers is considered as a disturbance.

The contact with the environment occurs when a part of the trajectory is inside of an object. In our experiment the trajectory reference was 5 cm inside of the object in X direction. The stiffness of the object was 5000 N/m. The initial configuration of the robot is shown in Fig. 7. This figure shows the top view of the mobile manipulator. Large circle corresponds to the platform. On top of the platform is the manipulator marked with straight lines. Next figures show the error in task space (Fig. 5) and the force on the end-effector (Fig. 6), which results from the contact.

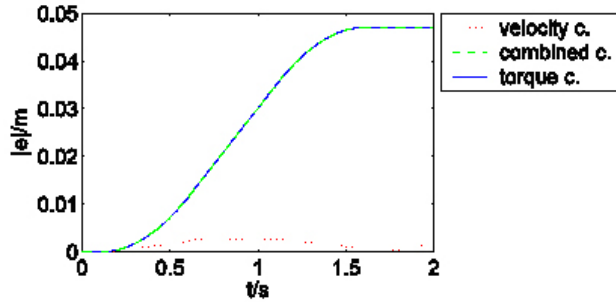


Fig. 5. Error in the task space.

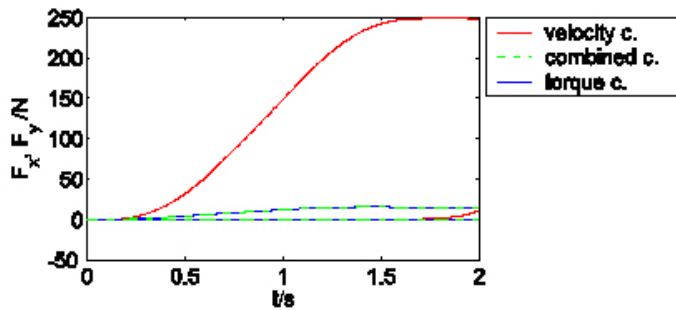


Fig. 6. Force on the end-effector.

The velocity controlled system is usually very stiff to external force, as can be seen in the figures. The error is small, and consequently the force of the contact is large, since the end-effector is “deep” in the object. The stiffness of the system is defined by internal joint velocity controllers that usually have large gains in order to achieve good tracking performance despite disturbances.

In the case of torque controlled system an external force produces torques in the joints. If the system is not dynamically consistent, i.e. the weight is not the inertia matrix, the external force affects the motion in the null space. This often leads to singular configuration of the manipulator. Hence, the dynamic consistency is preferred. Here, the torques in the joints resulting from the external force are completely compensated by the control torques.

Responses of the torque controlled dynamically compensated system are shown in the Fig. 5 and Fig. 6. The external force is significantly lower than with the velocity controller, since the torque controlled system is more compliant to external forces and consequently, task space error is higher. The compliance depends on the controller parameters and the robot configuration. The parameters were set in such a way that high compliance is achieved.

In the case of combined control the response of the system is similar as in the case of the torque controller. Here, the dynamic consistency is not feasible due to a facts described in the section 0. However, using the inertia matrix as a weight, the properties of the system can be quite similar to a dynamically consistent system. The self motion is very low due to the large weight corresponding to the platform and is noticeable after longer time.

4.4 Forces acting on the mobile manipulator segments

4.4.1 Forces acting on the manipulator

In our experiment we apply a force of 10 N on the end of 2-nd segment of the manipulator as shown in Fig. 7.

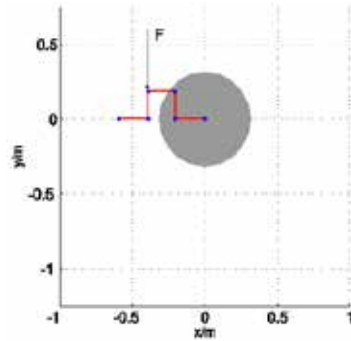


Fig. 7. The external force acting on the manipulator.

In the case of velocity controlled system the external force did not cause any noticeable configuration changes (Fig. 8). Consequently, the task space error was low (Fig. 11). As already mentioned, the velocity controlled system is stiff to an external force, regardless of where a force acts.

In the case of combined controller the external force causes a displacement of the manipulator in the point of the contact (Fig. 9 a)). Consequently, the task space error appears (Fig. 11) and the platform moves in order to decrease the task space error. The configuration after 100 s of acting the force is shown in Fig. 9 b). The platform has moved up and therefore part of the manipulator between point of the contact and a base or end-effector is singular and the manipulator is no more compliant to the external force (for details see Žlajpah [1999]).

We achieve best results with the torque controlled system. The external force causes a displacement of the entire system (including the platform) (Fig. 10). The task space error is approximately the same as with the combined control (Fig. 11).

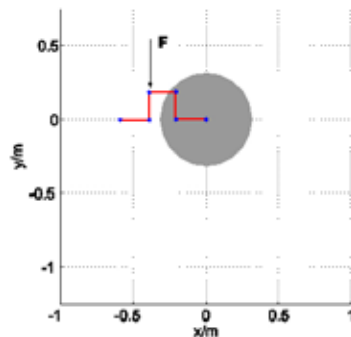
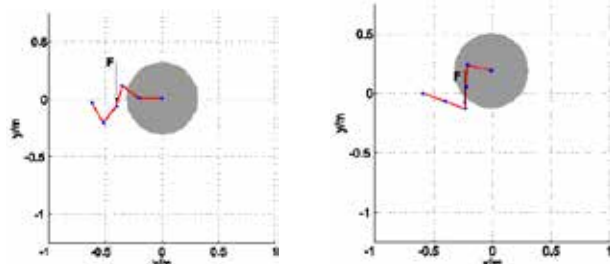
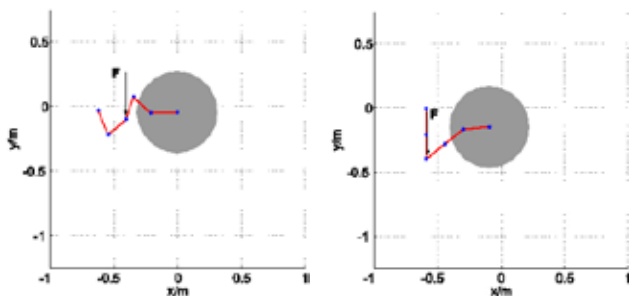


Fig. 8. Configuration of the robot after 2 s of acting the force (velocity control)



a) After 2 s of acting the force b) After 100 s of acting the force
 Fig. 9. Configuration of the robot after acting the force (combined control).



a) After 2 s of acting the force b) After 100 s of acting the force
 Fig. 10. Configuration of the robot after acting the force (torque control).

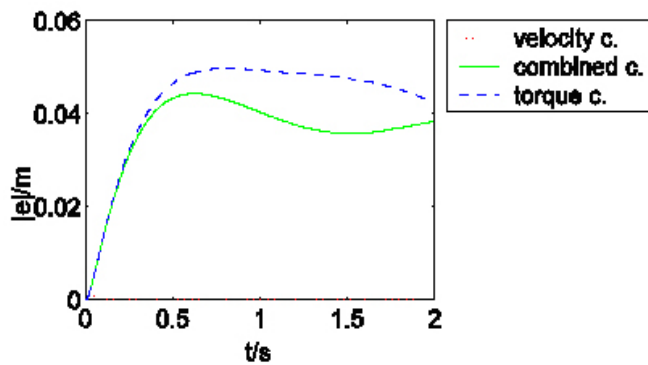


Fig. 11. Task space error.

4.4.2 Forces acting on the platform

We applied a force of 10 N on the platform as shown in Fig. 12. In this case the largest disadvantage of the combined control can be seen. Responses of the velocity and the combined controlled systems are the same, since in both cases the velocity controller assures

stiff response of the platform to an external force acting on the platform. The configuration remains the same in both cases despite of the external force (Fig. 12). The task space error is negligible (Fig. 14).

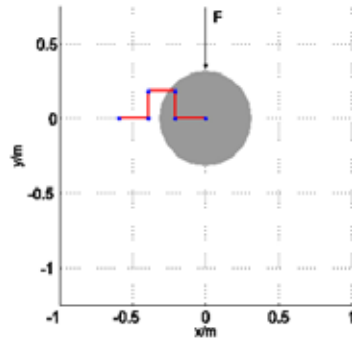


Fig. 12. Point of acting the force on the platform.

In the case of the torque controller the platform moves in the direction away from the force (Fig. 13), similar as the manipulator moved in the previous case. The task space error is consequently higher (Fig. 14).

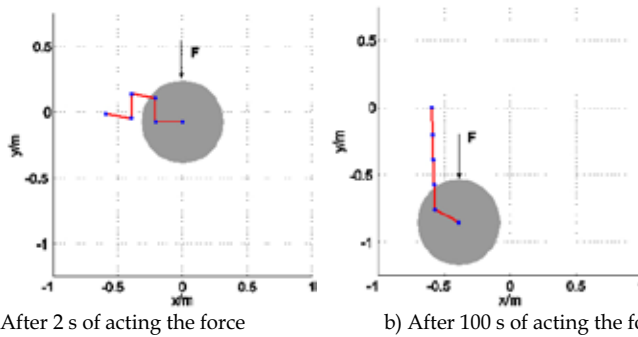


Fig. 13. Configuration of the robot after acting the force on the platform (torque control).

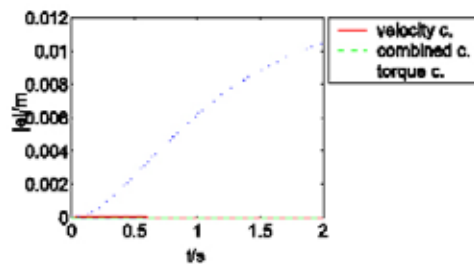


Fig. 14. Task space error.

5. Experiments on real system

5.1.1 Determination of the generalized inverse weight

The drawback of the proposed approach is that the system is not dynamically consistent although using inertia weighted generalized inverse. When using the inertia weighted generalized inverse the kinetic energy of the system is minimized. However, the minimization of the kinetic energy is not desired in our case. Namely, the platform is much heavier (150 kg) than the segments of the manipulator (50 g to 900 g). Because of this large difference in weights, the kinetic energy minimization results in motion, where most of the motion is done by the manipulator. Consequently, the manipulator often comes into singular configuration (manipulator stretches), while the platform is practically not moving. To avoid this problem, we propose to change the generalized inverse weight matrix as follows:

$$W = \begin{bmatrix} H_v(q) & H_w(q) \\ H_w(q) & k * H_t(q) \end{bmatrix}. \quad (17)$$

The value of the parameter k affects only the part corresponding to the kinetic energy of the manipulator used in the minimization. In our case the value of the k was defined empirically and set to 10.

5.2 The task of the mobile manipulator

In our case the mobile manipulator is supposed to move in unknown environment with unknown moving obstacles. The primary task is to track a prescribed end-effector trajectory and the secondary task is the obstacle avoidance.

5.2.1 Obstacle avoidance

For obstacle avoidance certain information of the environment is needed. Many commercially available mobile platforms already have integrated sonar sensors for detection of obstacles around the platform. Based on the proximity measurements the platform is moved away from the obstacles. Like most of the local strategies that solve the obstacle avoidance at the kinematic level, the approach we use is to assign each point, which is close to an obstacle, a motion component in a direction away from the obstacle. The repulsive velocity in the null space corresponding to the translational velocities of the platform $\dot{q}_{n_{sy}}$ can be defined as:

$$\dot{q}_{n_{sy}} = \begin{cases} -\sum_i w_i K_s \left(\frac{d_{soi}}{|d_0|} - 1 \right) \frac{d_0}{|d_0|}, & \forall d_0 < d_{soi} \\ 0, & \text{else} \end{cases}.$$

All obstacles inside the sphere of influence are included in the repulsive velocity. If only the closest obstacle is included in the repulsive velocity the switching between two or more obstacles could occur and consequently, the discontinuity in the repulsive velocity.

Unlike in the case of mobile platforms, robot manipulators usually do not possess integrated sensors for obstacle detection in 3D space. In the past, there were some attempts to integrate some kind of "sensitive skin" and similar sensors along the robot manipulator links (Lumelsky, Siemens tactile sensors, ...), but this solution is very complicated and not suitable for practical applications. Perhaps the most promising are the laser sensors and the 3D vision. As an alternative we propose an approach, where obstacles are actually not avoided,

but the contact forces, which occur after the collision between the manipulator links and obstacles, are minimized. The minimization of contact forces is done by a compliant control of a manipulator in the null space (Eq.(16)). For details see (Omrčen et al. [2004]). The same principle can be noticed by humans working in dark environments.

To achieve obstacle avoidance by compliant control the manipulator has to be torque controlled and back-drivable [Žlajpah [1998]], i.e. any external force at the manipulator is immediately felt at the motors of the manipulator.

5.3 Implementation on a real system

The proposed algorithm was implemented on a mobile manipulator and the results were compared to the results of a manipulator with a fixed base controlled only by the torque. The mobile manipulator used in our experiments was described in section 0 and is shown in Fig. 1. The primary task was trajectory tracking. The secondary task was the obstacle avoidance for the platform. The system was compliant in null space and stiff in the task space. We used $K_p = 1000$, $K_v = 160$, $K_n = 1$ for the controller gains. First we show the comparison between undisturbed motion of the mobile manipulator and the fixed manipulator. Later the mobile manipulator motion in space with obstacles and in contact with the environment is shown.

5.4 Free motion

The motion of both systems in the free space is shown. Two different trajectories were tested. The first was a circle with a radius 15 cm and a period of 20 s and the second was a larger circle with a radius 50 cm and a period of 50 s.

Fig. 15 a.) and Fig. 16 a.) show the trajectory of the end effector in (x-y) plane and b.) show the tracking error of the fixed and mobile manipulator, respectively. The average integral absolute error (AIAE) is defined as:

$$AIAE = \frac{1}{t_{\text{final}}} \int_0^{t_{\text{final}}} \sqrt{e_x^2 + e_y^2} dt, \quad (20)$$

where e_x and e_y are the tracking error in X and Y direction, respectively. The AIAE of the fixed manipulator was 7.5 mm and of mobile manipulator was 8.5 mm. The accuracy of the mobile manipulator was slightly smaller.

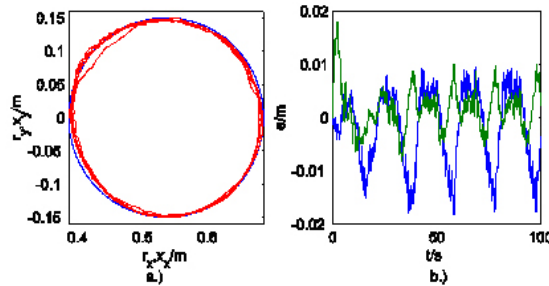


Fig. 15. Undisturbed motion of the fixed manipulator (circle $r=15$ cm). a.) trajectory, b.) tracking error.

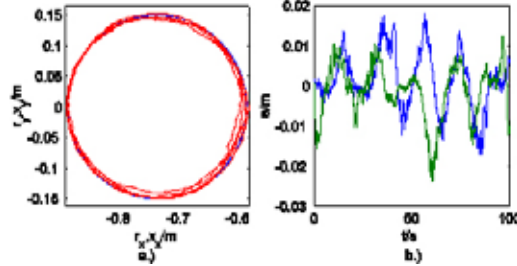


Fig. 16. Undisturbed motion of the mobile manipulator (circle $r=15$ cm). a.) trajectory, b.)tracking error.

The added mobility makes the workspace of the manipulator larger which can be seen when tracking the larger circle. Fig. 17 shows the motion of the fixed manipulator before reaching the end of its workspace, after that point the tracking error started to increase. With added mobility the workspace is infinite. The mobile manipulator tracked the whole trajectory, in spite of tracking the larger circle (Fig. 18). The AIAE of the larger circle was 7.5 mm and was approximately the same as the AIAE of the smaller circle.

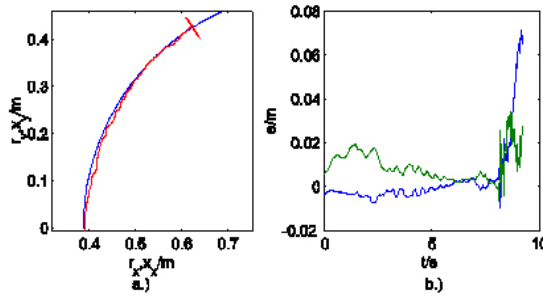


Fig. 17. Undisturbed motion of the fixed manipulator (circle $r=50$ cm). The manipulator came in the singularity and the motion is stopped. a.) trajectory, b.)tracking error.

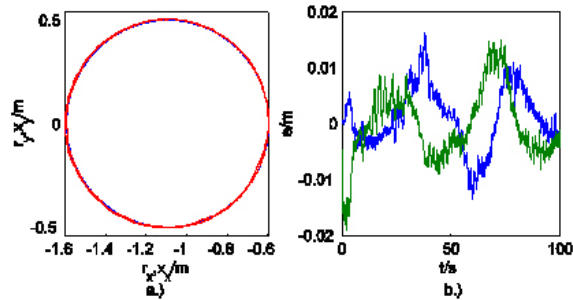


Fig. 18. Undisturbed motion of the mobile manipulator (circle $r=50$ cm). a.) trajectory, b.)tracking error.

5.5 Motion in unstructured space

The mobile manipulator moved in space with obstacles. The obstacles close to the platform were detected with the ultrasonic sensors. There were also obstacles that collided with the manipulator and were not detected. The trajectory reference was the larger circle. When the manipulator avoided the obstacles using the action-reaction principle the AIAE was 6.9 mm. When only the obstacles near the platform were present the AIAE was 8.6 mm. In case when obstacles near the platform and manipulator affected the motion the AIAE was 8.0 mm. Fig. 19 shows the motion when all obstacles were present. In this case the platform motion was much larger than in free motion case (Fig. 20). The AIAEs were approximately same as in undisturbed case. The maximal tracking error was larger when the obstacles collided the manipulator because the contact force affects the task space error. But after the contact the error decreased and the AIAE was approximately the same.

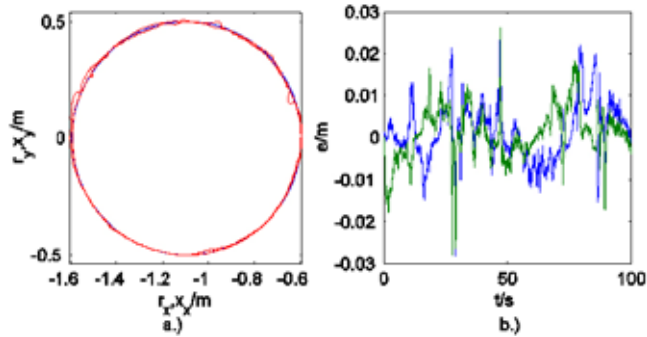


Fig. 19. Motion of the mobile manipulator in space with obstacles (circle $r=50$ cm).

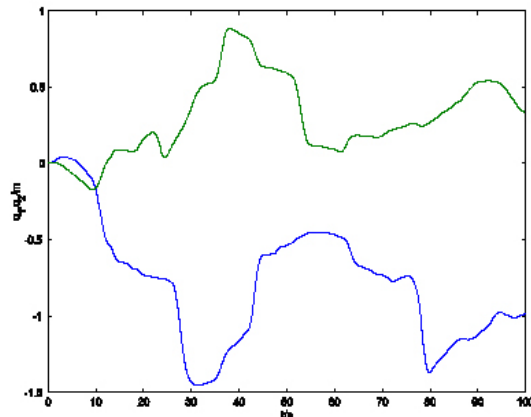


Fig. 20. Motion of the platform. The cause for this motion was mostly obstacle repulsion.

Fig. 21 shows the top view of the configuration of the mobile manipulator before and after the collision with the obstacle. The large blank circle shows the trajectory and

the smaller circles show obstacles detected with ultrasonic sensors. F denotes the force of the obstacle that collides with the platform. The collision place is only approximate because the obstacle position and force were not measured. The manipulator moved away from the obstacle after the collision in 0.8 s. The impact force was approximately 4 N.

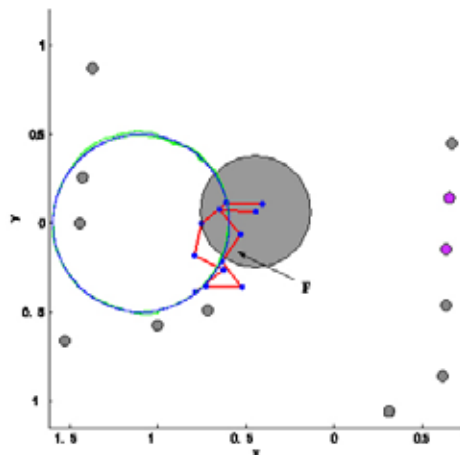


Fig. 21. Configuration of the mobile manipulator before and after the collision.

5.6 Motion in contact with the environment

One part of the trajectory reference (large circle) was inside of the wall. The mobile manipulator end-effector was tracking the trajectory to the wall and then it was pressing to the wall. The force of the contact is not controlled directly. It is dependent on the tracking error and the manipulator configuration. The motion of the manipulator in contact is shown in Fig. 22. Fig. 23 shows the force on the wall. Force measurement was used only for validation and not for the control.

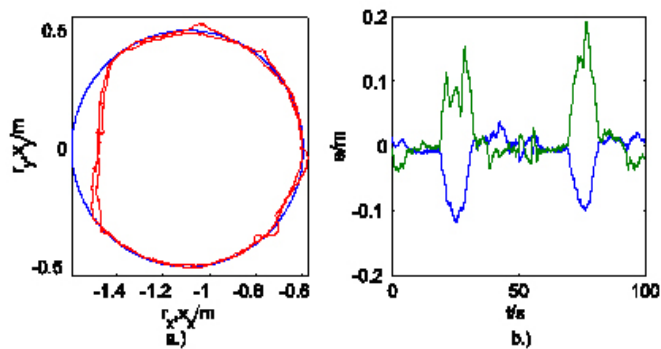


Fig. 22. Motion of the mobile manipulator in space with obstacles in contact with the wall.

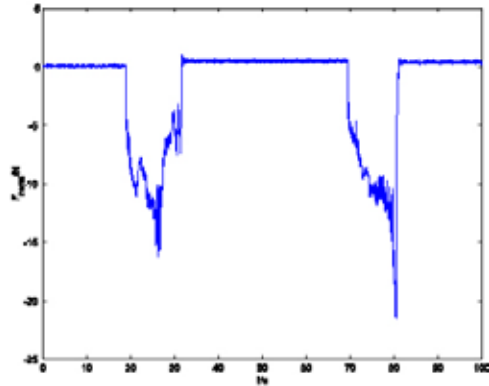


Fig. 23. Normal component of the manipulator force on the wall.

6. Conclusion

The use of the robots in the real world mostly depends on the development of a autonomous robotic systems that integrate mobility and manipulability. This chapter has presented a new approach to the robot control. The control is called combined control since it combines (integrates) two different types of robot control, i.e. the torque and the velocity control in a single robot system. Later on the mathematical analysis of the proposed combined control has been shown. Verification has been done on a simulated and a real system composed of a velocity controlled mobile platform and a torque controlled manipulator.

The results show good tracking of the desired trajectory in the task space and in the null space depending on the controller gains. Controller gains also define the compliance of the system in the task and in the null space. We defined the stability region of controller gains.

Regarding the system performance the combined control can be placed somewhere between velocity and torque control. Most of the properties of the combined control are comparable to those of the torque control and they are much better than those of the velocity control. When considering the trajectory tracking the combined control is comparable to the torque control, since in both cases most of dynamic interactions are compensated. The disadvantage of the combined control is inability of dynamic consistency and stiff behavior of the system when external forces act on the velocity controlled subsystem. On the other hand, the system with the combined controller can be compliant to external forces acting on the end-effector or on the segments of the torque controlled subsystem.

We have proved on a real system that the compliant control enables the mobile manipulator to avoid all obstacles in its workspace. Some of the obstacles are detected and they generate repulsive velocity. On the other hand we do not detect obstacles near the manipulator; avoidance is obtained by compliant control in the null space. We achieved a compliancy also in the task space.

Summarizing, the combined control is suitable for most applications where torque control is desired and part of the system can not be torque controlled. Additionally, no hardware modification of the motor controller in velocity controlled subsystem is necessary.

7. References

- C. Altafini. Inverse kinematic along a geometric spline for a holonomic mobile manipulator. In *International Conference on Robotics and Automation*, pages 1265-1270, Seoul, 2001.
- H. Asada and J.-J. E. Slotine. *Robot Analysis and Control*. A Wiley Interscience publication, John Wiley and Sons, Inc., 1986.
- N. Hogan. Impedance control: an approach to manipulation. *ASME J. Dynamic System, Meas. and Control*, 107:1-24, 1985.
- J. M. Hollerbach. Redundancy resolution of manipulators through torque optimization. *Journal of Robotics and Automation*, 3(4):308-316, 1987.
- R. Holmberg and O. Khatib. Development and control of a holonomic mobile robot for mobile manipulation tasks. *The International Journal of Robotics Research*, 19(11):1066-1074, 2000.
- P. Hsu, J. Hauser, and S. Sastry. Dynamic control of redundant manipulators. *Journal of Robotic Systems*, 6(2):133-148, 1989.
- O. Khatib. Mobile manipulation: The robotic assistant. *Robotics and Autonomous Systems*, 26:175-183, 1999.
- O. Khatib. Inertial properties in robotic manipulation: An object-level framework. *International Journal of Robotic Research*, 14(1):19-36, 1995.
- J. Lenarčič. On the execution of the secondary task of redundant manipulators. *Robotics and Autonomous Systems*, 30:231-236, 2000.
- J. Luh, M. Walker, and R. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3):468-474, 1980.
- Y. Nakamura. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Publishing Company, Inc., 1991.
- B. Nemeč. Force control of redundant robots. In *5-th IFAC Symposium on Robot Control*, Nantes, France, 1997.
- B. Nemeč and L. Žlajpah. Force control of redundant robots in unstructured environments. *IEEE Trans. on Industrial Electronics*, 49(1):233-240, 2002.
- D. Oetomo, M. Ang Jr., R. Jamisola, and O. Khatib. Integration of torque controlled arm with velocity controlled base for mobile manipulation: An application to aircraft canopy polishing. In *Fourteenth CISM-IFTToMM Symposium on Robotics RoManSy*, pages 308-317, Udine, Italy, 2002.
- D. Omrčen. *Kombinacija hitrostnega in navornega vodenja pri mobilnem robotskem manipulatorju*. Založba FE in FRI, Ljubljana, PhD disertaion, 2005.
- D. Omrčen and J. Lenarčič. Vodenje redundantnega sistema mobilne platforme z manipulatorjem. In *Electrotechnical and Computer Science Conference (ERK 2001)*, volume B, pages 425-426, Portorož, Slovenija, 2001.
- D. Omrčen, L. Žlajpah, and B. Nemeč. Autonomous motion of a mobile manipulator using a combined torque and velocity control. *Robotica*, 22 (6):623-632, 2004.
- R. S. Oropeza and M. Devy. Motion control using visual servoing and potential fields for a rover-mounted manipulator. In *International Conference on Robotics and Automation*, pages 233-240, Detroit, Michigan, 1999.
- L. Petersson, D. Austin, and D. Kragic. High-level control of a mobile manipulator for door opening. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2333-2338, Takamatsu, Kagawa, Japan, 2000.

- M.H. Raibert and J.J. Craig. Hybrid position/force control of manipulators. *Journal of Dynamic Systems, Measurement and Control*, 102:126-133, 1981.
- J. Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *Proceedings of the 19th IEEE Int. Conference on Decision and Control*, pages 95-100, Albuquerque, New Mexico, USA, 1980.
- Y. Yamamoto and X. Yun. Effect of the dynamic interaction on coordinated control of mobile manipulators. *IEEE Transactions on Robotics and Automation*, 12(5):816-824, 1996.
- L. Žlajpah. Influence of external forces on the behaviour of redundant manipulators. *Robotica*, 17:283-292, 1999.
- L. Žlajpah. Compliant motion control of redundant manipulators in constraint space. In *Fifth IFAC Symposium on Robot Control (SYROCO'97)*, pages 657-663, Nantes, France, 1997.
- L. Žlajpah. Obstacle avoidance control for redundant manipulators utilizing the contact forces. In *IEEE International Conference on Intelligent Engineering Systems*, pages 343-348, Vienna, Austria, 1998.

EMOBOT: A Robot Control Architecture Based on Emotion-Like Internal Values

Nils Goerke

*Dept of Neural Computation, University of Bonn
Germany*

1. Introduction

Controlling robots and other autonomous agents has been successfully performed over a long period using a wide variety of approaches within various applications. The dilemma between using predefined methods (reliable but rather inflexible) and learned capabilities (more flexible, but sometimes inexact) is still a challenge for the field.

Designing a structure that is in principle capable of performing the desired tasks and attempting to gain this capability throughout a learning process is one of the most interesting areas in contemporary design of robot controllers (Braitenberg, 1985, and Brooks, 1985 & 1986). Neural network approaches and neural learning paradigms have been widely used to implement parts of these control architectures.

The idea of generating “emotion-like” behaviour for technical agents seems to be a valuable approach to modelling more sophisticated and more flexible capabilities within robotics. An extension of the “emotion-like” approach to behaviour using the capabilities of adaptive mechanisms seems to yield a more complex class of behavioural features.

The realisations of learning emotion-driven robot controllers are not meant to compete with the established and approved paradigms in robotics and autonomous systems, but try to provide a set of “extra” features to the field (Gadanhó & Hallam, 1998a,b and Cos & Hayes 2002 and Gadanhó & Custodio, 2002). The author is convinced that the proposed “exotic” approach is a valuable alternative to established approaches to learning behaviour.

Since no “teacher” in the classical sense is available for the envisaged task, a reinforcement based learning scheme has been developed to alter the entries of the action selection matrix.

2. Controlling autonomous robots by selection of behaviour primitives

2.1 Different levels of control

Autonomous robots can be controlled on different levels with different scientific disciplines involved (classical control theory, neuroinformatics, biocybernetics, artificial intelligence, psychology, cognitive science). From the low-level control like motor speed, driving duration and movement direction via basic capabilities like obstacle avoidance and simple control loops to mid-level behavioural capabilities like searching, foraging map making...

up to high-level behaviours that we are used to now from our domestic animals (McFarland, 1999) and that we would like to obtain for our robots.

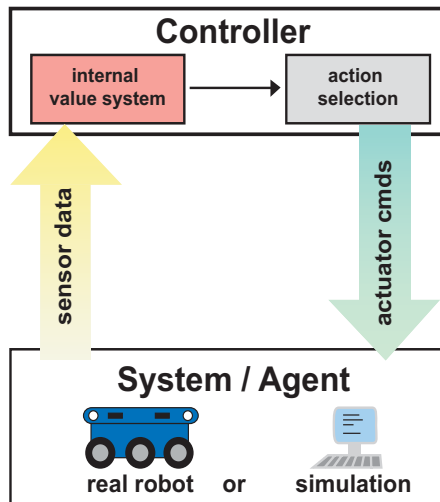


Fig. 1. Systemic architecture with the robot, the sensory upstream (left) the actuary downstream (right), the action selection (upper right) and the internal value system.

2.2 Low-level control

The basic level of robot control is directly aligned to the physical properties of the robot system: actuators (which are mostly electrical motors) and sensors (which typically yield electrical values). The robot actuators are controlled following the solutions from control theory: typically the motor position, motor speed and the motor torques are governed using P-, I-, D- or PID-controllers. Autonomous agents are typically controlled via the speed of their wheels and the by the duration of activity. Different speed demands for the wheels will result in different turning angles, and thereby in different movement directions (drive forward, turn left, turn right, turn around, stop ...).

Controlling autonomous robots just by means of low-level control is annoying and time consuming. Complex tasks can only be accomplished with a big effort.

2.3 Mid-level control 1

The next level of complexity in robot control realises basic robot capabilities that imply a minimal level of internal representation. A basic control mechanism is a feed-back loop that uses the sensor values to calculate the motor activities as set points for the low-level control. Typical examples are: reactive movements like Braitenberg type vehicles (e.g. 3b), wall-following behaviour, reactive obstacle avoidance, a "follow me" behaviour, random-move until a special condition is reached...

Each of these mid-level capabilities can be implemented into a controller as behaviour primitive of its own. A clever choice of these behavioural modules can lead to a more

complex and smart behaviour of the robot. Still, controlling an agent by these mid-level capabilities can be very laborious.

2.3 Mid-level control 2

The really interesting part in robot control is dealing with the combination of lower levelled capabilities. The mid-level control is combining and sequencing several of the capabilities that have been realised in the lower levels of the control hierarchy to gain more complex behaviour.

This stage of robot control requires an internal memory structure, some planning capabilities and an explicit task to be performed. Dependent on the task and the effective situation the robot is in, different behaviour primitives (mid-level 1) can be triggered, stopped, or modified to complete the desired task.

Typical examples for mid-level-2 capabilities are: search for an object, search for a location, foraging behaviour, map making, escape from a labyrinth, kick a ball into a predefined direction...

All of these mid-level 2 behaviours depend on other mid-level capabilities, and are goal directed behaviours. Most nowadays robotic tasks, and most autonomous robot jobs, can be realised by combining mid-level-2 behaviours. The clever combination of these capabilities is still considered "artwork" and has not yet become a closed theory.

2.4 High-level control

High-level behaviours for autonomous robots are typically aligned with behaviours that we are aware of from our day-to-day life. We might know this kind of behaviours from our domestic animals and from our own (human) behaviour. We certainly want our robots to gain these behaviours, but today we are still lacking these capabilities in modern autonomous systems. Most high-level control tasks have linguistic commands that deal with symbolic information and with complex tasks. The following sentences might serve as examples for the kind of high-level control the robotic community is looking for: "Search for the third room on your left.", "Go and count the number of chairs.", "Deliver the mail.", "Look out for unusual objects.", "Tell me what you have found in room 3."

High-level control covers a wide range of capabilities that we would like to have for our robots. Although today only a few convincing implementations can be found, the progress achieved during the last years is tremendous and increases further. The author is convinced, that in a few years truly high-level control for robots will be available.

2.5 Controlling by Action Selection

One way to control a robot making benefit of all described levels of control is to combine the realised capabilities from the different levels by explicitly selecting them. Other approaches with action selection for robots can be found in (Arkin, 1998) and (Burghouts et al., 2003) and (Velásquez, 1997). Each capability is treated as an individual behaviour primitive which can be an action from one of the control levels, or a clever combination of several of these actions. The action selection mechanism triggers one of these behaviour primitives with respect to the condition the robot is in. The action selection can depend directly on the sensory state of the robot (e.g. mid-level control), can depend on some internally calculated results (e.g. world model, a map), will probably depend on the given task for the robot (e.g. goal directed behaviour), or can depend on the specially designed set of internal values.

The task of mapping the state of the robot onto one of the available behaviour primitives can be realised in various ways: e.g. neural networks, action selection matrix, mixing of experts, rule based, artificial immune system approach, fuzzy controllers, expert systems and a wide variety of further approaches.

3. The need for an internal value system

Beside pure reactive behaviours autonomous agents, like robots can be governed by a set of internal values, giving these robots a sort of internal state. This internal state can serve as a data basis (even growing data basis) for guaranteeing autonomy in planning and behaviour for the robot. The more complex this internal data basis is, the better the involved planning algorithms can determine and calculate the robot actions, or sequences of actions.

Most robot control systems rely on some sort of world model as basis for the planning of actions, action sequences or even basic movements (see Murphy, 2002 or Siegwart and Nourbakhsh, 2004). The internal world model can be extremely basic and the resulting control mechanisms thus even "hard-wired" (e.g. Braitenberg (1985) vehicles type 3b), the world model can be more complex including topologic assumptions (e.g. maps) and calculations that are based on the monitored actions of the robot actuators (e.g. dead reckoning, SLAM). The world model can be extremely complex including a physics based simulation of the robot, the environment and/or even other non stationary entities (e.g. moving obstacles, multi robotics, robot soccer...). Beside the simulation of the world, internal values from the robot system can be used to model and represent the robot state: e.g. battery condition, speed of the wheels, duration of movement, and percentage of task fulfilled.

In addition to values and conditions that are related to physically measurable events of the robot and the environment, a special set of values can be defined, that represent internal values of the robot control system, e.g.: the condition of the task execution, the overall performance of the system, a time and/or distance dependent measure... This set of internal values is denoted Internal Value System (IVS). The action selection mechanism depends not only on physically measurable effects, but additionally on the internal value system of the robot.

4. Internal values inspired by psychology (emotion-like values)

4.1 Inspired by psychological concepts

Using the idea of an internal values system that is not only dependent of physical conditions but in addition on some-how meta-states of the robot the IVS has been designed. These internal values are designed in accordance with psychological terms that we (human beings) associate with "*Drives*" and "*Emotions*", (Damasio, 1994) and (Velásquez 1997). These internal values do not realise real "*Drives*" and "*Emotions*", but the robot controller is designed in such a way, that the robot shows a behaviour that seems to be governed by "*Drives*" and "*Emotions*" (see Canamero, 1997 for an alternative approach). The psychological concepts used, help the human operator to design and judge the resulting behaviour of the robot system. The robot behaviour is imitating emotionally driven behaviour. Yet we emphasise that at no stage in the proposed approach we claim to realise "*Drives*" or "*Emotions*" as they are assigned to living entities

4.2 “Drives”

The internal values that are aligned with the psychological term “Drives” are implemented in a way that tries to match with these psychological terms. We call these internal values “Primary Internal Values” because they depend directly and only on the results of the sensory upstream of the systemic architecture and on time. Drive values are bounded within the range from -1 to $+1$, with a desired value of 0 which is indicating that the robot is in a balanced internal state.

We have implemented four primary internal values:

- **Fatigue:**

Movement makes the fatigue value rise linearly, resting makes it decrease linearly, but steeper than rising. The virtual binary sensor “Active” indicates if the robot is doing something or not.

$$Fa_{t+\Delta t} = Fa_t + \frac{\Delta t}{120s} * Active + \frac{\Delta t}{40s} * (1 - Active) \quad (1)$$

- **Hunger:**

Hunger is linearly increasing in time, and reduced by a fixed amount through “Food”. A virtual binary sensor indicating “Food” has been implemented.

$$Hu_{t+\Delta t} = Hu_t + \frac{\Delta t}{60s} * (1 - Food) + \frac{2}{3} * Food \quad (2)$$

- **Homesickness:**

The longer the robot is not at the home position, the more the homesickness value rises, linearly in time. A linear but faster decrease occurs when the home position is reached. “Home” was realised via a virtual binary sensor indicating if a special area within the environment has been reached.

$$Hq_{t+\Delta t} = Hq_t + \frac{\Delta t}{240s} * Home + \frac{\Delta t}{20s} * (Home - 1) \quad (3)$$

- **Curiosity:**

The more similar sensory input the robot encounters, the more the curiosity drive rises. Novel input patterns make the curiosity decrease. We have implemented a virtual sensor “Distraction” measuring the flow of information through the sensors to guide the curiosity value. The curiosity rises linear with time and is decreased, if something interesting occurs. As long as the distraction is above a fixed threshold, indicating something interesting, the curiosity decreases linearly.

$$Cu_{t+\Delta t} = Cu_t + \frac{\Delta t}{60s} * (1 - Distr) + \frac{\Delta t}{1s} * Distr \quad (4)$$

4.3 “Emotions”

The secondary set of internal states is called “Emotions”. These internal values depend mainly on the satisfaction of the primary internal values (i.e. their closeness to 0) and are roughly aligned with the respective psychological term “Emotion” that we are familiar with. Still, we are not claiming to model the psychology. Our robot does not have any real “Emotions” (and will probably never have). These internal values reflect internal states that we have designed to show a specific behaviour. The selected four “Emotions” are four of the five universal emotions (Damasio, 1994), only sadness was replaced by boredom, since this emotion better fits the desired and implemented functionality.

- **Fear:**
The internal value Fear increases with rising Fatigue and rising Homesickness values and decreases with falling Curiosity values as well as with time.

$$Fe_t = Fe_{t-\Delta t} + \frac{Fa_t}{15} + \frac{Ho_t + |Ho_t|}{20} + \frac{|Cu_t|}{30} - dec_{Fe} \quad (5)$$

- **Anger:**
All of the four drives influence the emotion of Anger. Hunger causes it to increase strongly, Homesickness raises it a little, and even Curiosity increases it a little bit, while Fatigue decreases the Anger moderately, as it does with time.

$$An_t = An_{t-\Delta t} + \frac{Hu_t}{10} + \frac{Ho_t}{35} + \frac{Cu_t + |Cu_t|}{100} - \frac{Fa_t}{30} - dec_{An} \quad (6)$$

- **Boredom:**
Boredom is increased if the drives are negative, and is decreased if the drives have positive value.

$$Bo_t = Bo_{t-\Delta t} + \frac{-Hu_t - Ho_t - Cu_t - Fa_t}{30} - dec_{Bo} \quad (7)$$

- **Happiness:**
The internal value of Happiness increases when the drives are close to the balanced state (i.e. close to 0). High Hunger or Homesickness values decrease the Happiness more than Curiosity or Fatigue do. The other "Emotions" influence the Happiness as well: The lower the values for Fear, Anger and Boredom are, the more the Happiness value raises with time.

$$Ha_t = Ha_{t-\Delta t} - \frac{(Hu_t + |Hu_t|) + (Ho_t + |Ho_t|) + 2An_t + 2Fe_t}{40} - \frac{(Cu_t + |Cu_t|) + (Fa_t + |Fa_t|) + 2Bo_t}{100} + dec_{Ha} \quad (8)$$

4.4 The EMOBOT internal value system

The internal value system is part of the controller. It is designed to reflect internal states of the robot to support the control and selection processes. The internal values are a set of real valued components derived from the results of the sensory upstream and the inherent rules that govern the dynamics of these internal values. Within the IVS we have implemented two classes of values: primary ones ("Drives") that depend directly on time and the sensory upstream and secondary ones ("Emotions") that depend mainly on the primary internal values and on each other.

5. The EMOBOT control architecture

5.1 Systemic Architecture

The proposed control architecture for the robot is based on the ideas of systemic intelligence to span the bridge between sub-symbolic representation of knowledge (neural networks,

fuzzy control, fuzzy logic, rules, differential equations...) and symbolic described capabilities (goals, reasoning, behaviour, intention...). The intelligent behaviour shown by living systems and some technical artefacts is neither the consequence of the symbolic description of their tasks, nor the consequence of the sub-symbolic representation of information. It is postulated that intelligent behaviour arise only if the design of the system has been performed adequate (Kintzler, 2002 and 2003).

The systemic architecture is a design concept for building and structuring the architecture of controlling systems (Goerke, 2001 and 2003). Based on experience and knowledge from neurosciences, neurocomputing, psychology and cognitive sciences, a systemic architecture features a set of capabilities and principles to be portable onto a technical artefact. The design principles governing the systemic architecture are not meant to be strict, but are understood as an overall guideline for the design of a systemic architecture controller.

Following these design-principles, a systemic architecture consists of a hierarchical structured network of simple and if possible independently adaptable building blocks. Each building block consists of modules which process information and learn capabilities. Thereby the layers refer to the layers below by using the provided information and the range of functions and capabilities immanent in these underlying layers.

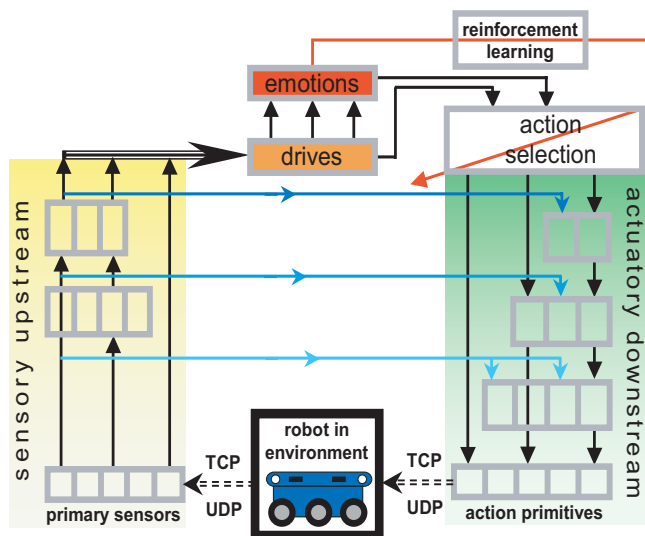


Fig. 2. Systemic architecture with the robot, the sensory upstream (left) the actuator downstream (right), the action selection (upper right) with the reinforcement learning module and the internal value system with drives and emotions.

The presented approach implements the controller by a set of modules with distinct functionality. These modules are organised in layers, where higher layers are designed to implement higher functions. In addition, the architecture is organised into two main branches: One branch going from low level sensory data, upwards to higher sensory capabilities (sensory upstream, left branch in Fig. 1 and 2). The other branch is directed from

high level, highly sophisticated control schemes, down to low level motor functions (actuator downstream, right branch in Fig. 1 and 2).

5.2 Mixing controllers and switching controllers

Within the low-level control and the mid-level-1 control layers of the architecture several modules have been realised. Each of these modules implements a behaviour primitive. This assembly of modules is called a "bank of behaviour primitives". The controller is now selecting and combining the different behaviour primitives to generate the steering commands for the actuator system, with respect to sensory values and the to the internal value system "Drives" and "Emotions" (Goerke & Müller, 2003).

Three possible variations of implementing the controller are reasonable: switching controller σ, α , switching controller σ, Ω and mixing controller μ .

Switching controller:

The switching controller switches explicitly between the different behaviour primitives. If the switching controller is positioned before the bank of behaviour primitives, it will be denoted σ, α ; if it is positioned behind the bank, it is denoted σ, Ω .

For the σ, α type, only one of the behaviour primitives have to be active at a time (faster). The disadvantage is that the results of the other behaviour primitives can not be taken into account for the switching process. For the σ, Ω variant all behaviour primitives have to be calculated (resource consuming) before the decision (switching) takes place.

Mixing controller:

The mixing controller combines (or mixes) the different results from the bank of implemented behaviour primitives to compute the control command for the robot. The combination can be realised as a liner-combination, or as a non-linear mapping from behaviour primitive outputs onto motor values (Skarmeta et al, 1999). The abbreviation for a mixing controller is μ . Mixing controllers are always positioned behind the bank of behaviour primitives, and therefore always of type μ, Ω .

5.3 Action selection

The task of selecting an adequate action from the available behaviour primitives with respect to the actual state of the robot is called action selection. This is reduced to the mapping from the continuous state space of the internal values to the discrete set of behaviour primitives. Several approaches for this mapping operation have been tested and implemented: e.g. action selection matrix, neural network (type MLP).

5.4 Action selection matrix

Within the presented approach, the EMOBOT controller is realised as a switching controller of type σ, α . With respect to the state of the internal values, the action selection chooses one of the available behaviour primitives. Therefore the space of possible states has been partitioned into disjoint areas. To each of these areas a behaviour primitive is assigned to (action selection matrix).

For this action selection, we have quantised each of the real valued "Drives" into the four regions: very low (-1.0, -0.5); low (-0.5; 0.0); high (0.0, +0.5); very high (+0.5, +1.0) see Fig. 2 left part. Thus, the four dimensional state space of the four "Drives" is segmented into $4 \times 4 \times 4 \times 4 = 256$ disjoint areas. Each of these areas is assigned to one of the behavioural primitives (see Fig. 3 right part and Fig. 10).

Depending on the state of the “Drives” from the internal value system, the switching controller triggers one of the behavioural primitives.

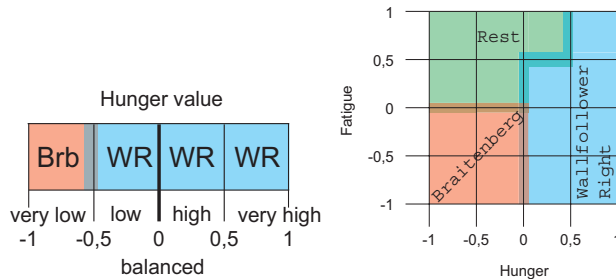


Fig. 3. The quantised internal values (e.g. Hunger and Fatigue) are quantised into four segments; each segment is assigned a behaviour primitive (action selection matrix).

5.5 Learning action selection

Learning the matrix means to assign adequate actions to the 256 state boxes. Since we have no real teacher available we can not use a supervised learning paradigm. Therefore, we have implemented a reinforcement based scheme to change the action (behavioural primitive) within each box.

Four paradigms to obtain an adequate reinforcement signal have been realised:

- **Reinforcement calculated by an objective function:**
An external objective function is calculated to gain a reinforcement signal: e.g. distance to obstacles, time until a special situation is reached, number of turns, ratio of forward to turning movements, number of times a special position has been visited...
- **Reinforcement provided by a human observer:**
A human operator observes the behaviour of the robot system and judges this behaviour with a reinforcement signal (reward and punishment). This reinforcement signal is extremely depending on the human observer, it is not reproducible nor is it stationary, as the mood the human operator is in, might change during the experiment.
- **Reinforcement calculated by a trigger signal or trigger event:**
The reinforcement signal is triggered by an external event, e.g. a special position within the environment, a special obstacle configuration, or a special sensory input.
- **Reinforcement based on secondary internal values (state of “Emotions”):**
The overall condition of the “Emotions” is used to generate the reinforcement signal. As long as the robot is in a “good emotional state” a positive reinforcement is generated to further foster the most recent behavioural primitives, and a negative reinforcement is generated as soon as the “condition of the robot” becomes unwanted (threshold) to punish the actions that has caused this.

5.6 Adaptive matrix entries

To make the action selection matrix learnable, we have extended each entry of the matrix into a priority vector, containing one entry for each of the possible behavioural primitives. The resulting matrix contains one priority value for each available action, for each of the possible IVS situations.

Now it is possible to use the priority value within the matrix for a winner-takes-all strategy, choosing the behavioural primitive with the highest priority. As a second variant the priority value can be used to determine the selection probability for choosing the respective action. The winner-takes-all strategy implements a pure exploitation, and the probability view implements the exploration mechanism of reinforcement learning (Sutton & Barto, 1998) and (Kaelbling et al., 2002).

The reinforcement learning is now capable of changing the priority values for each matrix entry with respect to the reinforcement signal. Punishment of a specific behavioural primitive is realised as decrease and rewarding this very action as moderate increase of the respective priority value (see Fig. 11).

5.7 Adaptive regions of interest

As an alternative to the fixed tiling of the state space into unit size boxes, the partitioning can be made adaptable. We propose to learn this partitioning with a neural network, e.g.: Self Organising feature Map (SOM), Learning Vector Quantisation (LVQ), Regional and Online Learnable Fields (ROLFs), Multi-SOMs.

5.8 Neural network based controller

As a first step towards a more general realisation, we propose a neural network, type Perceptron (P) or Multi Layer Perceptron (MLP). The input to the neural net, denoted with \mathbf{X} , is the vector of the internal values. The output vector \mathbf{Y} of the neural network is now post-processed with a winner takes all, 1 out of n , decision process to form the binary decision vector DV , with only one component being active $dv_w = +1$ and all other components being inactive $dv_i = -1$. The active component w indicates the winner, and thus the behavioural primitive that has been chosen for activation. Thus, the network is switching between the available actions with respect to the sensor and internal values, for further details see Goerke et al, 2004.

6. Realisations of the EMOBOT approach

To test and evaluate the approach we describe three realisations of the EMOBOT with different complexity. A simulation of an EMOBOT within a grid world, and with reduced capabilities. A simulation of an EMOBOT in a continuous valued environment. An implementation of the EMOBOT control approach for a real robot system within a well controlled testing environment. All realisations were focused on different aspects of the EMOBOT system and are therefore deliberately different.

6.1 Grid world EMOBOT

The simple EMOBOT realisation is based on a 26×22 grid world, with a boundary and several obstacles. Each cell of the grid can monitor and remember how often the robot has visited this very site. The grid based robot is one square of the grid in size; its orientation is limited to multiples of 45° . The robot sensors can monitor the three directly adjacent squares in front of the robot (yielding 8 possible obstacle situations). The sensory system can register from the grid world how often one of the eight surrounding squares have been visited by the robot up to now $M_{x,y}$. The actions the robot can perform within each time step are either to move one square forward, to turn 45° and to move, and to perform a 135° escape turn. Two basic behaviours have been realised for the grid based EMOBOT realisation. Both schemes are completely deterministic and have no stochastic components.

- **A pure Reactive behaviour R:**
Depending of the actual obstacle information (one of eight possible situations), the reactive controller is switching between the different actions (see Fig. 4).
- **An eXplorative behaviour X:**
Knowing how often the robot has visited the surrounding squares, the robots turns via the cell visited the least, and moves one step forward onto it, thus further exploring its world.

Sensory values			Action	$t_i \rightarrow t_{i+1}$		$t_i \rightarrow t_{i+1}$	
left	front	right					
0	0	0	forward				
0	0	1	forward				
0	1	0	left,forward				
0	1	1	left,forward				
1	0	0	forward				
1	0	1	forward				
1	1	0	right,forward				
1	1	1	escape turn				

Fig. 4. The reactive behaviour for the grid based EMOBOT implementation.

The switching controller is choosing the explorative behaviour if the curiosity value is rising above a certain threshold. For the grid world based robot, the curiosity has been model different to equation (4) taking the discovered places and two control parameters p and q into account:

$$C_{i+1} = C_i + p + q \cdot \frac{C_i}{M_{x,y}} \quad (9)$$

To get a feeling what kind of behaviour the robot is establishing, the two behaviour primitives were tested separately. Within a second test phase the two actions were combined using the curiosity value to decide between the two behaviours: Reactive behaviour (R), and eXplorative behaviour (X). Using only the explorative movements the robot is exploring the whole world (or 90% of it). Depending on the actual starting position the robot will pass the different squares of the grid world different times (counted within $M_{x,y}$). To reach 90%= 377 out of the possible 419 squares the explorative control needs just 550 steps (averaged over all 419 starting positions). Since no real searching strategy has been implemented, 550 required steps is regarded as reasonably good. The author is aware of exploration algorithms that perform a lot better; but the efficiency of the exploration is not the scope of the presented research.

Equipped with only the reactive behaviour the robot would never be able to explore the whole world, because the implemented control tends to establish stereotypic movements (see Fig. 5). In detail, only three stable trails emerge from the chosen combination environment and reactive behaviour (A,B,C). Depending on the starting position one of the three trails is reached, and the robot's movement is stuck within that trail.

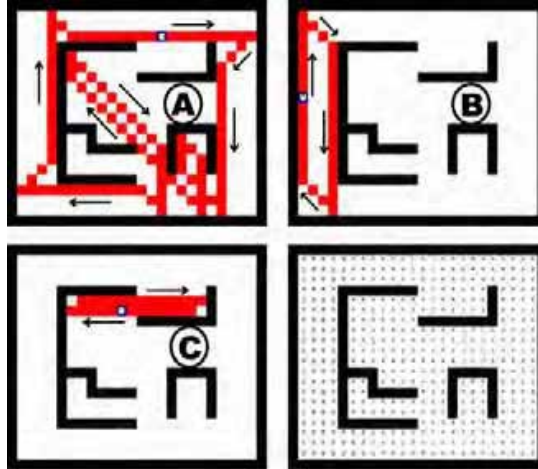


Fig. 5. The three trails (A,B,C) that result when using the reactive behaviour. The dependency of the trail from the starting position is depicted in lower right.

When both control mechanisms are combined with respect to curiosity and the threshold, the robot consequently shows a complex mix of both behaviours. With respect to the starting position and to the parameters (p and q) that model the curiosity the final behaviour has been investigated in extensive simulations (see Fig. 6). For each combination of p, q values all starting positions within the grid world have been tested and the resulting behaviour (Trail A,B,C or eXplorative behaviour) monitored and visualised (see Fig. 6). It is easy to see, that a variety of different behaviour can be reached by changing the Curiosity parameters p and q .

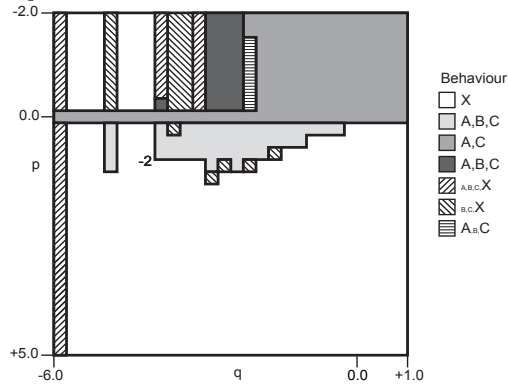


Fig. 6. The resulting behaviour with respect to Curiosity control parameters p, q . The behaviour B,C,X for example indicate mostly eXplorative movements with some starting positions leading to the trails B and C.

6.2 Real valued EMOBOT

The second implementation is the simulation of EMOBOT in a more realistic environment (real valued, several arbitrary shaped objects, Fig. 7), with more behavioural primitives (12) and with more internal values (5) as input for the action selection. The simulated robot has five distance measuring sensors (three front, two rear, see Fig. 7) and has the capability to move forward and to make turns.

The internal values modelled for this EMOBOT realisation are:

- **Boredom:** rising with time, decreasing with changing sensor values.
- **Curiosity:** rising with time, decreasing when new terrain is explored.
- **Home sweet Home:** small near the home position, increasing with Euclidian distance.
- **Joy:** rising in time, decreasing when a special sensory pattern is encountered.
- **Tiredness:** decreasing when the robot is inactive, moderate increase while turning strong increase when the robot is moving forward.

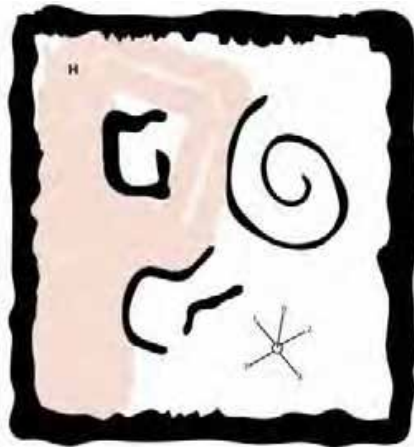


Fig. 7. The real values testing environment with some obstacles and the Home position, including the robot with the five sensors.

The implemented behaviour primitives are taken from low-level and mid-level control schemes:

1. *Sit and rest:* just stop, rest and look for a while.
2. *20° left turn.*
3. *Forward:* move forward 4 times the robot size.
4. *Escape movement:* 180° turn and run.
5. *Step back:* move backwards, 3 times the robot size.
6. *Search exit:* turn right until free space in front.
7. *Random walk:* turns and moves randomly.
8. *Side step right:* 90° turn right, 2 forward, 90° turn left.
9. *20° right turn.*
10. *Search object:* move forward until front sensors detect something;
11. maximal 10 times the robot size.

12. *Wander*: move and turn right right-before bumping

13. *Braitenberg vehicle*: type 3b, obstacle avoiding

To investigate the capabilities of this approach, several series of simulations have been conducted; part of the results can be found in (Goerke et al., 2004). From the status of the internal value system a reinforcement signal is computed that serves as basis for training the action selecting neural network. The neural network is trained by using the reinforcement-generated artificial teacher vector, and a temporal discounting factor for recent situations as extension to the learning rule backpropagation of error. The internal value system, the neural network, the emotional controller, the bank of 12 behaviour primitives, and the simulated robot within an environment are implemented. Extensive simulations yielded a feasible way to obtain a variety of different and interesting behaviours for the EMOBOT.

6.2 EMOBOT implemented on a real robot system

The EMOBOT approach has been implemented to control a real robot using the set of 8 internal values (the 4 primary “Drives” and the four secondary “Emotions”). The action selection is based on the matrix with 256 adaptable entries. Each entry is triggering one of 10 implemented behaviour primitives (4 mid-level-1, 5 mid-level-2 and sit-and-rest).

The robot is equipped with 2 motors, each driving 3 wheels on one side of the robot. The size of the robot is 35cm wide, 45cm long and 35cm high. The robot is equipped with 6 ultrasonic distance measuring sensors, 18 infrared distance measuring sensors, and two ambient light sensors, (see Fig 8).

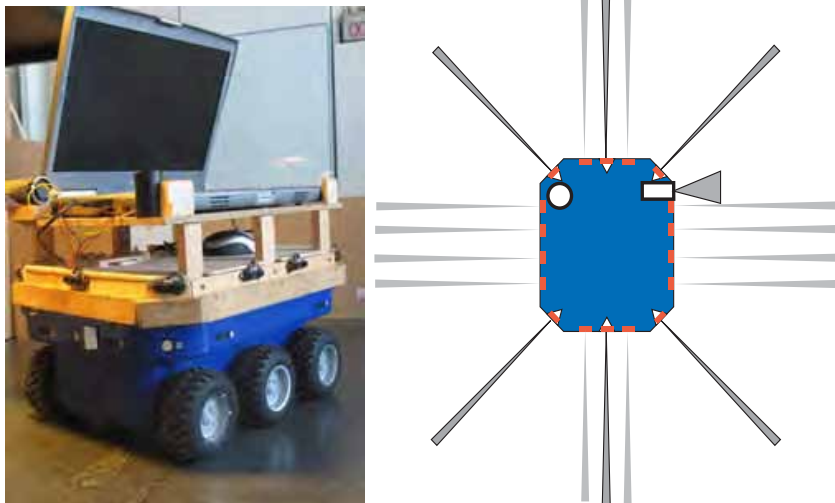


Fig. 8. The real robot with the controlling notebook on top (left picture) and the layout (right picture) for the distance and ambient light sensors (circle, rectangle).

In this work, only the infrared distance sensors and the ambient light sensors were used. The infrared sensors have a reliable range between 10cm and 80cm, the width of their beams

is 6-8cm. Obstacles closer than 10cm are not reliably detected by the infrared sensors, so that the controller needs to provide for maintaining a distance to obstacles greater than 10 cm. Objects that are smaller than the gap between the sensor beams (e.g. legs of tables) are invisible to the robot as well. One ambient light sensor heads to the right side of the robot (white rectangle in Fig. 8), the other to the ceiling (white circle in Fig. 8).

The robot control scheme is designed to work in a common office environment, with walls, hallways, rooms, doors, tables, chairs (non stationary, slow) and arbitrary moving obstacles (humans). For development and evaluation purposes we started with an artificial environment: smaller in size, no fast moving obstacles, and no legs of chairs that fit between the sensor beams and would thus be invisible for the robot.

The environment depicted in Fig. 9 was used for most of the experiments is static, well controlled testing ground of 3mx4m. All obstacles are stationary, or substantially slower than the robot; we can easily alter the shape of the environment by moving one of the wooden walls to a different location, even during a test run, without harming the robot behaviour as the movements of the robot do not rely on a map, but are a consequence of the current situation. The environment contains several passages with straight walls, 13 +90° corners, three sharp -180° corners and one -90° corner. In addition, four light sources have been placed in the environment for being visible by the ambient light sensor (lighter rectangles in Fig. 9). One place in the environment is lit (oval structure in Fig. 9) for being detected by the upward directed light sensor. This special location is regarded as home position for modelling homesickness.

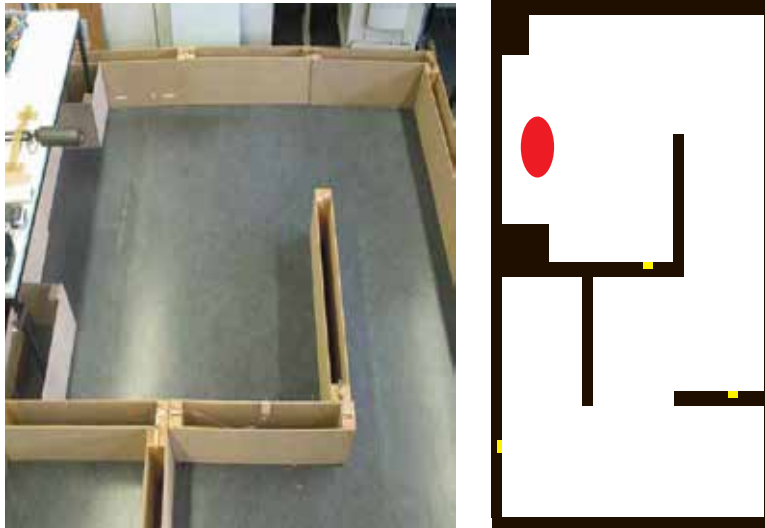


Fig. 9. The environment (picture) and a schema (right) with the home position (oval area) and the 4 locations of Food (lighter rectangles within walls).

The behaviour primitives implemented are only exemplary realisations that have been chosen with respect to the real robot system from the different controlling levels:

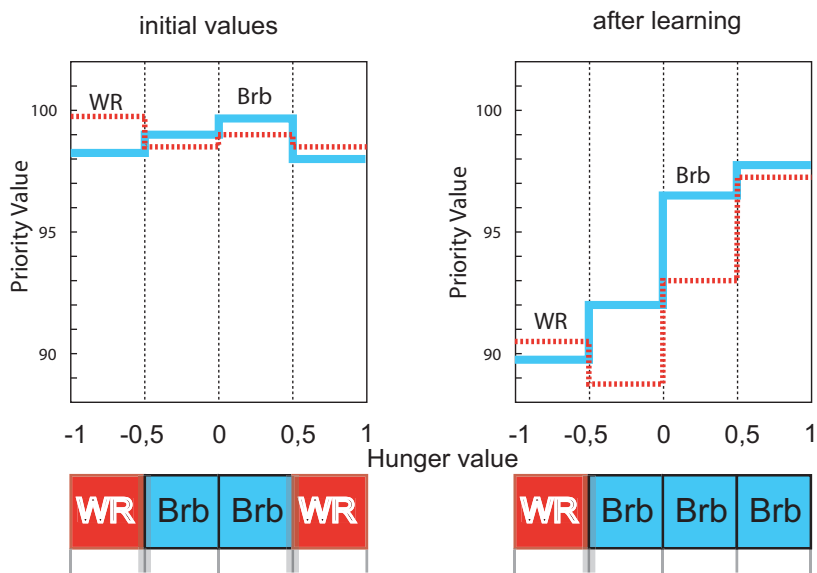
- **Low-level control:**
Forward moves are performed with 0.2m/sec and turning with 96°/sec. The seven implemented behaviour primitives are: Move forward, Move backwards, Turn left, Turn right, Drive a curve to the left, Drive a curve to the right, S "Sit and Rest".
- **Mid-level-1 behaviours:**
WL: Wall Following left wall, **WR** Wall Following right wall, **BrB** Braitenberg movement, type 3b, **Pos** Position the robot in front of an object.
- **Mid-level-2 behaviours:**
R Random Move, **Esc** Escape, **Sea** Search a Passage, **Rou** Round-Trip, **Or** Orbit around an object.

		Ho	vL: very Low				L: Low				H: High				vH: very High			
Fa	Ho	vL	L	H	vH	vL	L	H	vH	vL	L	H	vH	vL	L	H	vH	
	Cu	vL	L	H	vH	vL	L	H	vH	vL	L	H	vH	vL	L	H	vH	
vL	vL	Brb	Brb	WR	WR	Brb	Brb	WR	WR	WL	WL	WR	WR	WL	WL	WL	WR	
	L	Brb	Brb	WR	WR	Brb	Brb	WR	WR	Rnd	Rnd	WR	WR	WL	WL	WL	WR	
	H	Esc	Esc	WR	WR	Rnd	Rnd	WR	WR	Rnd	Rnd	WR	WR	WL	WL	WL	WR	
	vH	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	WL	WL	WL	WR	
L	vL	Brb	Brb	WR	WR	Brb	Brb	WR	WR	WL	WL	WR	WR	WL	WL	WL	WR	
	L	Brb	Brb	WR	WR	Brb	Brb	WR	WR	Rnd	Rnd	WR	WR	WL	WL	WL	WR	
	H	Esc	Esc	WR	WR	Rnd	Rnd	WR	WR	Rnd	Rnd	WR	WR	WL	WL	WL	WR	
	vH	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	WL	WL	WL	WR	
H	vL	R	R	WR	WR	R	R	Pos	WR	R	R	Pos	WR	WL	WL	WL	WR	
	L	R	R	WR	WR	R	R	Pos	WR	R	R	Pos	WR	WL	WL	WL	WR	
	H	R	R	Pos	WR	R	R	Pos	WR	R	R	Pos	WR	WL	WL	WL	WR	
	vH	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	Esc	Esc	Esc	WR	WL	WL	WL	WR	
vH	vL	R	R	R	WR	R	R	R	WR	R	R	R	WR	R	R	R	WR	
	L	R	R	R	WR	R	R	R	WR	R	R	R	WR	R	R	R	WR	
	H	R	R	R	WR	R	R	R	WR	R	R	R	WR	R	R	R	WR	
	vH	R	R	R	WR	R	R	R	WR	R	R	R	WR	R	R	R	WR	

Fig. 10. Action selection matrix with $4 \times 4 \times 4 \times 4 = 256$ segments for the 4-dimensional state space of the 4 "Drives", **Fa** Fatigue, **Hu** Hunger, **Ho** Homesickness and **Cu** Curiosity. Each segment contains one of the 10 implemented behaviour primitives.

The learning action selection was first tested with the real robot system using just the internal value Hunger and the two behaviour primitives Braitenberg (Bbr) and Wallfollowing Right (WR) using automatic reinforcement, and a moderate exploitation. The

priority values were initialized using random values between 95 and 100 for each action in each box. Now, every 30 seconds the current action in the current box was punished, while after every discovery of Food the action in the box encountered 1.5 seconds before, was rewarded. After a run of 20 minutes, the values of the actions had changed to favour Braitenberg only in the case of a very low Hunger value and Wallfollowing Right elsewhere. In Fig. 11 the initial priority values and the final depicted during the reinforcement based learning process. The resulting 1-dimensional action selection matrix is shown below for a winner takes all strategy.



Part of the action Selection Matrix

Fig. 11. The priority values are changed during the learning process using the reinforcement signal. Depicted is a part of the action selection matrix, where the active behaviour changes from Wall Following right (WR) to Braitenberg movement (Brb).

By this, the learning mechanism for matrices has been established. Now the secondary internal values can be used to generate a reinforcement signal and thus fill a randomly initialised matrix with action selection preferences. Further work is necessary to define adequate dependencies between the internal value system ("Emotion" values) and the reinforcement signal. Experiments conducted with the robot and with a human operator yielded changing action selection priority values and thus changing action selection matrices. One could easily "train" the robot to avoid certain actions, just by pure negative reinforcement (approx 5-10 minutes of training). More complex behaviour could be obtained by longer and more sophisticated reinforcement signals coming from the human operator. Due to the large action selection matrix (256 boxes) and the far larger priority value matrix (256 x 10) a completely human trained action matrix robot was omitted.

7. Conclusion

Getting autonomous robots to do the things we want them to do is still a challenge. Even the definition of parameters for a given controller type is very hard or realise for interesting robotic tasks. Designing controllers that are easily configured in an adequate way is far more complicated. The idea to make a controller learn an adequate set of parameters and functions for a given task is not completely new, but still not solved sufficiently. Biological entities demonstrate that it is in principle possible to realise such a system.

Based on these ideas, a hierarchical robot control structure, with three main information processing branches: a sensory upstream, an actuary downstream, and a controller was developed. The controller consists of two main units: an internal value system (IVS) and a learning action controller. The internal values (Drives, Emotions) are fed with the results from the sensory upstream. Several (17) primary and virtual sensor modules have been implemented within the different layers of the sensory upstream. The internal values are the decision basis for the learning action controller. The action controller activates the different action modules within the hierarchy of the actuary downstream. We have implemented a total of 14 action modules that realise basic, simple and complex motor actions, including several senso-motoric closed loop controlled behaviours.

The developed internal value system is aligned with psychological terms like "Drives" and "Emotions", without attempting to model these psychological concepts. The implemented internal values, like "Hunger", "Fatigue", "Fear", "Curiosity" or "Homesickness" are just meant as labels to indicate their specific functionality.

The action selection unit has been implemented as a matrix implementation of a switching controller. The content of the matrix is a result of pre-design and subsequent learning. Training the action selection is performed using part of the internal values as a reinforcement signal, as well as human teacher generated reinforcement.

The different realisation of the EMOBOT approach, with different kind of robots (grid world based, real value based and a real robot system) and with different set of internal values ("Drives" and "Emotions") show that the idea of controlling a robot with a set of meta-values, that are not directly connected to the outer world is a feasible approach to generate a complex behaviour.

For a given configuration of the behaviour primitives and the action selection and the internal value system it is difficult to predict the overall behaviour of the robot system. The other way round, to construct a special set-up with the goal to create a specific behaviour is far more complicated, but on at the same time far more interesting.

The results obtained, and partially presented within this work show clearly that the approach of learning hierarchical action selection based on internal values is a valuable way of finding robust robot controllers. Although a lot of scientific questions still remain unanswered we believe that the results presented are encouraging for the future.

8. References

- Arkin, R. C. (1998). *Behaviour based robotics*. Bradford Books, MIT Press, Cambridge, Massachusetts.
- Braitenberg, V.(1985). *Vehicles: Experiments in Synthetic Psychology*, The MIT Press, Cambridge, Massachusetts; London, England.

- Brooks, R. (1985) *A robust layered control system for a mobile robot*, Technical Report: AIM-864, Massachusetts Institute of Technology, Cambridge, MA, USA.
- Brooks, R. (1986), *A Robust Layered Control System for a Mobile Robot*, IEEE Journal of Robotics and Automation, Vol. 2, No. 1, March 1986, pp. 14-23;
- Burghouts, G.J.; Heylen, D.; Poel, M.; op den Akker, R.; Nijholt, A. (2003). An Action Selection Architecture for an Emotional Agent. In *Proc. of 16th International Florida Artificial Intelligence Research Society Conference*, St. Augustine, Florida, USA.
- Canamero D (1997) Modeling motivations and emotions as a basis for intelligent behavior , In *Proc. First Int. Symp. on Aut. Agents, AA'97*, The ACM Press.
- Cos I, Hayes G (2002) Behaviour Control Using a Functional and Emotional Model. In *From animals to animats 7: Proc. 7th International Conference on Simulation of Adaptive Behaviour*. pp. 226-227, MIT Press, Massachusetts, USA.
- Damasio, A.R. (1994). *Descartes' Error: emotion, reason and the human brain* Robot, New York, USA: Picador, 1994
- Gadano, S.C.; Hallam, J. (1998a). Emotion-driven Learning for Animat Control. In *From animals to animats 5: Proc. 5th International Conference on Simulation of Adaptive Behaviour*, pp. 354-359, MIT Press, Massachusetts, USA.
- Gadano, S.C.; Hallam, J. (1998b). Exploring the Role of Emotions in Autonomous Robot Learning. In *AAAI Fall Symposium - Emotional and Intelligent: The tangled knot of cognition*. Also DAI Research Paper 909, University of Edinburgh.
- Gadano, S.C.; Custodio, L. (2002) Asynchronous Learning by Emotions and Cognition. In *From animals to animats 7: Proc 7th International Conference on Simulation of Adaptive Behaviour*. pp. 224-225, MIT Press, Massachusetts, USA.
- Goerke, N. (2001). Perspectives for the next decade of neural computation, In: *Proc. of NATO Advanced Research Workshop on: Limitations and Future Trends in Neural Computation (LFTNC'01)*, pp. 1-9, Siena, Italy.
- Goerke, N., (2003). Systemic Intelligence: Methods for Growing up Artefacts that Live', In: *Proceedings of the Int. Joint Conf. on Neural Networks, IJCNN03*, Portland, Oregon, USA, 2003.
- Goerke, N.; Müller, J. (2003), Emotions are driving a robot, In: *Proceedings of the 8th Engineering Application of Neural Networks Conference (EANN03)*, pp. 56-63, University of Malaga, Malaga, Spain..
- Goerke, N.; Henne, T.; Müller, J. (2004) *Neural networks for the EMOBOT robot control architecture*, Neural Computing and Applications, Vol. 13, No. 4, pp. 299-308, Springer, Heidelberg.
- Kaelbling, L.P.; Littman, M.L.; Moore, A.W. (1996). *Reinforcement Learning: A Survey*, J. Artificial Intelligence Research, 4:237-285, Morgan Kaufmann.
- Kintzler, F. (2002) *Entwicklung, Implementation und Test einer Kontroll-Architektur für aufwachsende Animaten*, Diploma-thesis, Dept. of Computer Science, University of Bonn: Bonn, Germany.
- Kintzler, F. (2003). Layout and first tests of a systemic architecture for growing up artefacts that live, In: *Proceedings of the 8th Engineering Application of Neural Networks Conference (EANN03)*, pp. 72-79, University of Malaga, Malaga, Spain.
- McFarland, D. (1999). *Animal Behaviour*. Addison Wesley, Longman, Harlow.
- Murphy, R. (2000). *An Introduction to AI Robotics: Intelligent Robotics and Autonomous Agents*, The MIT Press, Cambridge, Massachusetts; London, England.

- Siegwart, R.; Nourbakhsh, I.R.. (2004) *Introduction to Autonomous Mobile Robots*, A Bradford Book - The MIT Press, Cambridge, Massachusetts; London, England.
- Skarmeta, A. G.; Barberà, H.M.; Alonso, M.S. (1999) *Learning Behaviour Fusion in Autonomous Mobile Robots*. ESTYLF'99, London, England.
- Sutton, R.S. Barto, A.G. (1998). *Reinforcement Learning*, The MIT Press, Cambridge, Massachusetts; London, England.
- Velásquez, J.D. (1997). When Robots Weep: Emotional Memories and Decision-Making, In *Proc. 15th National Conference on Artificial intelligence*, AAAI Press, Madison, Wisconsin, USA.

Mobile Robotics, Moving Intelligence

Souma Alhaj Ali*; Masoud Ghaffari, Xiaoqun Liao and Ernest Hall

**The Hashemite University, Jordan
University of Cincinnati, U.S.A.*

1. Introduction

After proving to be an efficient tool for improving quality, productivity, and competitiveness of manufacturing organizations, robots now expand to service organizations, offices, and even homes. Global competition and the tendency to reduce production cost and increase efficiency creates new applications for robots that stationary robots can't perform. These new applications require the robots to move and perform certain activities at the same time. The availability and low cost of faster processors, better programming, and the use of new hardware allow robot designers to build more accurate, faster, and even safer robots. For example, Egemin Automation has been selling mobile robots for a number of years, available as a specialized unit that delivers mail within a large building to warehouse automation systems. Currently, mobile robots are expanding outside the confines of buildings and into rugged terrain, as well as familiar environments like schools and city streets (Wong, 2005).

The problem addressed in this paper is to describe the theory and architecture of robust learning for mobile robots and to illustrate the theory for designing intelligent mobile robots for a wide variety of applications anywhere in the world including complex and uncertain environments. The proposed architecture for machine learning is also based on the perceptual creative controller for an intelligent robot that uses a multi-modal adaptive critic for performing learning in an unsupervised situation but can also be trained for tasks in another mode and then is permitted to operate autonomously. The robust nature is derived from the automatic changing of modes based on internal measurements of error at appropriate locations in the controller.

2. Types of Mobile Robots

Many different types of mobile robots had been developed depending on the kind of application, velocity, and the type of environment whether its water, space, terrain with fixed or moving obstacles. Four major categories had been identified (Dudek & Jenkin, 2000):

Terrestrial or ground-contact robots: The most common ones are the wheeled robots; others are the tracked vehicles and Limbed vehicles.

Aquatic robots: Those operate in water surface or underwater. Most use water jets or propellers.

Airborne robots: Flying robots like Robotic helicopters, fixed-wing aircraft, robotically controlled parachutes, and dirigibles.

Space robots: Those are designed to operate in the microgravity of outer space and are typically envisioned for space station maintenance. Space robots either move by climbing or are independently propelled.

2.1 Terrestrial or Ground-contact Robots

There are three main types of ground-contact robots: wheeled robots, tracked vehicles, and limbed vehicles. Wheeled robots exploit friction or ground contact to enable the robot to move. Different kinds of wheeled robots exist: the differential drive robot, synchronous drive robot, steered wheels robots and Ackerman steering (car drive) robots, the tricycle, bogey, and bicycle drive robots, and robots with complex or compound or omnidirectional wheels. Army Research Labs with NIST support demonstrated defence robot vehicles, called Experimental Unmanned Vehicles (XUV) that developed at a cost of approximately \$50 million over four years. It used technology from leading robotics laboratories in the US and Germany. XUV performed autonomous scout missions in difficult off-road terrain and run with general goal points and mission profiles given by Army scouts. XUV navigates through woods and fields to find and report enemy targets. In a demonstration, the XUVs drove autonomously over difficult terrain including dirt roads, trails, tall grass, weeds, brush, and woods. Using on board sensors, the XUVs were able to detect and avoid both positive and negative obstacles. The Demo III XUVs have repeatedly navigated kilometres of difficult off-road terrain with only high-level mission commands provided by an operator from a remote location. Figure 1 shows the XUV in action at Ft. Indiantown Gap (Alhaj Ali, 2003, Greenhouse & Norris, 2002). Another example is Bearcat III which was developed in the University of Cincinnati Robotics Center and is shown in Fig. 2. It has two driven wheels and a caster wheel. The driven wheels are of a fixed-wheel type.



Fig. 1. Experimental Unmanned Vehicle in action at Ft. Indiantown Gap. Photo courtesy of the Army Research Labs (Greenhouse & Norris, 2002).



Fig. 2. University of Cincinnati Bearcat III.

For more information on the construction, kinematic, and dynamic models of wheeled mobile robots, refer to Alhaj Ali (2003), de Wit, et al. (1996), and Dudek & Jenkin, (2000).

Tracked vehicles are robust to any terrain environment, their construction are similar to the differential drive robot but the two differential wheels are extended into treads which provide a large contact area and enable the robot to navigate through a wide range of terrain.

Limbed vehicles are suitable in rough terrains such as those found in forests, near natural or man-made disasters, or in planetary exploration, where ground contact support is not available for the entire path of motion. Limbed vehicles are characterized by the design and the number of legs, the minimum number of legs needed for a robot to move is one, to be supported a robot need at least three legs, and four legs are needed for a statically stable robot, six, eight, and twelve legs robots exists. For example of a limbed robot, ASIMO which is a humanoid two legs walking robot developed by Honda and features the ability to pursue key tasks in a real-life environment such as an office, ASIMO has an advanced level of physical capabilities. The new ASIMO model is shown in Fig. 3 (Honda, 2006).



Fig. 3. The new ASIMO model (Honda, 2006).

2.2 Aquatic Robots

Aquatic vehicles support propulsion by utilizing the surrounding water. There are two common structures (Dudek & Jenkin, 2000): torpedo-like structures (Ferguson & Pope, 1995, Kloske et al., 1993) where a single propeller provides forward, and reverse thrust while the navigation direction is controlled by the control surfaces, the buoyancy of the vessel controls the depth. The disadvantage of this type is poor manoeuvrability.

Twin-burger (Fuji et al., 1939) and URV vehicles (Choi, 1993, Choi et al., 1995) robots which uses a collection of thrusters that are distributed over the vessel, more manoeuvrable are attained by controlling sets of the thrusters to change the vehicle orientation and position

independently, however, the URV comes with the expense of operational speed. An example of an aquatic robot is shown in Fig. 4.a.

2.3 Flying Robots

Fixed-wing autonomous vehicles: This utilizes control systems very similar to the ones found in commercial autopilots. Ground station can provide remote commands if needed, and with the help of the Global Positioning System (GPS) the location of the vehicle can be determined (Dudek & Jenkin, 2000).

Automated helicopters (Baker et al., 1992, Lewis et al., 1993): These use onboard computation and sensing and ground control, their control is very difficult compared to the fixed-wing autonomous vehicles.

Buoyant (aerobots, aerovehicles, or blimps) vehicles: These vehicles can float and are characterized by having high energy efficiency ration, long-range travel and duty cycle, vertical mobility, and they usually has no disastrous results in case of failure (Dudek & Jenkin, 2000).

Unpowered autonomous flying vehicles: These vehicles reach their desired destination by utilizing gravity, GPS, and other sensors (Dudek & Jenkin, 2000). An example of a flying robot is shown in Fig. 4.b.



Fig. 4. Aquatic and Flying vehicle (SSC San Diego, 2002).

2.4 Space Robots

These are needed for applications related to space stations like construction, repair, and maintenance. Free-flying systems have been proposed where the spacecraft is equipped with thrusters with one or more manipulators, the thrusters are utilized to modify the robot trajectory. For more information on the construction and kinematic models of mobile robots please refer to (Dudek & Jenkin, 2000).

3. Navigation

Navigation is the major challenge in the autonomous mobile robots; a navigation system is the method for guiding a vehicle. Several capabilities are needed for autonomous navigation (Alhaj Ali, 2003):

- The ability to execute elementary goal achieving actions such as going to a given location or following a leader;
- The ability to react to unexpected events in real time such as avoiding a suddenly appearing obstacle;

- The ability to formulate a map of the environment;
- The ability to learn which might include noting the location of an obstacle and of a three-dimensional nature of the terrain and adapt the drive torque to the inclination of hills (Golnazarian & Hall, 2000).

During the last fifteen years, a great deal of research has been done on the interpretation of motion fields as the information they contain about the 3-D world. In general, the problem is compounded by the fact that the information that can be derived from the sequence of images is not the exact projection of the 3D-motion field, but rather information about the movement of light patterns and optical flow (Golnazarian & Hall, 2000, Alhaj Ali, 2003).

3.1 Systems and Methods for Mobile Robot Navigation

Odometry and other dead-reckoning methods: These methods use encoders to measure wheel rotation and/or steering orientation (Alhaj Ali, 2003).

Vision based navigation: Computer vision and image sequence techniques were proposed for obstacle detection and avoidance for autonomous land vehicles that can navigate in an outdoor road environment. The object shape boundary is first extracted from the image, after the translation from the vehicle location in the current cycle to that in the next cycle, the position of the object shape in the image of the next cycle is predicted, then it is matched with the extracted shape of the object in the image of the next cycle to decide whether the object is an obstacle (Alhaj Ali, 2003, Chen & Tsai, 2000).

Sensor based navigation: Sensor based navigation systems that rely on sonar or laser scanners that provide one dimensional distance profiles have been used for collision and obstacle avoidance. A general adaptable control structure is also required. The mobile robot must make decisions on its navigation tactics; decide which information to use to modify its position, which path to follow around obstacles, when stopping is the safest alternative, and which direction to proceed when no path is given. In addition, sensors information can be used for constructing maps of the environment for short term reactive planning and long-term environmental learning (Alhaj Ali, 2003).

Inertial navigation: This method uses gyroscopes and sometimes accelerometers to measure the rate of rotation and acceleration.

Active beacon navigation systems: This method computes the absolute position of the robot from measuring the direction of incidence of three or more actively transmitted beacons. The transmitters, usually using light or radio frequencies must be located at known sites in the environment (Janet, 1997, Premvuti & Wang, 1996, Alhaj Ali, 2003).

Landmark navigation: In this method distinctive artificial landmarks are placed at known locations in the environment to be detected even under adverse environmental conditions (Alhaj Ali, 2003).

Map-based positioning: In this method information acquired from the robot's onboard sensors is compared to a map or world model of the environment. The vehicle's absolute location can be estimated if features from the sensor-based map and the world model map match (Alhaj Ali, 2003).

Biological navigation: biologically-inspired approaches were utilized in the development of intelligent adaptive systems; biomimetic systems provide a real world test of biological navigation behaviours besides making new navigation mechanisms available for indoor robots (Alhaj Ali, 2003).

Global positioning system (GPS): This system provides specially coded satellite signals that can be processed in a GPS receiver, enabling it to compute position, velocity, and time (Alhaj Ali, 2003).

4. Mobile Robots and Artificial Intelligence

While robotics research has mainly been concerned with vision (eyes) and tactile, some problems regarding adapting, reasoning, and responding to changed environment have been solved with the help of artificial intelligence using heuristic methods such as ANN. Neural computers have been suggested to provide a higher level of intelligence that allows the robot to plan its action in a normal environment as well as to perform non-programmed tasks (Golnazarian & Hall, 2000, Alhaj Ali, 2003).

A well established field in the discipline of control systems is the intelligent control, which represents a generalization of the concept of control, to include autonomous anthropomorphic interactions of a machine with the environment (Alhaj Ali, 2003, Meystel & Albus, 2002). Meystel and Albus (2002) defined intelligence as "the ability of a system to act appropriately in an uncertain environment, where an appropriate action is that which increases the probability of success, and success is the achievement of the behavioural sub goals that support the system's ultimate goal". The intelligent systems act so as to maximize this probability. Both goals and success criteria are generated in the environment external to the intelligent system. At a minimum, the intelligent system had to be able to sense the environment which can be achieved by the use of sensors, then perceive and interpret the situation in order to make decisions by the use of analyzers, and finally implements the proper control actions by using actuators or drives. Higher levels of intelligence require the abilities to: recognize objects and events store and use knowledge about the world, learn, and to reason about and plan for the future. Advanced forms of intelligence have the ability to perceive and analyze, to plan and scheme, to choose wisely, and plan successfully in a complex, competitive, and hostile world (Alhaj Ali, 2003).

Intelligent behaviour is crucial to mobile robots; it could be supported by connecting perception to action (Kortenkamp et al., 1998). In the following a brief review for the literature in the use of artificial intelligence in mobile robots will be presented (Alhaj Ali, 2003).

4.1 Use of Artificial Neural Networks (ANN)

ANN has been applied to mobile robot navigation. It had been considered for applications that focus on recognition and classification of path features during navigation. Kurd and Oguchi (1997) propose the use of neural network controller that was trained using supervised learning as an indirect-controller to obtain the best control parameters for the main controller in use with respect to the position of the AGV. A method that uses incremental learning and classification based on a self-organizing ANN is described by Vercelli and Morasso (1998). Xue and Cheung (1996) proposed a neural network control scheme for controlling active suspension. The presented controller used a multi-layer neural network and a prediction-correction method for adjusting learning parameters. Dracopoulos (1998) present the application of multi-layer perceptrons to the robot path planning problem and in particular to the task of maze navigation. Zhu, et al. (1998) present results of integrating omni-directional view image analysis and a set of adaptive networks to understand the outdoor road scene by a mobile robot (Alhaj Ali, 2003).

To navigate and recognize where it is, a mobile robot must be able to identify its current location. The more the robot knows about its environment, the more efficiently it can operate (Cicirelli, 1998). Grudic and Lawrence (1998) used a nonparametric learning algorithm to build a robust

mapping between an image obtained from a mobile robot's on-board camera, and the robot's current position. It used the learning data obtained from these raw pixel values to automatically choose a structure for the mapping without human intervention, or any prior assumptions about what type of image features should be used (Alhaj Ali, 2003).

4.2 Use of Fuzzy Logic

Fuzzy logic and fuzzy languages have also been used in navigation algorithms for mobile robots as described in (Wijesoma et al., 1999, Mora and Sanchez, 1998). Lin and Wang (1997) propose a fuzzy logic approach to guide an AGV from a starting point toward the target without colliding with any static obstacle as well as moving obstacles; they also study other issues as sensor modelling and trap recovery. Kim and Hyung (1998) used fuzzy multi-attribute decision-making in deciding which via-point the robot should proceed to at each step. The via-point is a local target point for the robot's movement at each decision step. A set of candidate via-points is constructed at various headings and velocities. Watanabe, et al. (1998) described a method using a fuzzy logic model for the control of a time varying rotational angle in which multiple linear models are obtained by utilizing the original non-linear model at some representative angles (Alhaj Ali, 2003).

4.3 Use of Neural Integrated Fuzzy Controller

A neural integrated fuzzy controller (NiF-T) that integrates the fuzzy logic representation of human knowledge with the learning capability of neural networks has been developed for nonlinear dynamic control problems (Alhaj Ali, 2003). Daxwanger and Schmidt (1998) presented their neuro-fuzzy approach for visual guidance of a mobile robot vehicle.

5. Navigation in Unstructured Environments

Some research has been conducted regarding robotics in unstructured environment. Alhaj Ali (2003) presented the development of an autonomous navigation and obstacle avoidance system for a wheeled mobile robot operating in unstructured outdoor environments. The algorithm presented produces the robot's path positioned within the road boundaries and avoids any fixed obstacles along the path. The navigation algorithm was developed from a feedforward multilayer neural network. The network used a quasi-Newton backpropagation algorithm for training. Computed-torque, digital, and adaptive controllers were developed to select suitable control torques for the motors, which cause the robot to follow the desired path from the navigation algorithm.

Martínez and Torras (2001) presented visual procedures especially tailored to the constraints and requirements of a legged robot that works with an un-calibrated camera, with pan and zoom, freely moving towards a stationary target in an unstructured environment that may contain independently-moving objects.

Kurazume and Hirose (2000) proposed a new method called Cooperative Positioning System (CPS). The main concept of CPS is to divide the robots into two groups, A and B, where group A remains stationary and acts as a landmark while group B moves. Then group B stops and acts as a landmark for group A. This process is repeated until the target position is reached. Their application was a floor-cleaning robot system. Torras (1995) reviewed neural learning techniques for making robots well adapted to their surroundings.

Yahja et al. (2000) propose an on-line path planner for outdoor mobile robots using a framed-quadtrees data structure and an optimal algorithm to incrementally re-plan optimal paths. They showed that the use of framed-quadtrees leads to paths that are shorter and more direct compared to the other representations; however, their results indicate that starting with partial information is better than starting with no information at all. Baratoff et al. (2000) designed a space-variant image transformation, called the polar sector map, which is ideally suited to the navigational tasks.

Yu et al. (2001) presented a hybrid evolutionary motion planning simulation system for mobile robots operating in unstructured environments, based on a new obstacle representation method named cross-line, a follow boundary repair approach, and a hybrid evolutionary motion planning algorithm. Yan et al. (1997) presents an attempt to devise and develop a domain-independent reasoning system scheme for handling dynamic threats, and uses the scheme for automated route planning of defence vehicles in an unstructured environment (Alhaj Ali, 2003).

Computer vision and image sequence techniques were proposed for obstacle detection and avoidance for autonomous land vehicles that can navigate in an outdoor road environment. Jarvis (1996) report some preliminary work regarding an autonomous outdoor robotic vehicle navigation using flux gate compass, DGPS and range sensing, and distance transform based path planning. An adaptive navigation method suited for the complex natural environments had been proposed based on a multi-purpose perception system that manages different terrain representations, the method focuses on the functions that deal with the navigation planning and the robot self-localization which have been integrated within the robot control system (Devy, 1995).

Krishna and Kalra (2001) proposed incorporating cognition and remembrance capabilities in a sensor-based real-time navigation algorithm, they stated that these features enhance the robots performance by providing a memory-based reasoning whereby the robot's forthcoming decisions are also affected by its previous experiences during the navigation, which is apart from the current range inputs, the suggested robot navigates in a concave maze-like unstructured altered environment which has been modelled by classifying temporal sequences of special sensory patterns.

Marco et al. (1996) developed a hybrid controller for semi-autonomous and autonomous underwater vehicles in which the missions imply multiple task robot behaviour. They proposed the use of Prolog as a computer language for the specification of the discrete event system (DES) aspects of the mission control, and made the connections between a Prolog specification and the more common Petri Net graphical representation of a DES (Alhaj Ali, 2003).

6. Controllers for Autonomous Mobile Robots

Robots and robots manipulators have complex nonlinear dynamics that make their accurate and robust control difficult. On the other hand, they fall in the class of Lagrangian dynamical systems, so that they have several extremely nice physical properties that make their control straight forwarded (Lewis et al., 1999). Different controllers had been developed for the motion of robot manipulators, however, not until recently where there has been an interest in moving the robot itself, not only its manipulators (Alhaj Ali, 2003).

Shim and Sung (2003) proposed a WMR asymptotic control with driftless constraints based on empirical practice using the WMR kinematic equations. They showed that with the appropriate selection of the control parameters, the numerical performance of the

asymptotic control could be effective. The trajectory control of a wheeled inverse pendulum type robot had been discussed by Yun-Su and Yuta (1998), their control algorithm consists of balance and velocity control, steering control, and straight line tracking control for navigation in a real indoor environments (Alhaj Ali, 2003).

Rajagopalan and Barakat (1997) developed a computed torque control scheme for Cartesian velocity control of WMRs. Their control structure can be used to control any mobile robot if its inverse dynamic model exists. A discontinuous stabilizing controller for WMRs with nonholonomic constraints where the state of the robot asymptotically converges to the target configuration with a smooth trajectory was presented by Zhang and Hirschorn (1997). A path tracking problem was formulated by Koh and Cho (1999) for a mobile robot to follow a virtual target vehicle that is moved exactly along the path with specified velocity. The driving velocity control law was designed based on bang-bang control considering the acceleration bounds of driving wheels and the robot dynamic constraints in order to avoid wheel slippage or mechanical damage during navigation. Zhang et al. (2003) employed a dynamic modelling to design a tracking controller for a differentially steered mobile robot that is subject to wheel slip and external loads (Alhaj Ali, 2003).

A sliding mode control was used to develop a trajectory tracking control in the presence of bounded uncertainties (Corradini & Orlando, 2001). A solution for the trajectory tracking problem for a WMR in the presence of disturbances that violate the nonholonomic constraint is proposed later by the same authors based on discrete-time sliding mode control (Corradini et al., 2002).

An electromagnetic approach for path guidance of a mobile-robot-based automatic transport service system with a PD control algorithm was investigated by Wu et al. (2001). Jiang, et al. (2001) developed a model-based control design strategy that deals with global stabilization and global tracking control for the kinematic model with a nonholonomic WMR in the presence of input saturations. An adaptive robust controller was proposed for the global tracking problem for the dynamic of the non-holonomic systems with unknown dynamics (Dong, 1999). However, real time adaptive controls are not common in practical applications due partly to stability problems (Werbos, 1999, Alhaj Ali, 2003).

A fuzzy logic controller had been tried for WMRs navigation. Montaner and Ramirez-Serrano (1998) developed a fuzzy logic controller that can deal with the sensors inputs uncertainty and ambiguity for direction and velocity manoeuvres. A locomotion control structure was developed based on the integration of an adaptive fuzzy-net torque controller with a kinematic controller to deal with unstructured unmodeled robot dynamics for a non-holonomic mobile robot cart (Topalov, 1998). Toda et al. (1999) employed a sonar-based mapping of crop rows and fuzzy logic control-based steering for the navigation of a WMR in an agricultural environment. They constructed a crop row map from the sonar readings and transferred it to the fuzzy logic control system, which steers the robot along the crop row. A local guidance control method for wheeled mobile robots using fuzzy logic for guidance, obstacle avoidance and docking was proposed by Vázquez and Garcia (1994), the method provide a smooth but not necessary optimal solution (Alhaj Ali, 2003).

7. Creative Control for Intelligent Mobile Robots

Since mobile robots must be able to select among many concurrent tasks, such as moving while sensing and reacting and learning, a new control structure is needed. The creative controller concept is excellent for complex environments since it permits a divide and conquer approach,

with one or more tasks at a time in a multi-threaded, distributed computing environment. Creative learning architectures integrate a Task Control Center (TCC) and a dynamic database (DD) and adaptive critic learning algorithms to permit these solutions. Determining the task to be performed and the data base to be updated are the two key elements of the design. These new decision processes encompass both decision and estimation theory and can be modeled by neural networks and implemented with multi-threaded computers.

The control architectures for neural network control of vehicles in which the kinematic and dynamic models are known but one or more parameters must be estimated is a simple task that has been demonstrated. The mathematical models for the kinematics and dynamics were developed and the main emphasis was to explore the use of neural network control and demonstrate the advantages of these learning methods. The results indicate the method of solution is appropriate and that it has potential application to a large number of currently unsolved problems in complex environments. The adaptive critic neural network control is an important starting point for future learning theories that are applicable to robust control and learning situations.

To obtain broadly applicable results, a generalization of adaptive critic learning called Creative Control (CC) for intelligent robots in complex, unstructured environments has been used. The creative control learning architecture integrates a Task Control Center (TCC) and a Dynamic Knowledge Database (DKD) with adaptive critic learning algorithms. Recently learning theories such as the adaptive critic have been proposed in which a critic provides a grade to the controller of an action module such as a robot. The creative control process is used that is "beyond the adaptive critic". A mathematical model of the creative control process is presented that illustrates the use for mobile robots.

7.1 Dynamic Programming

The intelligent robot is defined as a decision maker for a dynamic system that may make decisions in discrete stages or over a continuous or discrete time horizon. The outcome of each decision may not be fully predictable but may be anticipated or estimated to some extent before the next decision is made. Furthermore, an objective or cost function can be defined for the decision. There may also be natural constraints. Generally, the goal is to minimize this cost function over some decision space subject to the constraints. With this definition, the intelligent robot can be considered as a set of problems in dynamic programming and optimal control (Liao, 2002).

Dynamic programming (DP) is the only approach for sequential optimization applicable to general nonlinear, stochastic environments. However, DP needs efficient approximate methods to overcome its dimensionality problems. It is only with the presence of artificial neural network (ANN) and the invention of back propagation that such a powerful and universal approximate method has become a reality.

The essence of dynamic programming is Bellman's *Principle of Optimality*. (Bellman 1957)

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

The original Bellman equation of dynamic programming for adaptive critic algorithms may be written as shown in Eq (1):

$$J(R(t)) = \max_{u(t)} (J(R(t), u(t)) + \gamma (J(R(t+1)) - J(R(t)))) / (1 + r) - U_0 \quad (1)$$

where $R(t)$ is the model of reality or state form, $U(R(t), u(t))$ is the utility function or local cost, $u(t)$ is the action vector, $J(R(t))$ is the criteria or cost-to-go function at time t , r and U_0 are constants that are used only in infinite-time-horizon problems and then only sometimes, and where the angle brackets refer to expected value.

The user provides a utility function, U , and a stochastic model of the plant, R , to be controlled. The expert system then tries to solve the Bellman equation for the chosen model and utility function to achieve the optimum value of J by picking the action vector $u(t)$. If an optimum J cannot be determined, an approximate or estimate value of the J function is used to obtain an approximate optimal solution. Regarding the finite horizon problems, which we normally try to cope with, one can use Eq (2):

$$J(R(t)) = \max_{u(t)} (U(R(t), u(t)) + \gamma J(R(t+1))) / (1 + r) \quad (2)$$

Dynamic programming gives the exact solution to the problem of how to maximize a utility function $U(R(t), u(t))$ over the future times, t , in a nonlinear stochastic environment. Dynamic programming converts a difficult long-term problem in optimization over time $\langle U(R(t)) \rangle$, the expected value of $U(R(t))$ over all the future times, into a much more straightforward problem in simple, short-term function maximization after we know the function J . Thus, all of the approximate dynamic programming methods discussed here are forced to use some kind of general-purpose nonlinear approximation to the J function, the value function in the Bellman equation, or something closely related to J .

In most forms of adaptive critic design, we approximate J by using a neural network. Therefore, we approximate $J(R)$ by some function $\hat{J}(R, W)$, where W is a set of weights or parameters, and \hat{J} is called a Critic network. If the weights W are adapted or iteratively solved for, in real time learning or offline iteration, we call the Critic an Adaptive Critic.

An adaptive critic design (ACD) is any system which includes an adapted critic component; a critic, in turn, is a neural net or other nonlinear function approximation which is trained to converge to the function $J(X)$. In adaptive critic learning or designs, the critic network learns to approximate the cost-to-go or strategic utility function J and uses the output of an action network as one of its inputs, directly or indirectly. When the critic network learns, back propagation of error signals is possible along its input feedback to the action network. To the back propagation algorithm, this input feedback looks like another synaptic connection that needs weights adjustment. Thus, no desired control action information or trajectory is needed as supervised learning.

7. 2. Adaptive Critic and Creative Control

Most advanced methods in neurocontrol are based on adaptive critic learning techniques consisting of an action network, adaptive critic network, and model or identification network as show in Figure 5. These methods are able to control processes in such a way, which is approximately optimal with respect to any given criteria taking into consideration of particular nonlinear environment. For instance, when searching for an optimal trajectory to the target position, the distance of the robot from this target position can be used as a criteria function. The algorithm will compute the proper

steering, acceleration signals for control of vehicle, and the resulting trajectory of the vehicle will be close to optimal. During trials (the number depends on the problem and the algorithm used) the system will improve performance and the resulting trajectory will be close to optimal. The freedom of choice of the criteria function makes the method applicable to a variety of problems. The ability to derive a control strategy only from trial/error experience makes the system capable of semantic closure. These are very strong advantages of this method.

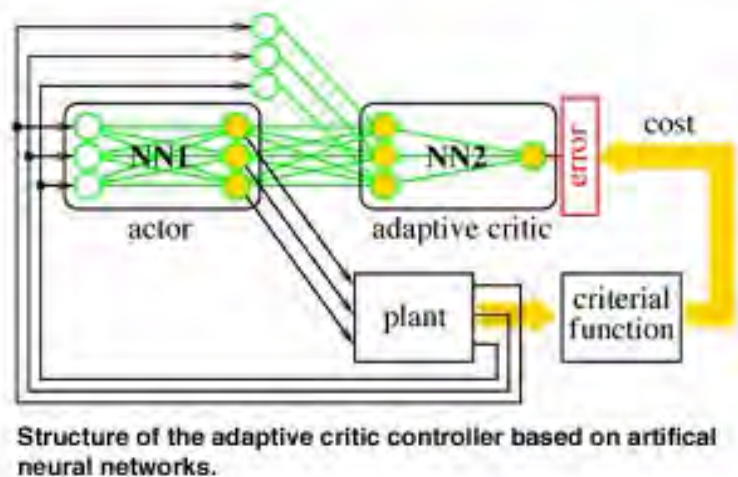


Fig. 5. Structure of the adaptive critic controller (Liao et al., 2003).

7.3 Creative Learning Structure

It is assumed that we can use a kinematic model of a mobile robot to provide a simulated experience to construct a value function in the critic network and to design a kinematic based controller for the action network. A proposed diagram of creative learning algorithm is shown in Figure 6. In this proposed diagram, there are six important components: the task control center, the dynamic knowledge database, the critic network, the action network, the model-based action and the utility function. Both the critic network and action network can be constructed by using any artificial neural networks with sigmoidal function or radial basis function (RBF). Furthermore, the kinematic model is also used to construct a model-based action in the framework of adaptive critic-action approach. In this algorithm, dynamic databases are built to generalize the critic network and its training process and provide environmental information for decision making. It is especially critical when the operation of mobile robots is in unstructured environments. Furthermore, the dynamic databases can also be used to store environmental parameters such as Global Position System (GPS) way points, map information, etc. Another component in the diagram is the utility function for a tracking problem (error measurement). In the diagram, X_k , $X_{k,d}$, $X_{k,d+1}$ are inputs and Y is the output and $J(t)$, $J(t+1)$ is the critic function with respect to time.

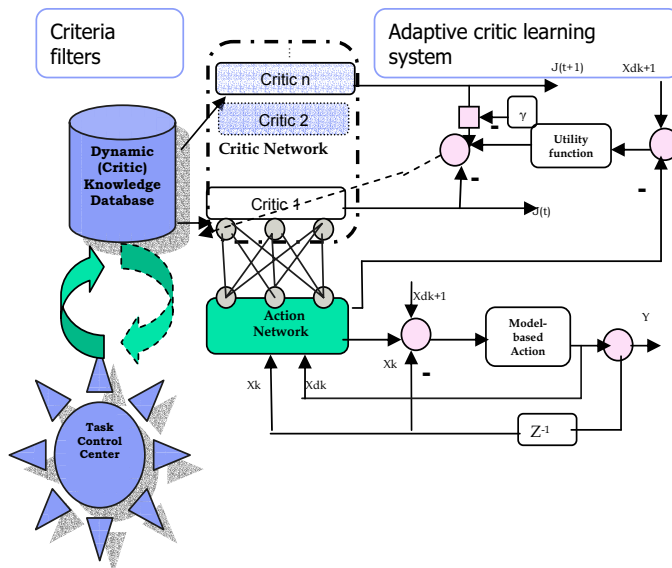


Fig. 6. Proposed creative learning algorithm structure.

7.4 Dynamic Knowledge Database (DKD)

The dynamic databases contain domain knowledge and can be modified to permit adaptation to a changing environment. Dynamic knowledge databases may be called a “neurointerface” in a dynamic filtering system based on neural networks (NNs) and serves as a “coupler” between a task control center and a nonlinear system or plant that is to be controlled or directed. The purpose of the coupler is to provide the criteria function for the adaptive critic learning system and filter the task strategies commanded by the task control center. The proposed dynamic database contains a copy of the model (or identification). Action and critic networks are utilized to control the plant under nominal operation, as well as make copies of a set of parameters (or scenario) previously adapted to deal with a plant in a known dynamic environment. The database also stores copies of all the partial derivatives required when updating the neural networks using backpropagation through time¹³. The dynamic database can be expanded to meet the requirements of complex and unstructured environments.

The data stored in the dynamic database can be uploaded to support offline or online training of the dynamic plant and provide a model for identification of nonlinear dynamic environment with its modeling function. Another function module of the database management is designed to analyze the data stored in the database including the sub-task optima, pre-existing models of the network and newly added models. The task program module is used to communicate with the task control center. The functional structure of the proposed database management system (DBMS) is shown in Figure 7. The DBMS can be customized from an object-relational database.

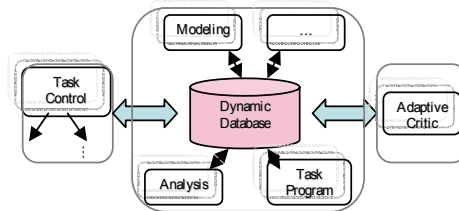


Fig. 7. Functional structure of dynamic database.

7.5 Task Control Center (TCC)

The task control center (TCC) can build task-level control systems for the creative learning system. By "task-level", we mean the integration and coordination of perception, planning and real-time control to achieve a given set of goals (tasks). TCC provides a general task control framework, and it is to be used to control a wide variety of tasks. Although the TCC has no built-in control functions for particular tasks (such as robot path planning algorithms), it provides control functions, such as task decomposition, monitoring, and resource management, that are common to many applications. The particular task built-in rules or criteria or learning J functions are managed by the dynamic database controlled with TCC to handle the allocation of resources. The dynamic database matches the constraints on a particular control scheme, sub-tasks or environment allocated by TCC.

The task control center acts as a decision-making system. It integrates domain knowledge or criteria into the database of the adaptive learning system. The task control architecture for mobile robots provides a variety of control constructs that are commonly needed in mobile robot applications, and other autonomous mobile systems. The goal of the architecture is to enable autonomous mobile robot systems to easily specify hierarchical task-decomposition strategies, such as how to navigate to a particular location, or how to collect a desired sample, or how to follow a track in an unstructured environment. This can include temporal constraints between sub-goals, leading to a variety of sequential or concurrent behaviors. TCC schedules the execution of planned behaviors, based on those temporal constraints acting as a decision-making control center.

Integrating the TCC with the adaptive critic learning system and interacting with the dynamic database, the creative learning system provides both task-level and real-time control or learning within a single architectural framework. Through interaction with human beings to attain the input information for the system, the TCC could decompose the task strategies to match the dynamic database for the rules of sub-tasks by constructing a distributed system with flexible mechanisms, which automatically provide the right data at the right time. The TCC also provides orderly access to the resources of the dynamic database with built-in learning mechanisms according to a queue mechanism. This is the inter-process communication capability between the task control center and the dynamic database. The algorithm on how to link the task control center and the dynamic database is currently done by the human designers.

7.6 Creative Learning Controller for Intelligent Robot Control

Creative learning may be used to permit exploration of complex and unpredictable environments, and even permit the discovery of unknown problems, ones that are not yet

recognized but may be critical to survival or success. By learning the domain knowledge, the system should be able to obtain the global optima and escape local optima. The method attempts to generalize the highest level of human learning – imagination. As a ANN robot controller, the block diagram of the creative controller can be presented as shown in Figure 8.

Experience with the guidance of a mobile robot has motivated this study and has progressed from simple line following to the more complex navigation and control in an unstructured environment. The purpose of this system is to better understand the adaptive critic learning theory and move forward to develop more human-intelligence-like components into the intelligent robot controller. Moreover, it should extend to other applications. Eventually, integrating a criteria knowledge database into the action module will develop a powerful adaptive critic learning module.

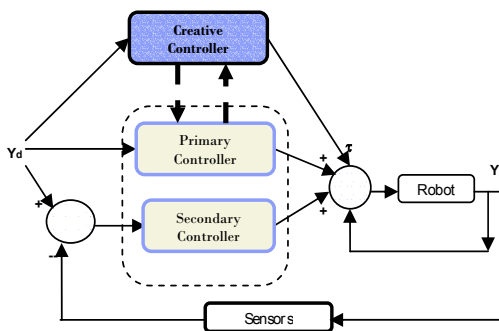


Fig. 8. Block diagram of creative controller.

A creative controller is designed to integrate domain knowledge or criteria database and the task control center into the adaptive critic neural network controller. It provides a needed and well-defined structure for autonomous mobile robot application. In effect, it replaces a human doing remote control. We have used the intelligent mobile robot as the test-bed for the creative controller.

The task control center of the creative learning system can be considered hierarchically as follows:

- * Mission for robot – e.g. mobile robot
 - * Task for robot to follow – J : task control
 - * Track for robot to follow
 - * Learn non-linear system model- model discovery
 - * Learn unknown parameters

7.7 Adaptive Critic System Implementation

Adaptive critic system and NN. In order to develop the creative learning algorithm addressed above, we have taken a bottom-up approach to implement adaptive critic controllers by first using neural network for on-line or off-line learning methods. (Campos and Lewis, 1999) Then the proposed dynamic knowledge database and task control center are added with some to be realized in future research projects.

Tuning algorithm and stability analysis. For linear time invariant systems it is straightforward to examine stability by investigating the poles in the s-plane. However, stability of nonlinear dynamic systems is much more complex, thus the stability criteria and tests are much more difficult to apply than those for linear time invariant systems (Stubberud, A.R. and S.C. Stubberud, (2000)). For general nonlinear continuous time systems, the state space model is

$$\begin{aligned}\dot{x} &= f[x(t), u(t)] \\ y &= g[x(t), u(t)]\end{aligned}\quad (8)$$

where the nonlinear differential equation is in state variable form, $x(t)$ is the state vector and $u(t)$ is the input and the second equation $y(t)$ is the output of the system.

7.8 Creative Controller and Nonlinear Dynamic System

For a creative controller, the task control center and the dynamic database are not time-variable systems; therefore, the adaptive critic learning component determines the stability of the creative controller. As it is discussed in the previous section, the adaptive critic learning is based on critic and action network designs, which are originated from artificial neural network (ANN), thus stability of the system is determined by the stability of the neural networks (NN) or convergence of the critic network and action network training procedure.

The creative controller is a nonlinear system. It is not realistic to explore all the possibilities of nonlinear systems and prove that the controller is in a stable state. We have used both robot arm manipulators and mobile robot models to examine a large class of problems known as tracking in this study. The objective of tracking is to follow a reference trajectory as closely as possible. This may also be called optimal control since we optimize the tracking error over time.

7.9 Critic and Action NN Weights Tuning Algorithm

In adaptive critic learning controller, both the critic network and the action network use multilayer NN. Multilayer NN are nonlinear in the weights V and so weight tuning algorithms that yield guaranteed stability and bounded weights in closed-loop feedback systems have been difficult to discover until a few years ago.

7.10 Urban Rescue Scenarios

Suppose a mobile robot is used for urban rescue as shown in Figure 9. It waits at a start location until a call is received from a command center. Then it must go rescue a person. Since it is in an urban environment, it must use the established roadways. Along the roadways, it can follow pathways. However, at intersections, it must choose between various paths to go to the next block. Therefore, it must use different criteria at the corners than along the track. The overall goal is to arrive at the rescue site in minimum time. To clarify the situation consider the following steps:

1. Start location – the robot waits at this location until it receives a task command to go to a certain location.
2. Along the path, the robot follows a road marked by lanes. It can use a minimum mean square error between its location and the lane location during this travel.
3. At intersections, the lanes disappear but a database gives a GPS waypoint and the location of the rescue goal.

This example requires the use of both continuous and discrete tracking, a database of known information and multiple criteria optimization. It is possible to add a large number of real-world issues including position estimation, perception, obstacles avoidance, communication, etc.

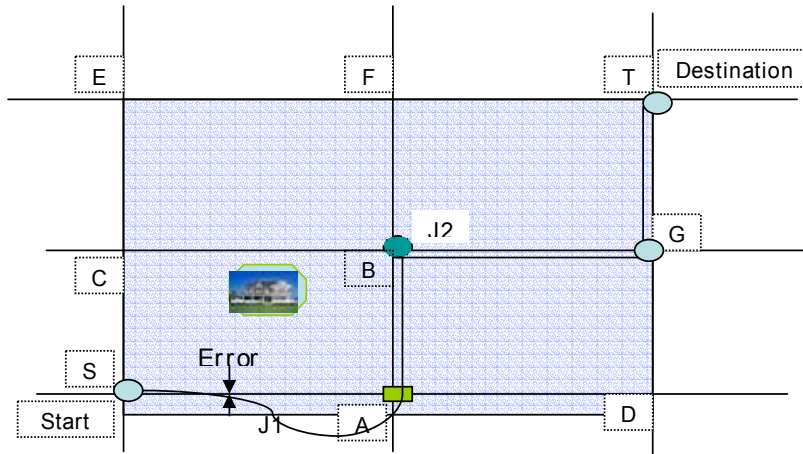


Fig. 9. Simple urban rescue site.

In an unstructured environment as shown in Figure 9, we assume that information collected about different positions of the environment could be available to the mobile robot, improving its overall knowledge. As any robot moving autonomously in this environment must have some mechanism for identifying the terrain and estimating the safety of the movement between regions (blocks), it is appropriate for a coordination system to assume that both local obstacle avoidance and a map-building module are available for the robot which is to be controlled. The most important module in this system is the adaptive system to learn about the environment and direct the robot action.

A Global Position System (GPS) may be used to measure the robot position and the distance from the current site to the destination and provide this information to the controller to make its decision on what to do at next move. The GPS system or other sensors could also provides the coordinates of the obstacles for the learning module to learn the map, and then aid in avoiding the obstacles when navigating through the intersections A, B or G, D to destination T.

Task control center. The task control center (TCC) acts a decision-making command center. It takes environmental perception information from sensors and other inputs to the creative controller and derives the criteria functions. We can decompose the robot mission at the urban rescue site shown as Figure 9 into sub-tasks as shown in Figure 10. Moving the robot between the intersections, making decisions is based on control-center-specified criteria functions to minimize the cost of mission. It's appropriate to assume that J_1 and J_2 are the criteria functions that the task control center will transfer to the learning system at the beginning of the mission from the Start point to Destination (T). J_1 is a function of t related to tracking error. J_2 is to minimize the distance of the robot from A to T since the cost is directly related to the distance the robot travels.

- From Start (S) to intersection A: robot follow the track SA with the J_1 as objective function

- From intersection A to B or D: which one will be the next intersection, the control center takes both J_1 and J_2 as objective functions.

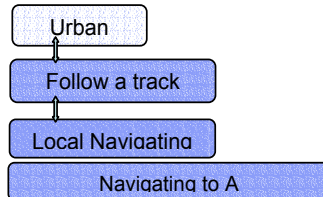


Fig. 10. Mission decomposition diagram.

Dynamic databases. Dynamic databases would store task-oriented environment knowledge, adaptive critic learning parameters and other related information for accomplishing the mission. In this scenario, the robot is commanded to reach a dangerous site to conduct a rescue task. The dynamic databases saved a copy of the GPS weight points S, A, B, C, D, E, F, G and T. The map for direction and possible obstacle information is also stored in the dynamic databases. A copy of the model parameters can be saved in the dynamic database as shown in the simplified database Figure 11. The action model will be updated in the dynamic database if the current training results are significantly superior to the previous model stored in the database.

Database fields	
Field	Description
MODEL_ID	Action model ID
MODEL_NAME	Action model name
UTILITY_FUN	Utility function
CRITERIA_FUN	Criteria function
...	...
<i>Adaptive Critic</i>	<i>Training Parameters</i>
INPUT_CRITIC	Input to critic network
DELT_J	$J(t+1)-J(t)$
...	...

Fig. 11. Semantic dynamic database structure.

Robot learning module. Initial plans such as road tracking and robot navigating based on known and assumed information can be used to incrementally revise the plan as new information is discovered about the environment. The control center will create criteria functions according to the revised information of the world through the user interface. These criteria functions along with other model information of the environment will be input to the learning system. There is a data transfer module from the control center to the learning system as well as a module from the learning system to the dynamic database. New knowledge is used to explore and learn, training according to the knowledge database information and then decide which to store in the dynamic database and how to switch the criteria. The simplest style in the adaptive critic family is heuristic dynamic programming (HDP). This is NN on-line adaptive critic learning. There is one critic network, one action network and one model network in the learning structure. $U(t)$ is the utility function. R is

the critic signal as J (criteria function). The learning structure and the parameters are saved a copy in the dynamic database for the system model searching and updating.

8. Conclusion

A new concept for mobile intelligence is presented in this paper. A series of experimental robots named the Bearcats, have been constructed at the University of Cincinnati over the past several years. This experience has evolved into our current, creative control design that is presented in this paper. Fortunately, our intelligent robots have been able to use our increasingly capable computer controls in which multi-threaded, distributed computing is now easily available.

The theory presented shows how to design intelligent robots that are capable of adapting, learning and predicting. This is a step toward understanding the semiotic closure exhibited by biological creatures and a further step toward appreciating the wonderful capabilities of human intelligence.

9. References

- Alhaj Ali, S. M. (2003). *Technologies for autonomous navigation in unstructured outdoor environments*, University of Cincinnati, Ohio.
- Baker, N. C.; MacKenzie, D. C. & Ingalls, S. A. (1992). Development of an autonomous aerial vehicle: A case study. *Applied Intelligence*, Vol. 2, No. 3, (1992) 271-298.
- Baratoff, G.; Toepfer, C. & Neumann, H. (2000). Combined space-variant maps for optical-flow-based navigation. *Biological Cybernetics*, Vol. 83, Issue 3, (Aug. 2000) 199-209.
- Bellman, R.E. (1957) *Dynamic Programming*, Princeton University Press, Princeton, N.J.
- Campos, J. and F.L. Lewis. (1999) Adaptive Critic Neural Network for Feedforward Compensation. Proc. of American Control Conference.
- Chen, K. H. & Tsai, W. H. (2000). Vision-based obstacle detection and avoidance for autonomous land vehicle navigation in outdoor roads. *Automation in Construction*, Vol. 10, Issue 1, (Nov. 2000) 1-25.
- Choi, S. K. (1993). Design of advanced underwater robotic vehicle and graphic workstation. *IEEE Int. Conf. on Robotics and Automation*, pp. 99-105, Atlanta, GA, 1993.
- Choi, S. K.; Yuh, J. & Takashige, G. Y. (1995). Development of the omnidirectional intelligent navigator. *IEEE Robotics & Automation Magazine*, Vol. 2, No. 1, (1995) 44-51.
- Cicirelli, G.; Distanto, D.; Orazio, T. D. & Attolico, G. (1998). Learning actions from vision-based positioning in goal-directed navigation, *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vo. 3, pp. 1715-1720, 1998.
- Corradini, M. L.; Leo, T. & Orlando, G. (2002). Experimental testing of a discrete-time sliding mode controller for trajectory tracking of a wheeled mobile robot in the presence of skidding effects. *Journal of Robotic Systems*, Vol. 19, Issue 4, (April 2002) 177-188.
- Corradini, M. L. & Orlando, G. (2001). Robust tracking control of mobile robots in the presence of uncertainties in the dynamical model. *Journal of Robotic Systems*, Vol. 18, Issue 6, (June 2001) 317-323.
- Daxwanger, W. A. & Schmidt, G. (1998). Neuro-fuzzy posture estimation for visual vehicle guidance, *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, Vol. 3, pp. 2086-2091, 1998.
- Devy, M.; Chatila, R.; Fillatreau, P.; Lacroix, S. & Nashashibi, F. (). On autonomous navigation in a natural environment. *Robotics and Autonomous Systems*, Vol. 16, (Nov. 1995) 5-16.

- de Wit, C. C.; Siciliano, B. & Bastin, G. (1996). *Theory of Robot Control*. Springer-Verlag London Limited, Britain.
- Dong, W.; Liang Xu, W. & Huo, W. (1999). Trajectory tracking control of dynamic non-holonomic systems with unknown dynamics. *International Journal of Robust and Nonlinear Control*, Vol. 9, Issue 13, (November 1999) 905-922.
- Dracopoulos, D. C. (1998). Robot path planning for maze navigation, *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, Vol. 3, pp. 2081-2085, 1998.
- Dudek, G. & Jenkin, M. (2000). *Computational Principles of Mobile Robotics*, Cambridge University Press, ISBN 0-521-56876-5, United Kingdom.
- Feruson, J. & Pope, A. (1995). *THESEUS: Multipurpose Canadian AUV*, Sea Technology Magazine.
- Fuji, T.; Tamaki, U. & Kuroda, Y. (1993). Mission execution experiments with a newly developed AUV 'The Twin-Burger', *Proc. 8th Int. Symp. Unmanned Untethered Submersible Technology*, pp. 92-105, Durham, NC, 1993.
- Ghaffari, M., Liao, X., Hall, E. (2004) A Model for the Natural Language Perception-based Creative Control of Unmanned Ground Vehicles, *Proceedings of the SPIE 2004 Conference*.
- Golnazarian, W. & Hall, E. L. (2000). Intelligent Industrial Robots, In: *Handbook of Industrial Automation*, 499-527, Marcel Dekker, Inc., New York.
- Greenhouse, L. & Norris, J. (2002). *Control system architecture for military robotics*. The NIST Virtual Timeline, Information Services Division, National Institute of Standards and Technology. Available online: <http://museum.nist.gov/exhibits/timeline/printerFriendly.cfm?itemId=38>.
- Grudic G. Z. & Lawrence, P. D. (1998). Nonparametric learning approach to vision based mobile robot localization, *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 724-729, 1998.
- Honda, (2005). *Honda Debuts New ASIMO*, Honda Motor Co., Ltd, Available online: <http://world.honda.com/news/2005/c051213.html>.
- Janet, J. A.; Luo, R. C. & Kay, M. G. (1997). Autonomous mobile robot global motion planning and geometric beacon collection using traversability vectors. *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 1, (Feb. 1997) 132-140.
- Jarvis, R. (1996). An all-terrain intelligent autonomous vehicle with sensor-fusion-based navigation capabilities. *Control Eng. Practice*, Vol. 4, No. 4, (April 1996) 481-486.
- Jiang, Z.-P.; Lefeber, E. & Nijmeijer, H. (2001). Saturated stabilization and tracking of a nonholonomic mobile robot. *Systems and Control Letters*, Vol. 42, Issue 5, (April 2001) 327-332.
- Kim, K. H. C. & Hyung, S. (1998). Mobile robot navigation based on optimal via-point selection method, *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 1242-1247, 1998.
- Koh, K. C. & Cho, H. S. (1999). A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints. *Journal of Intelligent and Robotic Systems*, Vol. 24, Issue 4, (April 1999) 367-385.
- Kortenkamp, D.; Bonasso, R. P. & Murphy, R. (1998). *Artificial Intelligence and Mobile Robots*, AAAI Press/The MIT Press, ISBN 0-262-61137-6, USA.
- Krishna K. M. & Kalra, P. M. (2001). Perception and remembrance of the environment during real-time navigation of a mobile robot. *Robotics and Autonomous Systems*, Vol. 37, (Oct. 2001.) 25-51.

- Kurazume, R. & Hirose, S. (2000). Development of a cleaning robot system with cooperative positioning system. *Autonomous Robots*, Vol. 9, Issue 3, (Dec. 2000) 237-246.
- Kurd, S. & Oguchi, K. (1997). Neural network control for automatic guided vehicles using discrete reference markers, *Industry Applications Society, Thirty-Second IAS Annual Meeting*, Vol. 2, pp. 886-891, 1997.
- Lewis, F. L.; Jagannathan, S. & Yesildirek, A. (1999). *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor and Francis Ltd, T. J. International Ltd, Padstow, UK.
- Lewis, M. A.; Fagg, A. H. & Bekey, G. A. (1993). The USC autonomous flying vehicle: An experiment in real-time behavior-based control. *Proc. IEEE/RSJ IROS*, Yokohama, Japan, 1993.
- Lin, C. H. & Wang, L. L. (1997). Intelligent collision avoidance by fuzzy logic control. *Robotics and Autonomous Systems*, Vol. 20, (April 1997) 61-83.
- Liao, X. and Hall, E. (2002) Beyond Adaptive Critic - Creative Learning for Intelligent Autonomous Mobile Robots, *Proceedings of Intelligent Engineering Systems Through Artificial Neural Networks, ANNIE, in Cooperation with the IEEE Neural Network Council*, St. Louis - Missouri.
- Liao, X., et al. (2003) Creative Control for Intelligent Autonomous Mobile Robots, *Proceedings of Intelligent Engineering Systems Through Artificial Neural Networks, ANNIE, in Cooperation with the IEEE Neural Network Council*, St. Louis - Missouri.
- Marco, D. B.; Healey, A. J. & McGhee, R. B. (1996). Autonomous underwater vehicles: Hybrid control of mission and motion. *Autonomous Robots*, Vol. 3, Issue 2, (May 1996) 169-186.
- Martinez, E. & Torras, C. (2001). Qualitative vision for the guidance of legged robots in unstructured environments. *Pattern Recognition*, Vol. 34, Issue 8, (August 2001) 1585-1599.
- Meystel, A. M. & Albus, J. S. (2002). *Intelligent Systems: Architecture, Design, and Control*, John Wiley & Sons, Inc., New York.
- Montaner, M. B. & Ramirez-Serrano, A. (1998). Fuzzy knowledge-based controller design for autonomous robot navigation. *Expert Systems with Applications*, Vol. 14, Issue 1-2, (January 1998) 179-186.
- Mora, T. E. & Sanchez, E. N. (1998). Fuzzy logic-based real-time navigation controller for a mobile robot, *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 612-617, 1998.
- Premvuti, S. & Wang, J. (1996). Relative position localizing system for multiple autonomous mobile robots in distributed robotic system: System design and simulation. *Robotics and Autonomous Systems*, Vol. 18, Issue 3, (August 1996) 319-326.
- Rajagopalan, R. & Barakat, N. (1997). Velocity control of wheeled mobile robots using computed torque control and its performance for a differentially driven robot. *Journal of Robotic Systems*, Vol. 14, Issue 4, (April 1997) 325-340.
- Shim, H.-S. & Sung, Y.-G. (2003). Asymptotic control for wheeled mobile robots with driftless constraints. *Robotics and Autonomous Systems*, Vol. 43, Issue 1, (April 2003) 29-37.
- Space and Naval Warfare Systems Center, San Diego (SSC San Diego). (2002). *Robotics at Space and Naval Warfare Systems Center*. The Space and Naval Warfare Systems Center, San Diego. Available online: <http://www.spawar.navy.mil/robots>.
- Steer, B.; Kloske, J.; Garner, P.; LeBlanc, L. & Schock, S. (1993). Towards sonar based perception and modelling for unmanned untethered underwater vehicles, *Proc. IEEE Int. Conf. Robotics and Automation*, Vol. 2, pp. 112-116, Atlanta, GA, 1993.
- Stubberud, A.R. and Stubberud, S.C., (2000) *Stability*, in *Handbook of Industrial Automation*, R.L. Shell and E.L. Hall, Editors. 2000, MARCEL DEKKER, INC.: New York.

- Toda, M.; Kitani, O.; Okamoto, T. & Torii, T. (1999). Navigation method for a mobile robot via sonar-based crop row mapping and fuzzy logic control. *Journal of Agricultural Engineering Research*, Vol. 72, Issue 4, (April 1999) 299-309.
- Topalov, A. V.; Kim, J.-H. & Proychev, T. P. (1998). Fuzzy-net control of non-holonomic mobile robot using evolutionary feedback-error-learning. *Robotics and Autonomous Systems*, Vol. 23, Issue 3, (April 1998) 187-200.
- Torras, C. (1995). Robot adaptivity. *Robotics and Autonomous systems*, Vol. 15, Issue 1-2, (July 1995) 11-23.
- Vázquez, F. & García, E. (1994). A local guidance method for low-cost mobile robots using fuzzy logic. *Annual Review in Automatic Programming*, Vol. 19, (1994) 203-207.
- Vercelli, G. & Morasso, P. (1998). Recognition and classification of path features with self-organizing maps during reactive navigation, *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 1437-1442, 1998.
- Watanabe, K. L.; Izumi, K.; Tang, J. & Han, F. (1998). Rotational control of an omnidirectional mobile robot using a fuzzy servo controller. *Advanced Robotics*, Vol. 12, Issue 3, (1998) 171-189.
- Werbos, P. J. (1999). New designs for universal stability in classical adaptive control and reinforcement learning, *Proceedings of the International Joint Conference on Neural Networks IJCNN*, pp. 2292-2295, Vol. 4, USA, 1999.
- Wijesoma, W.; Khaw, P. P. & Teoh, E. K. (1999). Control and navigation of an outdoor AGV using fuzzy reasoning, *Proceedings of IEEE/IEE/SAI International Conference on Intelligent Transportation Systems*, pp. 544-549, 1999.
- Wong, W. (2005). SCI-FI robots edge closer to reality. *Electronic Design*, Vol. 53, Issue 14, (June 2005) 65-68, ISSN 0013-4872.
- Wu, S.-F.; Mei, J.-S. & Niu, P.-Y. (2001). Path guidance and control of a guided wheeled mobile robot. *Control Engineering Practice*, Vol. 9, Issue 1, (January 2001) 97-105.
- Xue, H. & Cheung, E. (1996). Learning control for position tracking of active suspension of high-speed AGV via neural network, *Proceedings of 1996 IEEE Conference on Emerging Technologies and Factory Automation*, Vol. 2, pp. 523-527, 1996.
- Yahja, A.; Sanjiv, S. & Stentz, A. (2000). An efficient on-line path planner for outdoor mobile robots. *Robotics and Autonomous systems*, Vol. 32, Issue 2-3, (Aug. 2000) 129-143.
- Yan, X.; Iyengar, S. S. & Brener, N. E. (1997). An event driven integration reasoning scheme for handling dynamic threats in an unstructured environment. *Artificial Intelligence*, Vol. 95, Issue 1, (August 1997) 169-186.
- Yu, H.; Chi, C.; Su, T. & Bi, Q. (2001). Hybrid evolutionary motion planning using follow boundary repair for mobile robots. *Journal of Systems Architecture*, Vol. 47, Issue 7, (July 2001) 635-647.
- Yun-Su, H. & Yuta, S. (1996). Trajectory tracking control for navigation of the inverse pendulum type self-contained mobile robot. *Robotics and Autonomous Systems*, Vol. 17, Issue 1-2, (April 1996) 65-80.
- Zhang, M. & Hirschorn, R. M. (1997). Discontinuous feedback stabilization of nonholonomic wheeled mobile robots. *Dynamics and Control*, Vol. 7, Issue 2, (April 1997) 155-169.
- Zhang, Y.; Chung, J. H. & Velinsky, S. A. (2003). Variable structure control of a differentially steered wheeled mobile robot. *Journal of Intelligent and Robotic Systems*, Vol. 36, Issue 3, 301-314.
- Zhu, Z. Y.; Yang, S.; Shi, D. & Xu, G. (1998). Better road following by integrating omni-view images and neural nets, *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, Vol. 2, pp. 974-979.

Decentralized Robust Tracking Control for Uncertain Robots

Zongying Shi, Yisheng Zhong and Wenli Xu
Tsinghua University
P.R. China

1. Introduction

Theoretically, the motion of rigid robots can be described by a set of nonlinear differential equations or nonlinear differential algebraic equations. In practice, however, because of the model uncertainties such as parameter uncertainty (including unknown or varying payload), unmodelled joint friction, backlash and unknown external disturbances, it is almost impossible to obtain an exact mathematical model of a practical robot system. A lot of work has been done on the motion control of robots with uncertain dynamics (Sage, 1999), and systematic analysis and design methods have been proposed, such as Lyapunov-based algorithms (Spong, 1987 & 1992; Kelly, 1994; Qu, 1994; Liu & Goldenberg, 1996 & 1997), passivity-based algorithms (Canudas et al., 1996), robust adaptive algorithms (Slotine & Li, 1987), and so on. Most of these approaches, however, use the upper bound of perturbation to synthesize control laws, so may lead to conservative results. In addition, these approaches are based on a centralized control structure which requires complicated hardware configuration and tedious computation, therefore their practical application is limited.

Decentralized control structure is adopted by the majority of modern robots in favor of its computation simplicity and low-cost hardware set-up (Liu, 1997). For robots, decentralized control means that each joint has its own independent controller which uses only local state feedback. Several decentralized robust tracking approaches have been proposed, such as PD control, PD control plus nonlinear compensation (Liu, 1997 & 1999; Kelly & Salgado, 1994; Tang & Guerrero, 1998), robust adaptive control (Fu, 1992) and Riccati equation based control (Wang & Weng, 1999). PD controller is the most classical and simplest decentralized controller which results in local stability under sufficient large proportional and differential gains, and the additional nonlinear compensation may improve the performance of tracking errors from local convergence to global convergence. Riccati equation based approaches may lead to a simple controller, but the involvement of uncertain bound in control laws may lead to conservative results. These approaches consider the torque resulting from the couplings with robot links as exogenous disturbances, therefore the tuning of the controller parameters of multiple joint robot systems is a stiff problem.

This chapter presents a design method of decentralized robust controllers for rigid robots. A feedforward control is first applied and the trajectory tracking problem is reformed into a stabilizing problem of a dynamic error system. Then the method, proposed to deal with robust control problem for single-input single-output (SISO)

plants (Zhong, 1999 & 2002), is extended to the case of the nonlinear coupled error system which is a multi-input multi-output (MIMO) system. By this method, a robust controller for each joint error subsystem is designed in two steps: first, a nominal controller is designed for the nominal plant to get desired tracking, then, a robust compensator is added to restrain the influence of perturbation that is the difference of the real plant from the nominal plant. Analytical proofs and simulation results show that robust stability and robust tracking property can be achieved in the presence of parameter uncertainty and friction. A novel feature of the presented method is that the parameters of the designed controller can be tuned on-line mono-directionally (more precisely, in the way of increasing a parameter monotonously), and both the format and the bound of uncertainties are not needed to be known when we perform the parameter tuning. This feature greatly facilitates the application of the method to real cases where it is not easy to get the information on the format or the bound of uncertainties involved.

This chapter is organized as follows. Section 1 gives the introduction. Section 2 gives the problem statements. The robust controller design method is shown in Section 3. The stability and robust tracking properties of the closed-loop system are proven in Section 4. Section 5 gives simulation results on a 2 DOF (degrees of freedom) robot. Section 6 states the conclusions.

Notations

$\lambda_{\max}(\mathbf{B})$, $\lambda_{\min}(\mathbf{B})$: the maximum and minimum eigenvalues, respectively, of a symmetric positive definite matrix \mathbf{B} .

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}, \quad \mathbf{x} \in \mathfrak{R}^n.$$

$$\|\mathbf{A}\| = \sqrt{\lambda_{\max}(\mathbf{A}^T \mathbf{A})}, \quad \mathbf{A} \in \mathfrak{R}^{n \times n}.$$

2. Problem statements

The dynamical model of a rigid robot with n DOFs can be described by its Euler-Lagrange equation

$$\mathbf{M}(\theta)\ddot{\theta} + \mathbf{C}(\theta, \dot{\theta})\dot{\theta} + \mathbf{g}(\theta) = \boldsymbol{\tau} \quad (1)$$

where $\theta \in \mathfrak{R}^n$ is the vector of angular joint displacements, $\boldsymbol{\tau} \in \mathfrak{R}^n$ is the vector of applied joint torques, $\boldsymbol{\tau}_d \in \mathfrak{R}^n$ is the vector of disturbance torques, $\mathbf{M}(\theta) \in \mathfrak{R}^{n \times n}$ is the inertia matrix, $\mathbf{C}(\theta, \dot{\theta}) \in \mathfrak{R}^n$ is the vector of Coriolis and centripetal torques, and $\mathbf{g}(\theta)$ is the vector of gravitational torques.

In order to incorporate parameter uncertainties into the model (1), the matrices $\mathbf{M}(\theta)$ and $\mathbf{C}(\theta, \dot{\theta})$, and the vector $\mathbf{g}(\theta)$ are split up into a nominal part (indicated by the subscript zero) and an uncertain part

$$(\mathbf{M}_0(\theta) + \Delta\mathbf{M})\ddot{\theta} + (\mathbf{C}_0(\theta, \dot{\theta}) + \Delta\mathbf{C})\dot{\theta} + \mathbf{g}_0(\theta) + \Delta\mathbf{g} = \boldsymbol{\tau} \quad (2)$$

For the description (1) and (2), the following properties hold which can facilitate control system design and analysis.

Property 1 (Liu, 1997) If all the joints under consideration are revolute, the inertia matrix $\mathbf{M}(\theta)$ and its inverse matrix $\mathbf{M}^{-1}(\theta)$ are positive definite, and

$$\lambda_{\min}(M)I \leq M(\theta) \leq \lambda_{\max}(M)I$$

Property 2 For any bounded $\theta_d, \dot{\theta}_d$ and bounded parameter perturbations, we have

- (1) $\|M(\theta_d) - M_0(\theta_d)\| \leq k_M$
- (2) $\|C(\theta_d, \dot{\theta}_d) - C_0(\theta_d, \dot{\theta}_d)\| \leq k_C$
- (3) $\|g(\theta_d) - g_0(\theta_d)\| \leq k_g$

where k_M, k_C and k_g are positive constants.

Property 3 (Kelly, 1997) There exist positive constants k_{M1}, k_{C1}, k_{C2} and k_{g1} such that for all $x, y, z, v, w \in \mathbb{R}^n$, we have

- (1) $\|M(x)z - M(y)z\| \leq k_{M1}\|x - y\|\|z\|$
- (2) $\|C(x, z)w - C(y, v)w\| \leq k_{C1}\|z - v\|\|w\| + k_{C2}\|z\|\|x - y\|\|w\|$
- (3) $\|C(x, y)z\| \leq k_{C1}\|y\|\|z\|$
- (4) $\|g(x) - g(y)\| \leq k_{g1}\|x - y\|$

Now, we are in position to formulate the design problem of controller. Consider a control law which consists of a feedforward controller for the nominal system along the desired joint trajectory computed off-line and a robust controller for the real system, given by

$$\tau = u + M_0(\theta_d)\ddot{\theta}_d + C_0(\theta_d, \dot{\theta}_d)\dot{\theta}_d + g_0(\theta_d) \quad (3)$$

where subscript zero indicates the corresponding nominal values, $\theta_d, \dot{\theta}_d$ and $\ddot{\theta}_d$ denote the desired joint trajectory and its first and second derivatives respectively, and u is the output of robust controller to be designed.

Substituting the control law (3) into (1) yields the following error dynamics equation

$$\ddot{e} + H(\theta)C(\theta, \dot{\theta})\dot{e} + H(\theta)\Delta = H(\theta)u \quad (4)$$

where

$$\begin{aligned} e &= \theta - \theta_d \\ H(\theta) &= M^{-1}(\theta) \\ \Delta &= [M(\theta) - M_0(\theta_d)]\ddot{\theta}_d + [C(\theta, \dot{\theta}) - C_0(\theta_d, \dot{\theta}_d)]\dot{\theta}_d + [g(\theta) - g_0(\theta_d)] \end{aligned} \quad (5)$$

In a practical robot system, each joint is generally driven by an independent actuator, so each joint is treated as a subsystem, whose error dynamics equation can be obtained from (4) and (5)

$$\ddot{\xi}_i + \sum_{j=1}^n \sum_{k=1}^n H_{ik}(\theta)C_{kj}(\theta, \dot{\theta})\dot{\xi}_j + \sum_{j=1}^n H_{ij}(\theta)\Delta_j(\theta) = \sum_{j=1}^n H_{ij}(\theta)u_j \quad (6)$$

where ξ_i is the i th element of vector \ddot{e} , \dot{e}_j , u_j and Δ_j are the j th elements of the vectors \dot{e} , u and Δ respectively. $H_{ij}(\theta)$ and $C_{ij}(\theta, \dot{\theta})$ are the (i, j) elements of the matrices $H(\theta)$ and $C(\theta, \dot{\theta})$ respectively.

The remaining part of this paper is to show the design method of a robust controller u_i for i th joint subsystem ($i = 1, \dots, n$) so that the robot tracks the desired joint trajectory θ_d with a specified performance under the condition that θ_d is twice differentiable.

3. Robust controller design

The robust controller for each joint error subsystem is designed by applying the signal-compensation-based (SCB) robust control idea (Zhong, 1999 & 2002) in two steps: Firstly, an artificial nominal model is introduced and a corresponding nominal controller is designed to achieve desired tracking properties for the nominal closed-loop system. Then, the influence of the difference between the real robot system and its nominal model is regarded as an equivalent disturbance and a robust compensator is designed to restrain the influence of uncertainties and the nonlinear couplings with robot links.

3.1 Nominal controller design

An artificial nominal plant is introduced for the i th joint error subsystem

$$\ddot{e}_i = b_0 u_i \quad (7)$$

where b_0 is a positive constant such that $b_0 \geq \lambda_{\max}(H)$. Then the nominal plant can be described in frequency domain by

$$e_i = G_i(s) u_i \quad (8)$$

where

$$G_i(s) = \frac{b_0}{D_{pi}(s)}, \quad D_{pi}(s) = s^2$$

Here we call the nominal plant (7) an *artificial* one because it might by no means be a model of the real plant for any possible practical situation. Note that the nominal plant (7) is different from that one obtained from (4) by setting $\Delta M = \theta$, $\Delta C = \theta$, and $\Delta g = \theta$ respectively. Construct the nominal controller, which is a linear time-invariant output dynamical feedback controller, as

$$u_{oi} = \frac{N_{yi}(s)}{D_{ui}(s)} e_i \quad (9)$$

where $N_{yi}(s)$ and $D_{ui}(s)$ are polynomials of degree 1 determined by the following equation

$$D_{mi}(s)L_i(s) = D_{pi}(s)D_{ui}(s) - b_0 N_{yi}(s) \quad (10)$$

where $D_{mi}(s)$ and $L_i(s)$ are Hurwitz monic polynomials of degrees 2 and 1 respectively. Then the closed-loop system consisting of the nominal plant (7) and the nominal controller (9) has a characteristic polynomial $D_{mi}(s)L_i(s)$. The existence of the polynomials $N_{yi}(s)$ and $D_{ui}(s)$ satisfying (10) is evident.

Define

$$D_{ui}(s) = s + d_{ii}$$

$$\begin{aligned} N_{y_i}(\mathbf{s}) &= r_{0i}\mathbf{s} + r_{1i} \\ D_{m_i}(\mathbf{s}) &= \mathbf{s}^2 + \beta_{1i}\mathbf{s} + \beta_{2i} \\ L_i(\mathbf{s}) &= \mathbf{s} + \alpha_{ii} \end{aligned}$$

where $D_{m_i}(\mathbf{s})$, dominating the performance of $D_{m_i}(\mathbf{s})L_i(\mathbf{s})$ under the condition $\alpha_{ii} \gg \beta_{1i}$, is a characteristic polynomial of a 2 degree system which specifies the desired dynamic tracking performance for the i th joint, and d_{ii} , r_{0i} and r_{1i} can be expressed as functions of β_{1i} , β_{2i} and α_{ii} , respectively

$$\begin{cases} d_{ii} = \alpha_{ii} + \beta_{1i} > 0 \\ r_{0i} = -(\alpha_{ii}\beta_{1i} + \beta_{2i})/b_0 < 0 \\ r_{1i} = -\alpha_{ii}\beta_{2i}/b_0 < 0 \end{cases}$$

3.2 Robust compensator design

For the i th joint error subsystem, regarding the effect of the difference between the real robot system and the nominal one as that of an external disturbance, we have

$$D_{p_i}(\mathbf{s})\mathbf{e}_i = b_0\mathbf{u}_i + \mathbf{q}_i \quad (11)$$

where \mathbf{q}_i is called the equivalent disturbance and given by

$$\mathbf{q}_i = \sum_{j=1}^n H_{ij}(\theta)\mathbf{u}_j - b_0\mathbf{u}_i - \sum_{j=1}^n \sum_{k=1}^n H_{ik}(\theta)C_{ij}(\theta, \dot{\theta})\dot{\mathbf{e}}_j - \sum_{j=1}^n H_{ij}(\theta)\Delta_j \quad (12)$$

Define the real control input to the i th joint as

$$\mathbf{u}_i = \mathbf{u}_{0i} + \mathbf{v}_i \quad (13)$$

where \mathbf{v}_i is the output of the robust compensator to be designed.

To eliminate the influence of \mathbf{q}_i on the close-loop control system, we set

$$\mathbf{v}_i = -F_i(\mathbf{s})\frac{\mathbf{q}_i}{b_0} = -F_i(\mathbf{s})\frac{1}{b_0}[D_{p_i}(\mathbf{s})\mathbf{e}_i - b_0\mathbf{u}_i] \quad (14)$$

where $F_i(\mathbf{s}) = \frac{f_i}{\mathbf{s}(\rho_i\mathbf{s}+1) + f_i}$ is called robust filter with parameters $\rho_i > 0$, $f_i > 0$.

Combining (10), (13), and (14) yields

$$\mathbf{v}_i = -\frac{F(\mathbf{s})}{1-F(\mathbf{s})} \cdot \frac{D_{m_i}(\mathbf{s})L_i(\mathbf{s})}{b_0D_{p_i}}\mathbf{e}_i$$

and consequently

$$\mathbf{u}_i = \left[\begin{array}{c} r_{0i}\mathbf{s} + r_{1i} \\ \mathbf{s} + d_{ii} \end{array} - \frac{f_i(\mathbf{s}^2 + \beta_{1i}\mathbf{s} + \beta_{2i})(\mathbf{s} + \alpha_{ii})}{b_0\mathbf{s}(\rho_i\mathbf{s}+1)(\mathbf{s} + d_{ii})} \right] \mathbf{e}_i \quad (15)$$

which is a linear time-invariant controller using only position feedback, which can avoid the noise problem caused by speed measurement. Fig. 1 shows the configuration of the whole control system with decentralized robust controllers. Each controller appears in a two-loop form: the nominal control loop and the robust control loop. Only when the perturbation from the nominal case appears, the robust control loop is turned on. This feature is very similar to that of an adaptive controller, but the complexity of this controller is almost equivalent to that of a PID controller, therefore its realization is much easier than that of an adaptive controller.

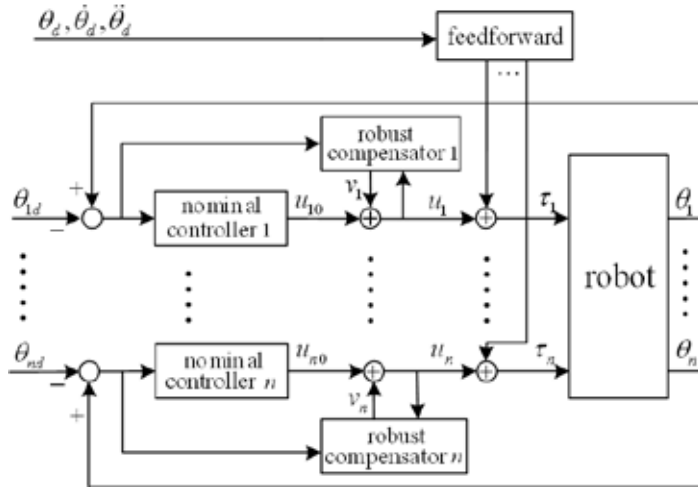


Fig. 1. Decentralized robust control system of n DOF robot.

4. Control performance analysis and parameter on-line tuning

4.1 Control performance analysis

Firstly we give the state equations of the closed-loop system, then construct a Lyapunov function to analyze the system stability and robust properties.

Let $X_i = [\hat{e}_i \ \dot{\hat{e}}_i \ u_{0i} \ v_{0i} \ \tilde{v}_i]^T$, where \hat{e}_i , $\dot{\hat{e}}_i$, u_{0i} , v_{0i} and \tilde{v}_i are the i th elements of vectors e , \dot{e} , u_0 , v_0 and \tilde{v} respectively, and $\tilde{v}_i = -v_i/\sqrt{f_i}$. Combining (9), (11) and (14) results in the state equation of the i th joint error subsystem:

$$\dot{X}_i = A_i X_i + B_{q_i} q_i \quad (16)$$

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & b_0 & 0 & -b_0 \sqrt{f_i} \\ r_{i1} & r_{0i} & -d_{ii} & 0 & 0 \\ 0 & 0 & 0 & \lambda_{1i} & 0 \\ 0 & 0 & 0 & 1/\rho_i & \lambda_{2i} \end{bmatrix}, \quad B_{q_i} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \sqrt{f_i}/b_0 \\ 0 \end{bmatrix}$$

where λ_{1i} and λ_{2i} are the poles of the robust filter or the roots of the polynomial $s(\rho_i s + 1) + f_i$, that is, $\lambda_{1,2i} = \frac{1}{2} \left(-\frac{1}{\rho_i} \pm \sqrt{\frac{1}{\rho_i^2} - \frac{4f_i}{\rho_i}} \right)$.

The characteristic polynomial of matrix A_i is

$$A_i(s) = D_m(s) L_i(s) \frac{1}{\rho_i} (s(\rho_i s + 1) + f_i)$$

which is a Hurwitz polynomial. Therefore matrix A_i is stable and there exists a positive definite matrix P_i which is the solution of the following Lyapunov equation.

$$P_i A_i + A_i^T P_i = -I \quad (17)$$

Assumption

A) $\frac{1}{\rho_i} \gg f_i \geq 1$.

B) $f_i = f_j, i \neq j, f = \text{diag}\{f_1, \dots, f_n\} = f_i I_{n \times n}$.

C) ε_A is a positive constant such that $\varepsilon_A \leq b_0^{-1} \lambda_{\min}(M^{-1})$.

Remark 1 Under Assumption A), we have that $1 \gg \rho_i f_i$. Since $M(\theta)$ (hence $M^{-1}(\theta)$) is a positive definite matrix, the constant ε_A satisfying Assumption C) always exists.

Lemma 1 Define $\psi_i = P_i B_{q_i}$ and $\frac{1}{\rho_i} = \mu f_i$, where $\mu \gg 1$, then vector ψ_i can be represented in the following form

$$\psi_i = \frac{1}{b_{0i} \sqrt{f_i}} \begin{bmatrix} \gamma_{1i} & \gamma_{2i} & \gamma_{3i} & 1 + \frac{\gamma_{4i}}{\mu} & \frac{\gamma_{5i}}{\mu} \end{bmatrix}^T$$

where $\gamma_{ji} (j=1, \dots, 5)$ are bounded by constants independent of f_i .

Proof: See the Appendix.

Lemma 2 Define the equivalent disturbance Q_i of i th joint error subsystem as (12), then the total equivalent disturbance q of system (4) can be expressed as

$$q = \tilde{q} + (M^{-1} - b_0 I) v \quad (18)$$

where $q = [q_1 \ \dots \ q_n]^T$, $\tilde{q} = -M^{-1} C \dot{e} - M^{-1} A + (M^{-1} - b_0 I) u_0$, and there exist positive constants $k_{q1}, k_{q2}, \dots, k_{q5}$ such that

$$\|\tilde{q}\| \leq k_{q1} + k_{q2} \|e\| + k_{q3} \|\dot{e}\| + k_{q4} \|\dot{e}\|^2 + k_{q5} \|u_0\| \quad (19)$$

Proof: See the Appendix.

Lemma 3 Define $A \in \mathfrak{R}^{n \times n}$ for $\bar{\sigma}(A) = \|A\|$. Under Assumption C), we have

$$\bar{\sigma}[b_0^{-1}(M^{-1} - b_0 I)] \leq 1 - \varepsilon_A$$

Proof: See the Appendix.

Lemma 4 Define μ_0 as

$$\mu_0 = \frac{k_{q1}}{b_0 \sqrt{f_i}} \sum_{i=1}^3 \gamma_{i \max} + \frac{k_{q1}}{b_0} [1 + (\gamma_{4 \max} + \gamma_{5 \max}) / \mu] \quad (20)$$

Under assumption A), if $f_i \geq f_{c1}$, where $f_{c1} = \max_{j=1}^3 (\gamma_{j \max})^2$, $\gamma_{j \max} = \max \left\{ 0, \max_{i=1}^n (\gamma_{ji}) \right\}$, $j = 1, \dots, 5$ then we have

$$\mu_0 \leq \frac{5k_{q1}}{b_0}$$

Proof: Omitted.

Theorem 1 Suppose Assumptions A), B) and C) are met. For any given constant $\eta > 0$, there exists a bounded f_i such that the closed-loop system has the following robust properties:

(1) If both the closed-loop system and the reference model are of zero initial conditions, then the closed-loop system has robust transient property:

$$\|X\|^2 < \eta, \quad \forall t \geq t_0$$

(2) If the initial conditions involved are non-zero but bounded, then the closed-loop system has robust asymptotical tracking property; that is, there exists a constant $T \geq t_0$ such that

$$\|X\|^2 < \eta, \quad \forall t \geq T$$

Proof: Consider the following Lyapunov function candidate

$$V = X^T P X = \sum_{i=1}^n V_i, \quad V_i = X_i^T P_i X_i \quad (21)$$

where

$$X = [X_1^T, X_2^T, \dots, X_n^T]^T, \quad P = \begin{bmatrix} P_1 & & \\ & \ddots & \\ & & P_n \end{bmatrix}$$

The derivative of V , along the solution of the controlled system (16), is given by

$$\begin{aligned} \dot{V} &= \sum_{i=1}^n \dot{V}_i = \sum_{i=1}^n (-X_i^T X_i + 2X_i^T P_i B_q q) \\ &\leq -(e^T e + \dot{e}^T \dot{e} + u_0^T u_0 + v_0^T v_0 + \tilde{v}^T \tilde{v}) + 2 \left\{ \sum_{i=1}^n \frac{1}{f_i b_0} \gamma_{i1} \epsilon q + \sum_{i=1}^n \frac{1}{f_i b_0} \gamma_{i2} \epsilon q \right. \\ &\quad \left. + \sum_{i=1}^n \frac{1}{f_i b_0} \gamma_{i3} u_{0i} q + \sum_{i=1}^n \frac{1}{\sqrt{f_i b_0}} (1 + \frac{\gamma_{4i}}{\mu}) v_{0i} q + \sum_{i=1}^n \frac{\gamma_{5i}}{\sqrt{f_i b_0} \mu} \tilde{v}_i q \right\} \end{aligned}$$

$$\begin{aligned}
&= -(e^T e + \dot{e}^T \dot{e} + \mathbf{u}_0^T \mathbf{u}_0 + \mathbf{v}_0^T \mathbf{v}_0 + \tilde{\mathbf{v}}^T \tilde{\mathbf{v}}) + 2 \left[e^T \text{diag} \left\{ \frac{\gamma_{i1}}{f_i b_0} \right\} + \dot{e}^T \text{diag} \left\{ \frac{\gamma_{2i}}{f_i b_0} \right\} \right. \\
&\quad \left. + \mathbf{u}_0^T \text{diag} \left\{ \frac{\gamma_{3i}}{f_i b_0} \right\} + \mathbf{v}_0^T \text{diag} \left\{ \frac{1}{\sqrt{f_i b_0}} \left(1 + \frac{\gamma_{4i}}{\mu} \right) \right\} + \tilde{\mathbf{v}}^T \text{diag} \left\{ \frac{\gamma_{5i}}{\sqrt{f_i b_0 \mu}} \right\} \right] \mathbf{q}
\end{aligned} \quad (22)$$

From Lemma 2 and the definition of $\gamma_{j \max}$ in Lemma 4, (22) becomes

$$\begin{aligned}
\dot{V} &\leq -(\|e\|^2 + \|\dot{e}\|^2 + \|\mathbf{u}_0\|^2 + \|\mathbf{v}_0\|^2 + \|\tilde{\mathbf{v}}\|^2) \\
&\quad + 2 \left\{ \left[\frac{1}{f_i b_0} (\gamma_{1 \max} \|e\| + \gamma_{2 \max} \|\dot{e}\| + \gamma_{3 \max} \|\mathbf{u}_0\|) + \frac{1}{\sqrt{f_i b_0}} \left(1 + \frac{\gamma_{4 \max}}{\mu} \right) \|\mathbf{v}_0\| + \frac{\gamma_{5 \max}}{b_0 \mu} \|\tilde{\mathbf{v}}\| \right] \cdot \|\mathbf{q}\| \right. \\
&\quad \left. + \left[\frac{1}{\sqrt{f_i}} (\gamma_{1 \max} \|e\| + \gamma_{2 \max} \|\dot{e}\| + \gamma_{3 \max} \|\mathbf{u}_0\|) \right. \right. \\
&\quad \left. \left. + \left(1 + \frac{\gamma_{4 \max}}{\mu} \right) \|\mathbf{v}_0\| + \frac{\gamma_{5 \max}}{\mu} \|\tilde{\mathbf{v}}\| \right] \cdot \|\mathbf{b}_0^{-1} (\mathbf{M}^{-1} - \mathbf{b}_0 \mathbf{I})\| \cdot \|\tilde{\mathbf{v}}\| \right\}
\end{aligned} \quad (23)$$

In the view of Lemma 1 and Lemma 3, (23) becomes

$$\begin{aligned}
\dot{V} &\leq -(\|e\|^2 + \|\dot{e}\|^2 + \|\mathbf{u}_0\|^2 + \|\mathbf{v}_0\|^2 + \|\tilde{\mathbf{v}}\|^2) \\
&\quad + 2 \left\{ \left[\frac{1}{f_i b_0} (\gamma_{1 \max} \|e\| + \gamma_{2 \max} \|\dot{e}\| + \gamma_{3 \max} \|\mathbf{u}_0\|) + \frac{1}{\sqrt{f_i b_0}} \left(1 + \frac{\gamma_{4 \max}}{\mu} \right) \|\mathbf{v}_0\| \right. \right. \\
&\quad \left. \left. + \frac{\gamma_{5 \max}}{b_0 \mu} \|\tilde{\mathbf{v}}\| \right] \cdot [\mathbf{k}_{q1} + \mathbf{k}_{q2} \|e\| + \mathbf{k}_{q3} \|\dot{e}\| + \mathbf{k}_{q4} \|\dot{e}\|^2 + \mathbf{k}_{q5} \|\mathbf{u}_0\|] \right. \\
&\quad \left. + \frac{1}{\sqrt{f_i}} (\gamma_{1 \max} \|e\| + \gamma_{2 \max} \|\dot{e}\| + \gamma_{3 \max} \|\mathbf{u}_0\|) \|\tilde{\mathbf{v}}\| \right. \\
&\quad \left. + \bar{\delta} (\mathbf{b}_0^{-1} (\mathbf{M}^{-1} - \mathbf{b}_0 \mathbf{I})) \left[\left(1 + \frac{\gamma_{4 \max}}{\mu} \right) \|\mathbf{v}_0\| + \frac{\gamma_{5 \max}}{\mu} \|\tilde{\mathbf{v}}\| \right] \cdot \|\tilde{\mathbf{v}}\| \right\}
\end{aligned} \quad (24)$$

Define $\mathbf{k}_{\text{sum}} = \mathbf{k}_{q1} + \mathbf{k}_{q2} + \mathbf{k}_{q3} + \mathbf{k}_{q5}$ and $\mathbf{k} = [\mathbf{k}_{q2}, \mathbf{k}_{q3}, \mathbf{k}_{q5}]^T$, and let k_i denote the i th element of the vector \mathbf{k} . Then (24) is further expressed as

$$\dot{V} \leq -\mu_1 \|e\|^2 - \mu_2 \|\dot{e}\|^2 - \mu_3 \|\mathbf{u}_0\|^2 - \mu_4 \|\mathbf{v}_0\|^2 - \mu_5 \|\tilde{\mathbf{v}}\|^2 + \frac{\mu_0}{\sqrt{f_i}} \quad (25)$$

where μ_0 is defined as (20),

$$\mu_i = \begin{cases} 1 - \rho_i / \sqrt{f_i} & (i = 1, 3) \\ 1 - (\bar{\rho}_i + \rho_i) / \sqrt{f_i} & (i = 2) \\ 1 - \bar{\delta} (\mathbf{b}_0^{-1} (\mathbf{M}^{-1} - \mathbf{b}_0 \mathbf{I})) (1 + \gamma_{4\max} / \mu) - \rho_i / \sqrt{f_i} & (i = 4) \\ 1 - \bar{\delta} (\mathbf{b}_0^{-1} (\mathbf{M}^{-1} - \mathbf{b}_0 \mathbf{I})) [1 + (\gamma_{4\max} + 2\gamma_{5\max}) / \mu] - \rho_i / \sqrt{f_i} & (i = 5) \end{cases} \quad (26)$$

$$\rho_i = \begin{cases} \frac{1}{\sqrt{f_i} \mathbf{b}_0} [\gamma_{1\max} k_{\text{sum}} + k_i \sum_{j=1}^3 \gamma_{j\max}] + \frac{k_i}{\mathbf{b}_0} [1 + \frac{\gamma_{4\max} + \gamma_{5\max}}{\mu}] + \gamma_{1\max} & (i = 1, 2, 3) \\ \frac{1}{\mathbf{b}_0} [1 + \frac{\gamma_{4\max}}{\mu}] k_{\text{sum}} & (i = 4) \\ \sum_{j=1}^3 \gamma_{j\max} + \frac{\gamma_{5\max} k_{\text{sum}}}{\mathbf{b}_0 \mu} & (i = 5) \end{cases} \quad (27)$$

$$\tilde{\rho}_2 = \frac{2}{\lambda_b} \left[\frac{\gamma_{1\max}}{\sqrt{f_i}} \|e\| + \frac{\gamma_{2\max}}{\sqrt{f_i}} \|\dot{e}\| + \frac{\gamma_{3\max}}{\sqrt{f_i}} \|\mathbf{u}_0\| + (1 + \frac{\gamma_{4\max}}{\mu}) \|\mathbf{v}_0\| + \frac{\gamma_{5\max}}{\mu} \|\tilde{\mathbf{v}}\| \right] k_{q4} \quad (28)$$

Define $\mathbf{Y} = [\mathbf{e}^T, \dot{\mathbf{e}}^T, \mathbf{u}_0^T, \mathbf{v}_0^T, \tilde{\mathbf{v}}^T]^T = [\mathbf{Y}_1^T, \mathbf{Y}_2^T, \mathbf{Y}_3^T, \mathbf{Y}_4^T, \mathbf{Y}_5^T]^T$, then under Assumption A), we have

$$\tilde{\rho}_2 \leq \phi_1 \sum_{i=1}^5 \|\mathbf{Y}_i\| \leq \sqrt{5} \phi_1 \|\mathbf{Y}\| \quad (29)$$

where $\phi_1 = 2k_{q4} \mathbf{b}_0^{-1} \max\{\gamma_{1\max}, \gamma_{2\max}, \gamma_{3\max}, 2\}$, is a bounded constant independent of f_i .

In the following, the conditions are analyzed that ensure $\mu_i > 0$, $i = 1, 2, \dots, 5$.

Let π_{\min} be a positive constant so that $\pi_{\min} < \min\{\varepsilon_{\Delta}, \mathfrak{F}\}$, and let $\bar{\rho}_i$ denote the values of ρ_i when $f_i = 1$ and $\mu = \gamma_{4\max} + 2\gamma_{5\max}$. Then under Assumption A) and the condition that $\mu \geq \gamma_{4\max} + 2\gamma_{5\max}$, we have $\rho_i \leq \bar{\rho}_i$.

If the following inequalities hold

$$f_i \geq f_{c2}, \quad \mu \geq \gamma_{4\max} + 2\gamma_{5\max} \quad (30)$$

$$\|\mathbf{Y}\| \leq \frac{1}{\sqrt{5}\phi_1} [(1 - \pi_{\min}) \sqrt{f_i} - \bar{\rho}_2] \quad (31)$$

where

$$f_{c2} = \max \left\{ \max_{i=1,3} \left(\frac{\bar{\rho}_i}{1 - \pi_{\min}} \right)^2, \max_{i=4,5} \left(\frac{\bar{\rho}_i}{\varepsilon_{\Delta} - \pi_{\min}} \right)^2 \right\} \quad (32)$$

then by (26) and from Lemma 3, we have

$$\mu_i \geq \pi_{\min}, \quad i = 1, 2, \dots, 5 \quad (33)$$

Note that (31) defines the attractive region.

If (33) holds and $\mathbf{f}_i \geq \mathbf{f}_{c1}$, from Lemma 4, one sees that (25) can be rewritten as

$$\dot{V} \leq -\pi_{\min} \|\mathbf{Y}\|^2 + \frac{5k_{q1}}{\sqrt{\mathbf{f}_i \mathbf{b}_0}} \quad (34)$$

Since

$$\|\mathbf{Y}\|^2 = \|\mathbf{X}\|^2 \geq \frac{\mathbf{X}^T \mathbf{P} \mathbf{X}}{\lambda_{\max}(\mathbf{P})} = \frac{V}{\lambda_{\max}(\mathbf{P})}$$

we obtain

$$\dot{V} \leq -\zeta_s V + \varepsilon_s \quad (35)$$

and

$$V(t) \leq V(t_0) e^{-\zeta_s(t-t_0)} + \frac{\varepsilon_s}{\zeta_s} (1 - e^{-\zeta_s(t-t_0)}) \quad (36)$$

where $\zeta_s = \frac{\pi_{\min}}{\lambda_{\max}(\mathbf{P})}$, $\varepsilon_s = \frac{5k_{q1}}{\sqrt{\mathbf{f}_i \mathbf{b}_0}}$.

Let

$$\bar{V} = \max\{V(t_0), 5k_{q1} \mathbf{b}_0^{-1} \zeta_s^{-1}\}$$

Then under Assumption A) and from (35) it is known that $V(t) \leq \bar{V}, \forall t \geq t_0$ and $\lim_{t \rightarrow \infty} V(t) \leq \varepsilon_s \zeta_s^{-1}$. Because

$$\|\mathbf{Y}\|^2 = \|\mathbf{X}\|^2 \leq \frac{\mathbf{X}^T \mathbf{P} \mathbf{X}}{\lambda_{\min}(\mathbf{P})} = \frac{V}{\lambda_{\min}(\mathbf{P})} \quad (37)$$

$\mathbf{Y}(t)$ (and $\mathbf{X}(t)$) starting from the attractive region given by (31) can keep staying inside that region, if

$$\frac{1}{\sqrt{5}\phi_1} [(1 - \pi_{\min})\sqrt{\mathbf{f}_i} - \bar{\rho}_2] \geq \sqrt{\bar{V} \lambda_{\min}^{-1}(\mathbf{P})} \quad (38)$$

By Assumption A), (38) holds, if

$$\mathbf{f}_i > \mathbf{f}_{c3}, \quad \mathbf{f}_{c3} = \frac{1}{(1 - \pi_{\min})^2} \left(\phi_1 \sqrt{5\bar{V} \lambda_{\min}^{-1}(\mathbf{P})} + \bar{\rho}_2 \right)^2 \quad (39)$$

In addition, from (35) and (37) we know that the state \mathbf{Y} converges to the set given by

$$\Omega_s = \left\{ \mathbf{Y} \mid \|\mathbf{Y}\|^2 \leq \varepsilon_s / (\zeta_s \lambda_{\min}(\mathbf{P})) \right\}$$

at a rate not slower than $\exp(-\zeta_s t/2)$.

By (37), if

$$V(t) \leq \lambda_{\min}(\mathbf{P})\eta$$

then

$$\|\mathbf{X}\|^2 \leq \eta$$

When both of the closed-loop system and the reference model are of zero initial conditions, $V(t_0) = 0$. In this case, from (37), we have $\|\mathbf{X}\|^2 < \eta$, if

$$\frac{\varepsilon_s}{\zeta_s} \leq \eta \lambda_{\min}(\mathbf{P}) \quad (40)$$

In the case of non-zero initial states, $V(t_0) \neq 0$. We can obtain $\|\mathbf{X}\|^2 < \eta$, if

$$V(t_0) e^{-\zeta_s(t-t_0)} + \frac{\varepsilon_s}{\zeta_s} \leq \eta \lambda_{\min}(\mathbf{P}) \quad (41)$$

If $T \geq t_0 + \frac{1}{\zeta_s} \ln\left(\frac{2V(t_0)}{\eta \lambda_{\min}(\mathbf{P})}\right)$, we have

$$V(t_0) e^{-\zeta_s(T-t_0)} \leq \frac{1}{2} \eta \lambda_{\min}(\mathbf{P})$$

Now to ensure (41) hold, it is required that

$$\frac{\varepsilon_s}{\zeta_s} \leq \frac{1}{2} \eta \lambda_{\min}(\mathbf{P}) \quad (42)$$

Incorporating (40) and (42) yields

$$f_i \geq f_{c4}, \quad f_{c4} = \left(\frac{10K_{q1} \lambda_{\max}(\mathbf{P})}{\eta D_0 \pi_{\min} \lambda_{\min}(\mathbf{P})} \right)^2 \quad (43)$$

From the previous analysis, we see that it is required

$$f_i \geq \max_{j=1}^4 f_{qj}, \quad \mu \geq \gamma_{4\max} + 2\gamma_{5\max}$$

where $f_{q_j}, j=1, \dots, 4$ are bounded constants and $\gamma_{4\max} + 2\gamma_{5\max}$ are bounded by a constant, all independent of f_i . When the parameters f_i and μ are set such that the above conditions hold, then the robust transient property and robust tracking property can be achieved. So the conclusions of the theorem are proven.

Remark 2 If the friction torque is bounded by $\beta_1 + \beta_2 \|\dot{\theta}\| + \beta_3 \|\dot{\theta}\|^2$, where β_1, β_2 , and β_3 are positive constants, then the upper bound of \bar{q} also can be expressed as the form of (19), therefore the conclusions of Theorem 1 are still available.

4.2 Parameter on-line tuning

In practical applications, needing not the tedious analysis and estimation of the uncertainty as in the previous proof, the parameters of robust compensator can be determined by on-line tuning as follows: 1) let $\mu \gg 1$, and $p_i = 1/(\mu f_i)$ (From Lemma 4, Theorem 1, and their proofs, one sees that it is enough to set $\mu \geq \gamma_{4\max} + 2\gamma_{5\max}$); 2) construct the controller as (15) and close the control loop; 3) tune f_i monotonously from some value greater than or equal to 1 until the satisfied tracking performance is achieved.

Because there is only *one* parameter to be tuned and the tuning is monotonously increasing, the parameter on-line tuning can be performed even easier than that for a traditional PID controller. In addition, the controller tuning of each joint subsystem is independent of each other.

In real cases, however, we should not apply a too large f_i to avoid too large control input caused by nonzero initial conditions or measure noise.

For a real system, it is difficult to exactly determine the range of uncertain parameters, so the feature of parameter on-line tuning of this controller possesses its unique superiority.

5. Numerical example and simulation results

We apply the method proposed in previous sections to a 2DOFs manipulator (Liu, 1999) to verify the monotonously tuning characteristic and effectiveness of the controller. The manipulator's physical parameters and their real values are shown in Table 1. The entries of $M(\theta)$, $C(\theta, \dot{\theta})$ and $g(\theta)$ are as follows respectively.

$$M_{11} = J_{11} + 2J_{22} \cos\theta_2, \quad M_{12} = M_{21} = J_{21} + J_{22} \cos\theta_2, \quad M_{22} = J_{21}$$

Let $h(\theta, \dot{\theta}) = C(\theta, \dot{\theta})\dot{\theta}$, then we have

$$\begin{aligned} h_1 &= -J_{22}(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2)\sin\theta_2, \quad h_2 = J_{22}\dot{\theta}_1^2\sin\theta_2 \\ g_1 &= J_{12}\cos\theta_1 + m_2g_2\cos(\theta_1 + \theta_2), \quad g_2 = m_2g_2\cos(\theta_1 + \theta_2) \end{aligned}$$

where J_{ij} are constant parameters given by

$$\begin{aligned} J_{11} &= m_1l_{c1}^2 + m_2(l_1^2 + l_{c2}^2) + I_1 + I_2 \\ J_{12} &= g(m_1l_{c1} + m_2l_1) \end{aligned}$$

$$J_{21} = m_2 l_{c2}^2 + I_2, \quad J_{22} = m_2 l_{c2}$$

The nominal values of robot's physical parameters are $\hat{m}_1 = 3.5\text{kg}$, $\hat{m}_2 = 1.7\text{kg}$, $\hat{I}_1 = 0.9\text{kgm}^2$, $\hat{I}_2 = 0.7\text{kgm}^2$, $\hat{l}_i = l_i$, $\hat{l}_{ci} = l_{ci}$, $i = 1, 2$. Let the initial tracking error $\mathbf{e} = [0.2 \quad -0.2]^T$ (rad) and $\dot{\mathbf{e}} = [-0.25 \quad 0.2]^T$ (rad/s).

The reference trajectories are also specified as in (Liu, 1999) for comparison:

$$\theta_{1d} = 0.2 + 2\sin 2t \quad (\text{rad})$$

$$\theta_{2d} = -1.7 + 1.8\cos 2t \quad (\text{rad})$$

The 2 degree reference model is set as follows: the rise-time $t_r = 0.5\text{s}$, the damp coefficient $\xi = 1.1$. Thus $D_m(s) = s^2 + 10.1s + 21.1$. Set $\mu = 10$ ($> \gamma_{4\max} + 2\gamma_{5\max} = 5$), and the

Items	symbol	Link1	Link2
Link length(m)	l_i	0.5	0.25
Distance from joint axis to COM of link(m)	l_{ci}	0.25	0.15
Link mass(kg)	m_i	4.0	2.0
Moment of inertia of link(kgm ²)	I_i	1.0	0.8

Table 1. Physical parameters of a 2DOFs manipulator.

value of \hat{f}_i is tuned from 1.0. The tracking errors of uncertain robot with parameter uncertainty corresponding to $f_i = 10$, $f_i = 20$ and $f_i = 50$ are shown in Fig. 2 and Fig. 3. The results indicate that, along with the gradual increment of \hat{f}_i , tracking errors gradually decrease and tracking performance is improved, which validates the monotonously tuning characteristic of the controller.

Fig. 4 ~ Fig. 7 show the tracking errors of three control schemes with almost equivalent proportional gain: signal-compensation-based (SCB) control law proposed in this paper, PD-plus-nonlinear (PD+NL) control law (Liu, 1999) and PD-plus-nonlinear-plus-adaptive (PD+NL+AD) control law (Liu, 1999). It can be seen that the tracking performance of SCB control is superior to that of PD+NL control and not inferior to that of PD+NL+AD control. The input torque signals are shown in Fig. 8 and Fig. 9, which indicate that, after a very short initial phase, the input torque curves of SCB control is smoother than that of PD+NL control.

Remark 4 To estimate the lower bound of \hat{f}_i for stable tracking control of the 2DOF manipulator, we calculate the value of f_{c2} which equals 4624. By Theorem 1, only if $f_i > 4624$, local convergence of the tracking errors may be ensured. But the simulation results show that the tracking errors are already convergent when $f_i \ll f_{c2}$. Theorem 1 only gives conservative sufficient conditions for convergence in quantity, however, its importance is that the existence of bounded \hat{f}_i ensuring the stability and robust properties of the controlled system is proved theoretically as shown in section 4.

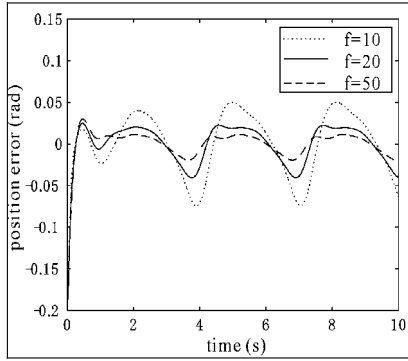


Fig. 2. Position tracking errors of joint 1 for different f_i 's values.

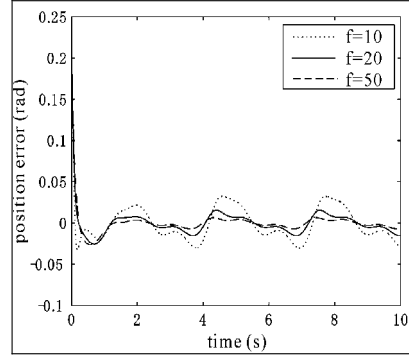


Fig. 3. Position tracking errors of joint 2 for different f_i 's values.

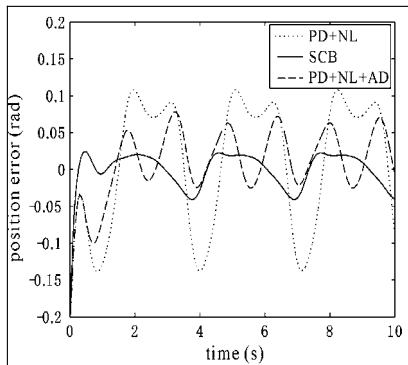


Fig. 4. Position tracking errors of joint 1.

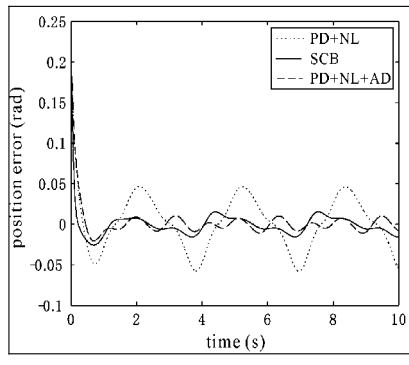


Fig. 5. Position tracking errors of joint 2.

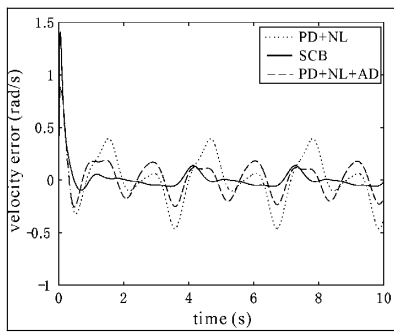


Fig. 6. Velocity tracking errors of joint 1.

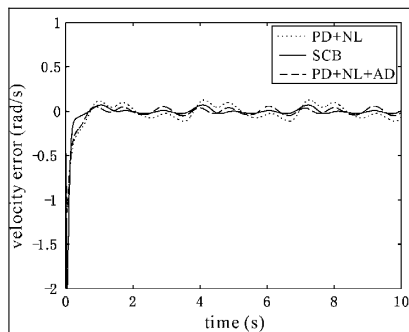


Fig. 7. Velocity tracking errors of joint 2.

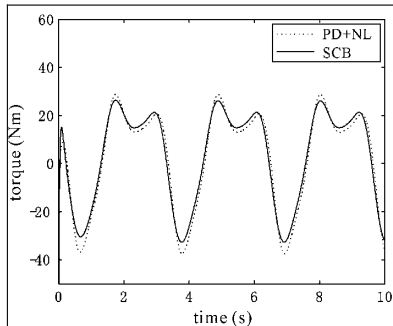


Fig. 8. Input torques of joint 1.

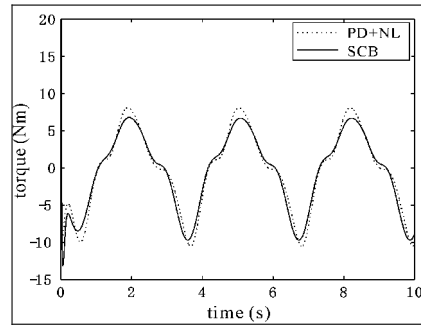


Fig. 9. Input torques of joint 2.

6. Conclusions

The controller designed by the method proposed in this chapter consists of a feedforward and a two-loop linear time-invariant robust controller. Because the robust compensation signal is generated based on the inner signals, there is no need to know the format of uncertainties. In addition, only using local joint position feedbacks, the noise problem possibly caused by velocity measurement is avoided. The most important is that the parameter tuning is monotonic which provides the unique superiority: tuning on-line is realizable, and the bound estimation of uncertainties is not needed. These characteristics enable the controller possess favorable adaptability and be prone to be realized easily.

7. References

- Canudas De Wit C.; Siciliano, B. & Bastin, G. (Ed.), (1996). *Theory of Robot Control*, 78-103, Springer, London
- Fu, L. (1992). Robust adaptive decentralized control of robot manipulators. *IEEE Trans. Automatic Control*, Vol.37, 106-110
- Kelly, R. & Salgado, R. (1994). PD control with computed feedforward of robot manipulators: a design procedure. *IEEE Trans. Robotics and Automation*, Vol.10, 566-571
- Kelly, R. (1997). PD control with desired gravity compensation of robotic manipulators: A review. *Int. J. robotics research*, Vol.16, 660-672
- Liu, G. & Goldenberg, A. (1996). Uncertainty decomposition -based robust control of robot manipulators. *IEEE Trans. Control Systems Technology*, Vol.4, 384-393
- Liu, G. & Goldenberg, A. (1997). Robust control of robot manipulators based on dynamics decomposition. *IEEE Trans. Robotics and Automation*, Vol.13, 783-789
- Liu, M. (1997). Decentralized PD and robust nonlinear control for robot manipulators. *J. Intelligent and Robotic Systems: Theory & Applications*, Vol.20, 319-332
- Liu, M. (1999). Decentralized control of robot manipulators: nonlinear and adaptive approaches. *IEEE Trans. Automatic Control*, Vol.44, 357-363
- Qu, Z.; Dawson, D. & Lim, S. (1994). A new class of robust control laws for tracking of robots. *Int. J. Robotics Research*, Vol.13, 355- 363
- Sage, H. & De Mathelin M. (1999). Robust control of robot manipulators: a survey. *Int. J. Control*, Vol. 72, 1498-1522

- Slotine, J. & Li, W. (1987). On the adaptive control of robot manipulators. *Int. J. Robotics Research*, Vol.6, 49-59
- Spong, M. & Thorp, J. (1987). Robust microprocessor control of robot manipulators. *Automatica*, Vol. 23, 373-379
- Spong, M. (1992). On the robust control of robot manipulator. *IEEE Trans. Automatic Control*, Vol. 37, 1782-1786
- Tang, Y. & Guerrero, G. (1998). Decentralized robust control of robot manipulators. ACC'98, pp.922-926
- Wang, J. & Wend, H. (1999). Robust decentralized control of robot manipulators. *Int. J. Systems Science*, Vol.30, 323-330
- Zhong, Y. ; Nagai, Y. & Takeuchi, Y. (1999). Robust output tracking of a class of nonlinear time-varying uncertain systems. *Proceedings of the 14th IFAC World Congress*, vol. G, pp.425-430, Beijing
- Zhong, Y. (2002). Robust output tracking control of SISO plants with multiple operating points and with parametric and unconstructed uncertainties. *Int. J. Control*, Vol.75, 219- 241

Appendix: Proofs of Lemma 1 ~ Lemma 3

Proof of Lemma 1:

Solving (17), we can find that $\psi_i = P_i B_{q_i}$ can be expressed in the form as stated in the lemma, and under the condition that $1/\rho_i \gg f \geq 1$, γ_{ji} can be represented as

$$\gamma_{ji} = \frac{a_{2i} f_i^2 + a_{1i} f_i + a_{0i}}{f_i^2 + (\alpha_{1i} + \beta_{1i}) f_i + (\alpha_{1i} \beta_{1i} + \beta_{2i}) + (\alpha_{1i} \beta_{2i}) f_i^{-1}}, \quad j=1,2,3$$

$$\gamma_{ji} = \frac{b_{3i} f_i^3 + b_{2i} f_i^2 + b_{1i} f_i + b_{0i}}{f_i^3 + (\alpha_{1i} + \beta_{1i}) f_i^2 + (\alpha_{1i} \beta_{1i} + \beta_{2i}) f_i + \alpha_{1i} \beta_{2i}}, \quad j=4,5$$

where $a_{ji} (j=0,1,2)$ and $b_{ki} (k=0,1,2,3)$ are constants independent of f_i . Since $\alpha_{1i}, \beta_{1i}, \beta_{2i} > 0$, obviously, the denominator of γ_{ji} , denoted by $\text{den}(\gamma_{ji})$, is larger than zero, so γ_{ji} is continuous in f_i and is bounded by constants which are independent of f_i . \square

Proof of Lemma 2:

From (5), we have

$$\begin{aligned} \|A\| \leq & \|M(\theta) - M(\theta_d)\| \|\ddot{\theta}_d\| + \|M(\theta_d)\ddot{\theta}_d - M_o(\theta_d)\ddot{\theta}_d\| + \|C(\theta, \dot{\theta}) - C(\theta_d, \dot{\theta}_d)\| \|\dot{\theta}_d\| \\ & + \|C(\theta_d, \dot{\theta}_d)\dot{\theta}_d - C_o(\theta_d, \dot{\theta}_d)\dot{\theta}_d\| + \|g(\theta) - g(\theta_d)\| + \|g(\theta_d) - g_o(\theta_d)\| \end{aligned}$$

Suppose the desired trajectory is bounded and satisfies

$$\|\theta_d\| \leq c_1, \quad \|\dot{\theta}_d\| \leq c_2, \quad \|\ddot{\theta}_d\| \leq c_3$$

where C_1 , C_2 and C_3 are constants. In the view of Property 1~3 stated in Section2, there exist positive constants $k_M, k_{M1}, k_C, k_{C1}, k_{C2}$, satisfying

$$\|A\| \leq k_M c_3 + k_{M1} c_3 \|e\| + k_C c_2 + k_{C1} c_2 \|\dot{e}\| + k_{C2} c_2^2 \|e\| + k_g + k_{g1} \|e\|$$

From Property 1 and Property 3(3), there exist positive constants k_{C3}, k_{M2} such that

$$\| -M^{-1} C \dot{e} + (M^{-1} - b_0 I) u_0 \| \leq k_{C3} \|\dot{e}\| + k_{C4} \|\dot{e}\|^2 + k_{M2} \|u_0\|$$

So we have

$$\|\hat{q}\| \leq k_{q1} + k_{q2} \|e\| + k_{q3} \|\dot{e}\| + k_{q4} \|\dot{e}\|^2 + k_{q5} \|u_0\|^2$$

where $k_{q1}, k_{q2}, \dots, k_{q5}$ are bounded positive constants. \square

Proof of Lemma 3:

Since $b_0 \geq \lambda_{\min}(M^{-1})$ which implies that $\bar{\sigma}(b_0^{-1} M^{-1}) \leq 1$, so we have

$$\bar{\delta}[b_0^{-1} M^{-1} - I] \leq 1 - b_0^{-1} \lambda_{\min}(M^{-1})$$

From Assumption C), it follows that

$$\bar{\delta}[b_0^{-1} M^{-1} - I] \leq 1 - \varepsilon_d$$

Therefore the conclusion holds. \square

Gauging Intelligence of Mobile Robots

Ka C Cheok
Oakland University,
Rochester, Michigan, USA

1. Introduction

This chapter offers approaches for gauging the intelligence of a mobile robot and case studies as illustrations. The meaning of “Intelligence” varies greatly depending on the context in which the word is used. A general discussion on various perspectives for intelligence is provided. A framework consisting of an agent, an environment and a goal is provided for viewing intelligent behaviors. Three possible means of gauging performance of intelligent mobile robots are suggested. The first is a qualitative perception that is based on an extension of the Turing test for perceiving an unsuspecting UVS behavior as that of a person. The second is a quantitative measure where task-specific intelligence performance of a smart system is evaluated. The third is the comparative scale that gauges the difficulty of challenges against the intelligent skills of humans. Two case studies, one involving a commercially available robotic floor vacuum cleaner and the other autonomous competition mobile robots, are gauged using the suggested measures. Included in this chapter is a description of a key experiment which revealed that it requires the mind of at least a four-year-old child to successfully navigate an autonomous navigation course.

2. Intelligence

Gauging intelligence is one of the most challenging and controversial topics among researchers in the fields of artificial intelligence (AI) (Nilson, 1980), intelligent systems sciences and robotics (McCorduck, 2004). We can simplify the views by realizing that human intelligence is not the only kind of intelligence in nature (McCorduck 2004, Armand, 2005). One can think of non-human intelligence as found in processes and powers of biological systems, or what if we encounter extraterrestrial intelligence. There have been dramatic advances in the use of artificial intelligence in the commercial sector in recent years. Today’s machines and computers incorporate some form of intelligence, albeit in different levels of sophistication. Realizing this, it is helpful to differentiate various kinds of intelligence: human, non-human, extraterrestrial, artificial, machine, etc.

Presence and context of Intelligence. One often recognizes intelligence, when one sees it. The meaning of intelligence depends on whom or what the word is affiliated with. There is no single one-size-fits-all definition although an acceptable informal working definition is that: *Intelligence is a measure of an agent’s ability to achieve goals in a wide range of environments* (Legg & Hutter, 2006). In practice, the definition would be better served by defining it with respect

to its associated context and scope. For example, intelligent skills expected of a human driven vehicle versus that of an autonomous vehicle.

Absence of Intelligence. One easily recognizes the absence of intelligence when something stupid or unintelligent happens. An autonomous vehicle that crashes into an unexpected obstacle in its path is a clear example of lacking a necessary intelligent reflexive response which would consist of sensory perception and reactive control action (Control Handbook, 1995). Now if the vehicle sees and avoids the obstacle but fail to complete an alternate path to its destination, assuming it is possible, then it lacks a sufficient intelligent decision and execution. Next if the vehicle does not learn from the environment and repeated experiences, then it lacks a higher intelligent cognition to learn and adapt. The reflexive response is a necessary part likened to the nervous system of a human, and the decision/execution and cognition is a sufficient part likened to the neural system. One recognizes an absence of intelligence when these necessary and sufficient factors of intelligence are not present.

Natural Intelligence (NI). Natural intelligence, including biological and psychological intelligence, is usually associated with a creature's ability to maximize the chance of success in achieving its goal in a uncertain, complex, and, often, hostile environment. High intelligence traits deal with intricate thinking processes involving perception, reasoning, emotion, and action, for formulating and re-formulating a course of undertaking. Humans exhibit a most complex form of such traits that include yearning to communicate, organize, and succeed. On the other hand, low intelligence traits of simplest life forms, such as microbes, may involve mainly survival and reproduction.

Human Intelligence. Psychologists still differ as to a precise definition of the comprehensiveness and functions of human intelligence. There are many definitions and theoretical models of human intelligence, and practical characterization of intellectual abilities can be controversial. Many associate human intelligence as a sum of specific clever abilities best displayed in specific situations in a timely manner. Human intelligence, at the minimum, has a capability to learn, understand, act, and adapt accordingly (Pinker, 1997).

Survival and Adaptation. Natural intelligence emerges from competitive struggles for survival and gene propagation over millions of years. All forms of life may increase their intelligence over successive generations via evolution, instinct, and learning. In almost all the cases, survival of the fittest in a natural selection process is often ensured for species that are the most adaptive. As humans and microbes adapt in varying ways, it is clear that adaptation is an essential intelligence trait that ensures survivability.

Relative Degree of Intelligence. While intelligence varies in complexity and ability to adapt, a measure of intelligence is often made by comparison to a norm of some sort. A means to gauge intelligence is to compare it with a known or expected level of competence. For instance, an intelligence quotient (IQ) is the ratio of individual intellectual performance over expected intellectual performance of a collection of individuals. Psychologists have developed many such scales for gauging human intelligence including children's cognitive capabilities.

Eras of Intelligence. As knowledge advances with time and so does perception of intelligence. What might have been considered intelligent may not be viewed as intelligent now. A simple example is to contrast a person from 1900 to that from 2000.

Technology and Intelligence Paradox. As technologies advances, the concept of what intelligence must continue to evolve. Memory and computation which were once the hallmark of intelligence are now trivial tasks for a computational machine. The mentally demanding tasks of challenging games have been mostly reduced to heuristics searches. A child genius

can recall volumes of information but so can a computer. There is an odd paradox that once a process or algorithm is understood, it tends not to be viewed as intelligent.

Machine and Intelligence Paradox. Many of today's automation and/or robotic systems can handle extremely well the tasks considered difficult by human standard. Strangely enough, it is not easy to build machines that can handle certain situations considered easy by human. An example is the task of driving an automobile, where clever but simple decisions are often needed. Such paradox is not apparent before one actually attempt to build intelligent machines (Albus & Meystel, 2001). The paradox is what makes research on intelligent systems exciting

Unmanned Vehicle Systems and Mobile Robots. Significant recent advances have been made in the field of autonomous/semi-autonomous (A/SA) unmanned aerial, ground, underwater, surface vehicles (UAV, UGV, UUV and USV). Because of complexities encountered in ground missions, progress in intelligent autonomous UGV systems is often considered to be most difficult. UGVs a.k.a. mobile robots presently pose challenges that the AI and intelligent systems communities can overcome and contribute significantly (Advanced robotics, 1996, Jones, et al., 1000, Martin, 2001).

Magic and Technology. In *The Wizard of Oz* (1939), Dorothy Gale is swept away to a magical land in a tornado and embarks on a quest to see the Wizard who can help her return home. The Wizard turns out to be a professor who uses technology that to perform magic. An automatic door at our supermarkets would be magical to a person from the past or rural parts of the world. Indeed, intelligent mobile robots and bipedal humanoids wandering around performing useful tasks or possibly acting as adversaries to human beings may be construed by many as indistinguishable from magic.

3. Problem & Approach

The chapter presents possible means of gauging intelligence of mobile robots. The first is a qualitative perception that is based on an extension of the Turing test to perceiving an unsuspecting UVS behavior as that of a person. The second is a quantitative measure where task-specific intelligence performance of a smart system is evaluated and scored. The third is the comparative scale that gauges the difficulty of challenges against the intelligent skills of humans.

The chapter will present a framework consisting of an agent, an environment and a goal for considering the performance of a robot. Case studies involving a commercially available Roomba vacuum cleaner and mobile robots for the annual Intelligent Ground Vehicle Competition will be presented to illustrate the proposed gauging of intelligence. The chapter will describe an experiment which showed that the mind of at least a four-year-old child was necessary to successfully complete an autonomous navigation course.

4. Framework for Agent, Environment & Goal

An agent, an environment and a goal are three essential components for modeling an intelligent system. A mobile robot whose intelligence is question would be referred to as an *agent*; and the external environment, condition, problem or situation that it interacts with as the *environment*.

Case Study A. Consider the iRobot Roomba robot (agent) whose goal is to vacuum floors (environment) that varies in sizes and shapes. It has a random cleaning pattern shown in

Figure 1. It also an ability to estimate the room size and so adjust its cleaning time, and to return to a Home Base for recharging before its battery is depleted

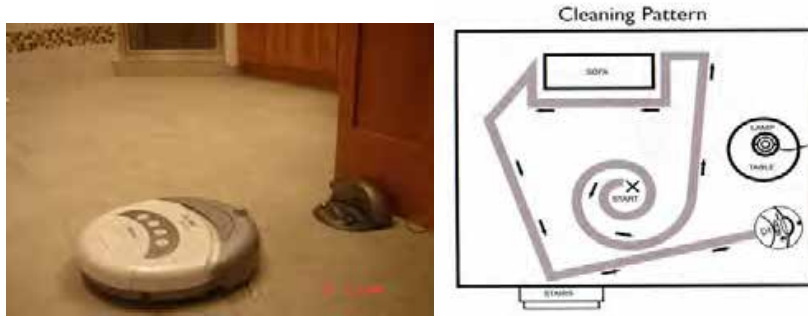


Fig. 1. The Roomba, its environment and cleaning pattern.

Case Study B. Next consider an annual Intelligent Ground Vehicle Competition vehicles and its challenges. A typical technology make-up for an autonomous IGVC vehicle to implement intelligent control schemes is shown in Figure 2. (See www.igvc.org for more details)



Fig. 2. Typical technology make-up of an IGVC vehicle.

The IGVC environments and goals consist of 1) Autonomous Challenge Event (ACE) where an agent must negotiate around an outdoor obstacle course in a prescribed time while staying within the 5 mph speed limit, and avoiding the random obstacles and escaping traps along the track (Figure 3), 2) Navigation Challenge Event (NCE) where agent autonomously travel from a starting point to a number of target destinations (waypoints or landmarks) and return to home base, given only a map showing the coordinates of those targets (Figure 4).



Fig. 3. Obstacle traps and lane for ACE.



Fig. 4. Waypoints and obstacles for NCE.

5. Gauging of Intelligence

The Turing test for AI essentially asserts: If you unknowingly communicated with a computer via text messages, and perceived that you have communicated with a person, then the computer has demonstrated a certain level of artificial intelligence. Turing also predicted that by the end of the century we would be able to speak of machine that thinks. Indeed, many of today's automated technology have surpassed the test and are capable of communicating with us better than a real person could.

A simple extension of the Turing test to IUVS can be asserted as follow:

If you perceive that a computer controlled system performs as well as a human would, then the system has demonstrated a certain level of intelligent system behavior.

In this qualitative perception, the communication could be entirely by means of observation by human observers.

Example. A car passed you by, stopped at red traffic lights, sped up on green, and drove safely through the thick of traffic, as though there was a person driving it. If that car was driven by an intelligent control system, then you have just witnessed an IUVS in action.

Second example. A jetliner with 400 passengers landed smoothly during a blizzard. As they were leaving the plane, the passengers, having endured a rough flight, thanked the captain for the safe landing. The captain smiled, pondering the fact that the aircraft had autonomously landed itself using a new automatic landing control system.

Occam's razor. One of the fine judgments in this intelligence test is to see if the agent employs a simplest and effective solution to a problem at hand that has several different possible alternatives consistent with observed environment data. For example, a simple left-right turn that allows the agent to return to the original should be preferred over a more elaborate alternate path. An agent that chooses the former would generally be considered rational and intelligent than if it chooses the latter. This common sense of embracing the less complicated option is known as Occam's razor, and is identified with intelligent behavior.

Case A. In the case of the Roomba, qualitative perception questions related to the goal would include: i) Does the agent perform (clean/vacuum) like a person would? ii) Is estimation of room size and adjustment of sweeping time noticeable? iii) Does it return to the Home Base? The answers are as follows: i) The agent does not clean like a person; its random cleaning pattern does not follow the principle of Occam's razor. ii) Sweeping time, even if adjusted accordingly, is not noticeable. iii) The agent does return to the Home Base to recharge its battery, and this is an outstanding feature of the Roomba.

Case B. In the case of the IGVC, the qualitative perception questions related to the goal would include: i) Does the agent perform (navigate) the autonomous challenge course like a person would? ii) Does the agent perform (traverse) the navigation challenge course like a person would? iii) Does the agent perform (follow) the autonomous challenge course like a person would? The answers are as follows: i) Winning vehicles/agents do navigate like a person would. ii) In this case, winning agents traverse the navigation challenge course much faster than a person would since the computers can interpret the GPS data with ease. iii) Winning agent does follow the leader but not as well or smoothly as a person would.

5.2 Quantitative Measure

Task Specific Intelligence. Because of the limitation in system capability, the scope of expected intelligent behavior for a UGV would always have to be task specific in nature. This allows one to set up a goal for the IUVS, define expected challenges, guess potential uncertainties, and measure success in task completion. The task specific intelligence problem should pose a goal of the task along with its expected challenges, uncertainties in challenges, and a measure of success.

Case A. Questions related to Quantitative Measure would ask for a score, say from 0 to 10, for the agent's performance in achieving the goal. Quantitative Measure questions would be: i) How clean is the vacuumed area? ii) How well does it cover areas, avoiding obstacles? The answer is: i) 10 out of 10 clean for the bathroom floor shown in the Figure. ii) 9 out of 10 for reaching areas that it can navigate to; 1 out of 10 for getting entangled with a loose item.

Case B. An elaborate scheme for scoring Quantitative Measure of IGVC agent performance is described on www.igvc.org. *Qualitative Measure for the Autonomous Challenge Event:* Judges will rank the entries that complete the course based on shortest adjusted time taken. In the event that a vehicle does not finish the course, the judges will rank the entry based on the longest adjusted distance traveled. Adjusted time and distance are the net scores given by judges after taking into consideration penalties, incurred from obstacle collisions, pothole hits, and boundary crossings. *Measure of success for Navigation Challenge:* Vehicles may seek the waypoints in any order, and the

vehicle actually reaching the most waypoints in the allotted seven minute run time will be the winner. If a vehicle fails to come within two meters of a target, it will not be judged to have reached that target.

5.3 Comparative Scale

Comparative Scale against Human Capability. As robotics and automation systems replace humans in carrying out all sorts of tasks, one may think of calibrating the levels of difficulty or challenges in the tasks against the skill capability of a person. In this way, one could gauge the intelligence of a robotics system to, say, the sensory-motor-decision skills of a qualified person. The person's qualification could provide a scale for rating the intelligence level required of a task.

Case A. There is no experimental data to establish the Comparative Scale for a Roomba. But it is safe to conjecture that the vacuuming agent has an equivalent skill set of a three year old child.

Case B. An experimental investigation into Comparative Scale for the autonomous challenge event was performed at the 1997 IGVC, held on the campus of Oakland University on May 28 - June 2, 1997. During the competition, experiments were conducted with the help of kindergarten children from the Lowry Child Care Center at Oakland University. The idea was to determine the performance and capability of children of ages two through six years in completing the course. The experimental data would provide a scale for gauging the performance of IGVC agents.

Experiment. In the field experiment, the children took turns driving a battery-powered cart (similar in size to the competition vehicles) around a 630 foot autonomous obstacle course. A camera recorded the driving behavior and head movements of the kids during the test. The videotape recorded the kids' initial driving reaction to the obstacle course and, in many cases, their improved driving skills that emerged from learned experience.

Subject samples. Twenty-two children participated. They were classified by age in Table 1 and labeled as 2X, 3X, 4X, 5X and 6X, where X is a letter in the alphabet. The experiment was assisted by a medical doctor (co-author), a day care provider from the Center, and the parents of the children. Prior to the experiment, a brief survey on each of the children was also conducted so that their background might be factored into consideration when interpreting their maneuvering skills. To check for presence of sensory-motor skill in the children sample, each child was tested by having him or her sketch a "figure of a person" on a paper. A sample of the sketches in Figure 5 showed that each child demonstrated some minimal necessary sensory-motor capability for the challenge at hand. The children's prior experience in driving a toy car and playing video games was also noted as shown in Table 1. However, the experience appeared to be irrelevant to the driving ability demonstrated in the field at this time.

Results. Figure 5 shows a montage of snapshots from the recorded experiment in progress. The photos A, B, & C demonstrate some of the challenges on the obstacle course. Photos D and G give an idea about unsuccessful attempts by a two-year-old. Photos E and H illustrate that three-year-olds have some problem steering and maneuvering the obstacle course. The obstacles often prove to be challenging even for some of the four-year old children, as evident in Photo F. Photo I shows a child receiving instructions; the child quickly learned how to handle the equipment. Table 1 describes the observations that support the experimental results and the photos.

The two-year old child had no idea initially on what to do, how to drive or steer the cart and about the objective of the experiment. In about 10 minutes, the child learned to drive the cart with the pedals. He held the steering wheel only as a support. The child did not achieve the objective of the obstacle challenge. Three-year old children could understand the objective of the challenge, and drive the cart properly. However, this age group had difficulty with motor skills, cart control, and accuracy. For the objective of the IGVC contest course, none of three-year-olds finished the course. Children of age four and above had little difficulty finishing the course. Two kids in this age group made some mistakes, but they all outperformed any of the robotic vehicles that competed in the IGVC. The five- and six-year-olds had a blast test-driving the obstacle course.

ID	Age (Yr/Mo)	Driven a toy cart?	Video games hrs/wk	Observation on driving and maneuvering skills ¹
2A	2/10	N	4	Initially clueless; manage to only throttle after 10 mins.
3D	3/3	Y	0	Hit two obstacles hard; did not finish course
3B	3/7	Y	0	Touched lines 6 times; did not finish
3E	3/7	N	0	Touched lines 2 times; did not finish
3A	3/9	N	0	Went out-of-bound at 315', did not finish
4I	4/4	Y	1	Hit one obstacle; finished course
4B	4/7	N	1	Finished course in 2:17 minutes (slow)
4D	4/8	N	3	Touched a boundary line; hit one obstacle; finished
4G	4/8	N	1	*
4H	4/8	Y	0	Finished course in 2:35 minutes (slow)
4J	4/8	N	3	*
4E	4/9	N	0	*
4F	4/11	N	0	*
5C	5/1	Y	7	Touched an obstacle; finished course
5F	5/1	N	0	*
5G	5/4	N	0	Finished course in 2:32 minutes (slow)
5A	5/9	Y	0	*
5D	5/9	Y	0	*
5B	5/11	Y	1	*
5H	5/11	N	3	*
6D	6/1	N	0	*
6E	6/2	Y	2	*
6C	6/4	Y	0	*
6A	6/7	Y	0	*
6B	6/11	Y	0	*

"*" entries indicate the child had no problem in completing the obstacle course under two minutes.

Table 1. Driving and maneuvering performance of children from the obstacle course experiment.

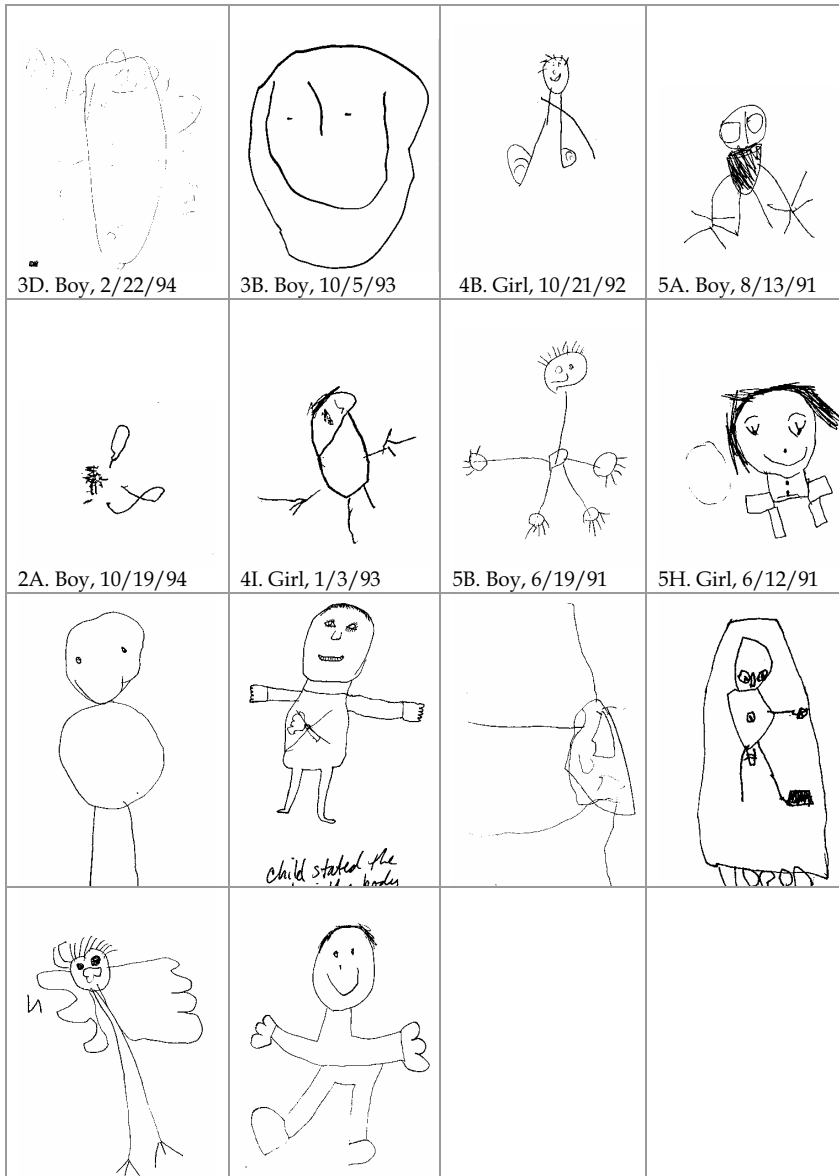


Fig. 5. Hand sketches used to confirm presence of sensory-motor capabilities.



Fig. 6. Field experiments in which children were challenged to complete the obstacle course at the IGVC.

Conclusion. An important conclusion from the experiment is that children approximately four-years old and above can handle the obstacle course. By virtue of comparison, a UVS that successfully completes obstacle course may be said to have an equivalent intelligent sensory-motor and control skill of a four-year old child.

6. Conclusion

The paper illustrates three means for gauging the intelligence of mobile robots: The *qualitative perception* of intelligence based on extending Turing test to the systems at hand; the *quantitative measure* that gauges task-specific intelligence from its performance in uncertain environments; the *comparative measure* that calibrates difficulty levels of what the system has to do against levels of human skills. The framework of an agent, an environment and a goal was applied to two case studies was helpful in gauging the intelligence of the mobile robots. An important field experiment reveals that the autonomous obstacle course challenge in the IGVC requires the *sensory-motor-decision skill* of at least a four-year old. As frontiers of and education efforts on IUVS are being pursued, it is necessary and worthwhile to consider means for improving as well as gauging the intelligence of the systems.

7. Reference

- Advanced robotics & intelligent machines (1996), J.O. Gray & D.G. Caldwell (Editors), IEE Control Engineering Series 51
- J.S. Albus & A.M. Meystel (2001) Engineering of mind – an introduction to the science of intelligent systems, Wiley-Interscience Publication
- Louis Armand (2005). Intelligence and Representation, 1000 Days of Theory, Arthur and Marilouise Kroker (Ed). www.ctheory.net/articles.aspx?id=496
- Ka C Cheok (2005), "Intelligent Autonomous Behavior Technologies for Small Unmanned Ground Vehicle Systems," Procs of the Int'l Conf on Intelligent Systems, Kuala Lumpur, Malaysia, 1-3 Dec 2005.
- Ka C Cheok, G.E. Smid, G.R. Lane, W.G. Agnew & A. Khan (2004), Gauging Intelligence of Unmanned Vehicle Systems – An IGVC Perspective, Journal of Robotics Systems, Vol. 21, No. 8, Aug 2004, pp. 405-418.
- The control handbook (1995), W. Levine (Ed), CRC Press/IEEE Press
- J.L. Jones, A.N. Flynn & B.A. Seiger (1999). Mobile robots – inspiration to implementation, 2nd edition, A.K. Peters Ltd
- S. Legg & M Hutter, (2006). A formal measure of machine intelligence, <http://arxiv.org/abs/cs.AI/0605024>
- F.G. Martin (2001). Robotic explorations – a hands-on introduction to engineering, Prentice-Hall
- Pamela McCorduck (2004). Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence.
- N.J. Nilsson (1980). Principles of artificial intelligence, Morgan Kaufmann
- S. Pinker (1997). How the mind works, W.W. Norton, New York

Steven Savage (2005), Magic and Technology. http://www.sffworld.com/authors/s/savage_steven/articles/magicandtechnology.html

A Reusable UART IP Design and its Application in Mobile Robots

Ching-Chang Wong & Yu-Han Lin
*Department of Electrical Engineering, Tamkang University,
Tamshui, Taipei 25137, Taiwan*

1. Introduction

A Universal Asynchronous Receiver and Transmitter (UART) is an integrated circuit which plays an important role in serial communication. The UART is a standard communication component that is provided by most of the available microprocessors. In general, the number of the UART in a microprocessor is limited and not enough for robot applications. In this chapter, a reusable Intellectual Property (IP) of UART design method by using VHDL Hardware Description Language (VHDL) is proposed and realized on a Field Programmable Gate Array (FPGA) chip. FPGA implementation is flexible because it can be easily reconfigured by the end user and reused for different designs. The proposed UART IP is composed of a baud rate generator, a receiver module, and a transmitter module. These modules are reusable and synthesizable. The popular N-8-1 (No parity (N), eight (8) data bits, and one (1) stop bit) data format is implemented in the proposed UART IP, but the parity setting, the number selection of data bits, and stop bits are user-enhancements. In this chapter, the proposed UART IP is applied to be a data detector of Infra-red (IR) ranging system to receive the distance information of objects, a data detector of digital compass to receive the head direction of robot, and a transceiver of a wireless modem to communicate with the other robots and a host computer. These application circuits have been implemented on a FPGA chip (Altera EP20K200EFC484-2X) to illustrate the robot can detect data from the IR ranging system and the digital compass exactly and communicate with the host computer and other robots successfully.

The importance for System-on-Chip (SOC) using a reusable IP is increasing in modern design methodology. The old design method is not suitable to design chip which operates on a required function in a given time. Therefore, using proper IP for requested specifications can reduce the design time and cope with time-to-market (Delforge, 1998). In this chapter, a reusable UART IP is designed for applications in the serial communication. A UART (Harvey, 1999; Michael, 1989) is an integrated circuit, which plays an important role in the serial communication. The UART contains a receiver (serial-to-parallel converter) and a transmitter (parallel-to-serial converter). It handles the conversion between serial and parallel data. Serial communication reduces the distortion of a signal, therefore makes data transfer between two systems separated in great distance possible. In most computer systems, the UART is connected to circuitry that generates signals that comply with the EIA (Electronic Industries Alliance) RS232-C specification. The advantages of UART systems are the simplicity of interconnection wiring and character transmission protocol and formats. Starting with the original IBM Personal Computer, IBM has selected the National Semiconductor INS8250 UART for use in the IBM PC parallel/serial

adapter. Subsequent generations of compatible computers from IBM and other vendors continued to use the INS8250 or improved versions of the National Semiconductor UART family. The 16550A and its successors have become the most popular UART design in the PC industry, mainly due to its ability to handle higher transfer rates of data reliably on operating systems with sluggish interrupt response times.

Recently, a reconfigurable computing system that uses the reconfigurable aspects of the FPGA to implement an algorithm has been attractive because it offers a compromise between special-purpose ASIC hardware and general-purpose processors. The FPGA system is flexible because it can be easily reconfigured by the end user and reused for many different designs. It also provides a rapid prototyping by synthesizing the desired system with an appropriate Electronic Design Automation (EDA) tool. The FPGA-based system is also useful because it can reduce development time greatly. In designing hardware, many design processes entail modelling hardware in varied levels of detail. Designs are described in a hardware description language like VHDL (Navabi, 1998; Roth, 1998) and are verified by simulation. Due to these useful features, we chose the reconfigurable FPGA system as the implementation platform of the mobile robot. The reusable UART IP designed in this chapter is composed of a baud rate generator, a receiver module, and a transmitter module. These modules are applied to the application of mobile robots, including the distance information detector of the IR ranging system, heading output detector of the digital compass module, and communications among soccer robots and a host computer.

This chapter is organized as follows: The concept of the UART and simulation of its individual modules are presented in Section 2. The applications of the reusable IP in autonomous robot soccer system are described in Section 3. The conclusions are given in Section 4.

2. UART IP Design

A UART frame consists of 1 start bit, a number of data bits, an optional parity bit and 1, 1.5, or 2 stop bits. The start bit goes low for one bit time, then a number of data bits are transmitted, least significant bit first, the number of data bits ranges is typically 5, 6, 7, or 8.

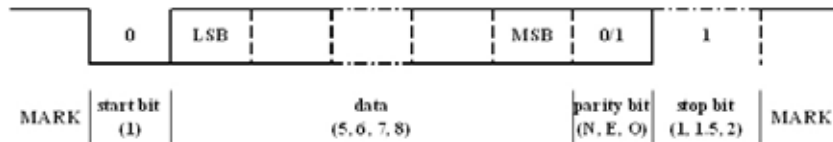


Fig. 1. Standard serial data format.

When no data is being transmitted, a logic 1 must be placed in the transmitted data line. Fig. 1 shows the standard format for serial data transmission. The number of data bits, the parity bit, and the number of stop bits must be set a priori in all communication partners. The design is minimalist, and no error checking logic is present by default. All of these features are to become user-enhancements. A frame format of 1 start bit, 8 data bits, 0 parity bit, and 1 stop bit is supported by all examined UART. In this chapter, the N-8-1 data format is implemented in the proposed UART IP, but the parity setting, the number selection of data bits, and stop bits are user-enhancements. Many UART perform multiple sample points to detect a bit cell and decide on a majority vote. This method affords a multiple of the sampling frequency for single bit detection but provides immunity to shorts spike disturbances on the communication line. The

UART IP is composed of a Baud Rate Generator (BRG), a receiver module, and a transmitter module. The design methods of these modules are described as follows.

2.1. Baud Rate Generator (BRG)

The number of bits transmitted per second is frequently referred to as the baud rate. The BRG divides the system clock by a divisor to provide standard RS-232C baud rate clock (bclk) and 8 times the data rate clock (bclkx8) for general purpose system use when using any one of three industry standard baud rate crystals (1.8432MHz, 2.4576MHz, and 3.072MHz). The divisor can be calculated as follows:

$$n = \frac{f_{\text{clk}}}{\text{BR}_{\text{max}} \times C \times 2} \quad (1)$$

where f_{clk} is the system clock frequency. The integer constant C represents the number of samples per bit cell. BR_{max} is the maximum baud rate frequency. We assumed that the system clock is 1.8432MHz and we want baud rates 300, 600, 1200, 2400, 4800, 9600, 19200, and 38400. The number of samples per bit cell is eight. Then the divisor can be calculated as 3 ($1843200 / 38400 \times 8 \times 2 = 3$). The architecture of the BRG is shown in Fig. 2. The f_{clk} is first divided by n using a counter. This counter output goes to an 8 bits binary counter. The outputs of the flip-flops in this counter correspond to divided by 2, 4, ..., and 256. One of these outputs is selected by a multiplexer. The simulation result of the BRG is shown in Fig. 3.

sel[2:0]	bclk	bclkx8
000	38400	307200
001	19200	153600
010	9600	76800
011	4800	38400
100	2400	19200
101	1200	9600
110	600	4800
111	300	2400

Table 1. Frequencies generated table.

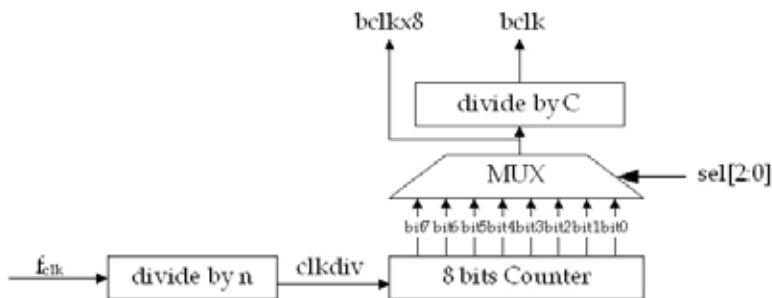


Fig. 2. Architecture of the baud rate generator.

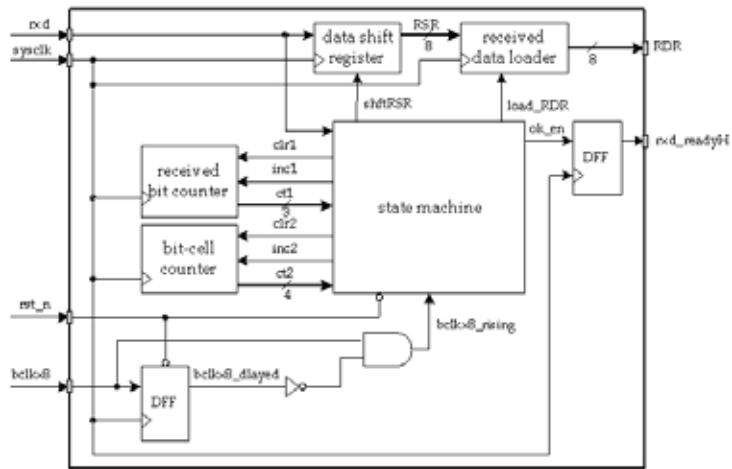


Fig. 5. Functional block-diagram of the UART receiver.

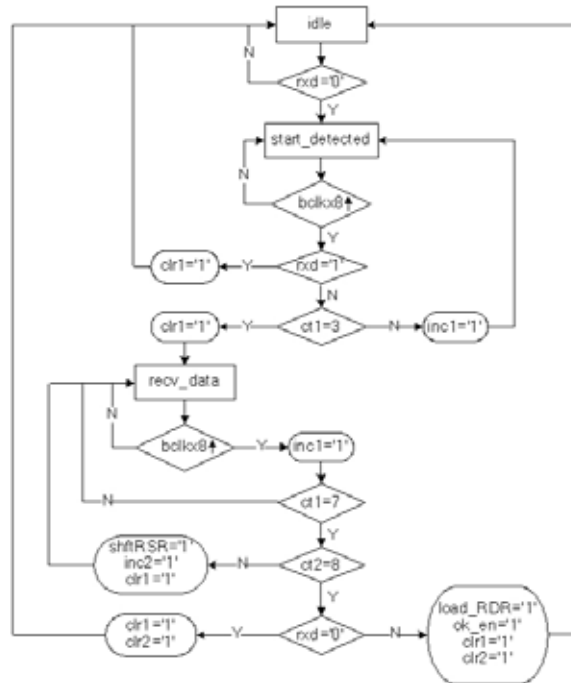


Fig. 6. SM chart for UART receiver.

Fig. 6 illustrates the SM chart of the UART receiver. The state machine is the Mealy machine and composed of three states (idle, start_detected, and recv_data). Two counters are used. ct1 counts the number of bclkx8 clocks. ct2 counts the number of bits received after the start bit. In the idle state, the SM waits for the start bit and then goes to the start_detected state. The SM waits for the rising edge of bclkx8 and then samples rxd again. Since the start bit should be '0' for eight bclkx8 clocks, we should read '0'. ct1 is still 0, so ct1 is incremented and the SM waits for bclkx8. If rxd='1', this is an error condition and the SM clears ct1 and resets to the idle state. Otherwise, the SM keeps looping. When rxd is '0' for the fourth time, ct1 = 3, so ct1 is cleared and the state goes to recv_data state. In this state, the SM increments ct1 after every rising edge of bclkx8. After the eighth clock, ct1=7 and ct2 is checked. If it is not 8, the current value of rxd is shifted in to RSR, ct2 is incremented, and ct1 is cleared. If ct2=8, all 8 bits have been read and we should be in the middle of the stop bit. If rxd = '0', the stop bit has not been detected properly, the SM clear ct1 and ct2 and resets to the idle state. If no errors have occurred, RDR is loaded from RSR and two counters are cleared and ok_en is set to indicate that the receive operation is completed. The simulation result of the UART receiver receiving 0x53 is shown in Fig. 7.

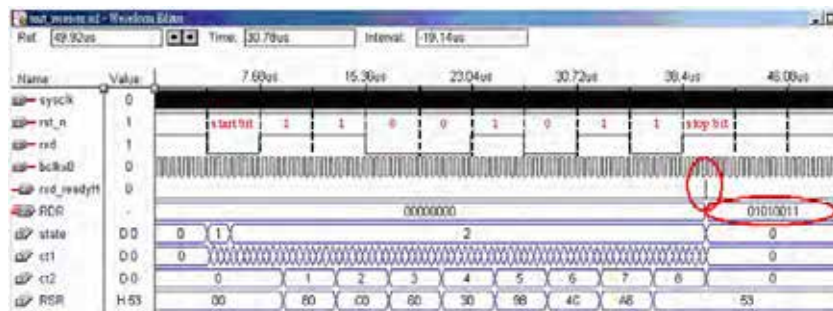


Fig. 7. UART receiver receiving 0x53.

2.3 Transmitter Module

The transmitter circuitry converts a parallel data word into serial form and appends the start, parity, and stop bits. The transmitter of UART is composed of transmitted bit counter, a data shift register, a state machine and support logic. Fig. 8 illustrates the functional block diagram of the UART transmitter. The transmitted bit counter has the same function and implementation as that of the receiver, only the signal name have changed slightly.

The data shift register is an 8 bits parallel-in-serial-out shift register. It has 3 control inputs: loadTSR, start and shiftTSR. An active high on the first signal loads the parallel data into the shift register. An active high on the second signal transmits a start-bit (logic 0) for one bit time. An active high on the last signal shifts the loaded data out by 1 bit. When the transmit operation is completed, the txd_doneH will be a high signal for a system clock period.

The data shift register is an 8 bits parallel-in-serial-out shift register. It has 3 control inputs: loadTSR, start and shiftTSR. An active high on the first signal loads the parallel data into the shift register. An active high on the second signal transmits a start-bit (logic 0) for one bit time. An active high on the last signal shifts the loaded data out by 1 bit. When the transmit operation is completed, the txd_doneH will be a high signal for a system clock period.

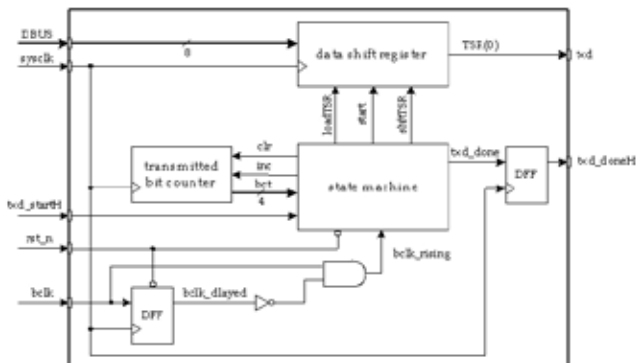


Fig. 8. Functional block-diagram of the UART transmitter.

Fig. 9 illustrates the SM chart of the UART transmitter. The state machine is the Mealy machine and composed of three states (idle, synch, and tdata). In the idle state, the SM waits until tx_startH has been set and then loads DBUS data into higher eight bits of TSR register. The TSR register is a nine bits data register and the low-order bit is initials to '1'. In the synch state, the SM waits for the rising edge of the bit clock (bclk↑) and then clears the low-order bit of TSR to transmit a '0' for one bit time. In the tdata state, each time bclk↑ is detected, TSR is shifted right to transmit the next data bit and the bit counter (bct) is incremented. When bct=9, 8 data bits and a stop bit have transmitted, bct is then cleared and tx_done is set to indicate that the transmit operation is completed. The simulation result of the UART transmitter transmitting 0xa5 is shown in Fig. 10.

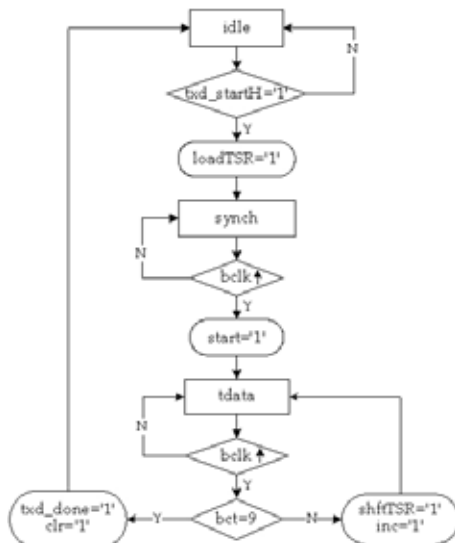


Fig. 9. SM chart for UART transmitter.

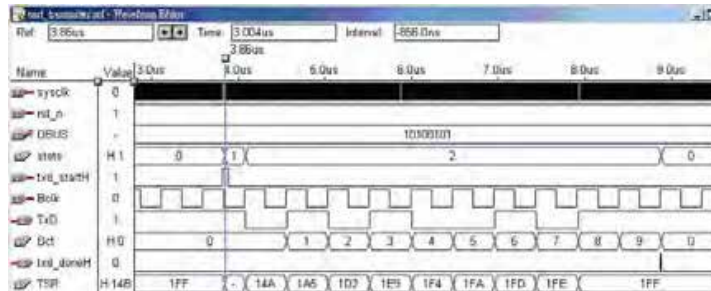


Fig. 10. UART transmitter transmitting 0xa5.

3. Application to Autonomous Soccer Robots

The reusable UART IP is used to detect the information of the IR ranging system and digital compass module of autonomous soccer robots. The communication between robots are using RF module.

3.1 IR Ranging System Data Detector

The IR ranging system of the autonomous robot soccer system we adopted is DIRRS+, which is manufactured by HVW Technologies Inc. (HVW Technologies Inc., 1998), and has a range of 10 to 80 cm. The actual photo of digital infra-red ranging system is shown in Fig. 11. The distance is output by the sensor on a single pin as a digital 8-bit serial stream. The specification of the serial stream is RS-232 8-N-1 at 4800 bps. A close object reports a larger value, and a distant object reports a smaller value. The BRG and receiver module of the UART IP can be used to receive the distance information of the IR ranging system.



Fig. 11. Digital infra-red ranging system.

3.2 Digital Compass Data Detector

The digital compass module of the autonomous robot soccer system we adopted is TDCM3, which is manufactured by Topteam Technology Corp. (Topteam Technology Corp., 2001). The digital compass outputs compass heading via electronic interface to a host system. It outputs compass readings by a host request. The interface is RS-232 8-N-1 at 4 levels speed to transmit data, these are 2400, 4800, 9600, 19200 bps. When the host sends a pulse via RTS pin to the device it will output heading via TX pin to the host. Before the host sends a pulse to the device the RX pin needs to be kept in high level. The normal mode timing diagram is shown in Fig. 12. The data

format is Status, θ_{MSB} and θ_{LSB} . The “status byte” is a flag that shows the TDCM3 status. In normal state the “status byte” is equal to 80H when distortion is detected the “status byte” is equal to 81H. The compass heading can be calculated as follows:

$$\theta = (\theta_{MSB} \times 256 + \theta_{LSB}) / 2 \quad (2)$$

The BRG and receiver module can be used to receive the heading information of the digital compass.

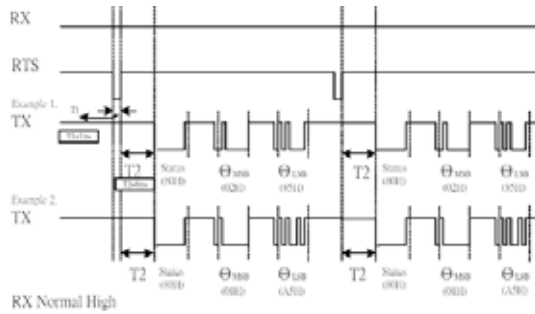


Fig. 12. Timing chart of normal mode.

3.3 Communication of robots

Autonomous robot soccer system consists of a multi-agent system that needs to cooperate in a competitive environment. Thus, it is a crucial issue how to realize effective and real time communications among soccer robots. We use the SST-2400 radio modem which is developed by IPC DAS Corp. (IPC DAS Corp., 1999) to construct a wireless LAN. The actual photo of SST-2400 wireless radio modem is shown in Fig. 13. The SST-2400 is a spread spectrum radio modem with RS-232 interface port. Based on direct sequence spread spectrum and RF technology operating in ISM bands. The Frequency Range is 2400MHz~2483.5MHz. Considering about experiment requirements for multi-robots communication, we set the wireless modems in the operation mode2 which is point to multi-point, half-duplex, synchronous, the fixed data format 8-N-1, and 9600 baud rate. Three modules of the UART IP can be used to receiving and transmitting data among host computer and other robots.



Fig. 13. SST-2400 wireless radio modem.

4. Conclusions

A reusable UART IP design and its application in mobile robots are presented in this chapter. The UART IP is composed of a baud rate generator, a receiver module, and a transmitter module. These modules are reusable and synthesizable and they are used to be a data detector of IR ranging system to receive the distance information of objects, a data detector of digital compass to receive the head direction of robot, and a transceiver of a wireless modem to communicate with the other robots and host computer. These application circuits have been implemented on a FPGA chip to illustrate the robot can detect data from the IR ranging system and the digital compass exactly and communicate with the host computer and other robots successfully.

5. References

- Delforge, P. (1998). IP Re-use Model, IP Market and IP Corporate Database, Proceedings of Intellectual Property System on Chip Conference, pp. 11-16, Mar, 1998.
- Harvey, M.S. (1999). Generic UART Manual, Silicon Valley, December, 1999.
- Michael, M.S. (1989). A Comparison of the INS8250, NS16450 and NS16550AF Series of UARTs, National Semiconductor Application Note 493, April, 1989.
- Navabi, Z. (1998). *VHDL: Analysis and Modeling of Digital Systems*, 2nd Edition, McGraw-Hill, 0070464790, New York.
- Roth, Charles H. (1998). *Digital System Design Using VHDL*, PWS, 053495099X, Boston, Mass. *Data Sheet DIRRS+*, Alberta, Canada, HVW Technologies Inc., 1998.
- Data Sheet TDCM3*, Taiwan, Topteam Technology Corp., 2001.
- User's Manual: SST-2400 Radio Modem Module*, Taiwan, ICP DAS Corp., 1999.

Intelligent Pose Control of Mobile Robots Using an Uncalibrated Eye-in-Hand Vision System

T. I. James Tsay and Y. F. Lai
*National Cheng Kung University
Taiwan (R.O.C.)*

1. Introduction

The use of material transfer units constructed from AGVs and robot manipulators is becoming popular in production lines with small production volumes and periods. This new material transfer unit is also called a mobile robot, which can quickly adapt to the change of the factory layout. A mobile robot is sufficiently mobile that it can very flexibly perform tasks in the production line. A guidance control system drives a mobile base with a robot manipulator mounted on it to a predefined station so the robot manipulator can pick up a workpiece in the station.

During pick-and-place operations, the relative location between a predefined station and the robot manipulator cannot be anticipated due to the possibility of the non-planar ground and/or the positioning errors caused by the guidance control system of the mobile base. Uncertainties in either the location of the mobile base or the level of ground flatness are responsible for a position mismatch, which normally causes failure in the pick-and-place operation. A more direct way of dealing with this problem is to use a CCD camera to provide visual feedback. The camera is typically mounted on the end-effector of the robot manipulator to compensate for these uncertainties using a visual servoing scheme.

Hutchinson et al., 1996 classified the visual feedback control systems into four main categories. Position-based control aims to eliminate the errors determined by the pose of the target with respect to the camera; the features extracted from the image planes are used to estimate the pose in space. The control law then sends the command to the joint-level controllers to drive the servomechanism. This control architecture is the so-called hierarchical control and is referred to as the dynamic position-based look-and-move control structure. The control architecture is referred to as the position-based visual servo control structure if the servomechanism is directly controlled by the control law mentioned above rather than by the joint-level controller. The errors to the control law in image-based control, are directly governed by the extracted features of the image planes. The control architecture is hierarchical and referred to as the dynamic image-based look-and-move control structure if the errors are then sent to the joint-level controller to drive the servomechanism. The visual servo controller eliminates the joint-level controller and directly controls the servomechanism. Such control architecture is termed an image-based visual servo control structure.

The mobile robots (Ting et al., 1997) made by Murata Machinery, Ltd. meet the necessary cleanliness standards required in current semiconductor production facilities. The mobile base of a mobile robot is first guided to a predefined station using CCD cameras mounted on the ceiling. Then, the eye-in-hand manipulator performs a pick-and-place operation by employing a static position-based look-and-move control structure. A pair of LEDs is installed in front of a specific location, where a cassette is picked up or put down, to provide a reference frame. The relative positions of the LEDs and the cassette is fixed and known. The trajectory along which the end-effector of the manipulator approaches the cassette can be divided into two parts. The first part of the trajectory is determined off-line using a teaching box to drive the end-effector to a location that allows the CCD camera mounted on the end-effector to obtain a pose to capture the image of both LEDs. After the end-effector follows the part of the trajectory planned off-line, the CCD camera guides the second part of the trajectory. The location of the cassette with respect to the manipulator base frame is first computed through the coordinate transformations and perspective transformation in response to the image of two LEDs. The end-effector is then commanded to move to the above computed location and pick up the cassette. The use of two LEDs makes success in the pick or place task highly dependent on the pose of the end-effector relative to the LEDs when the image of the two LEDs is captured. Consequently, the non-horizontality of the ground and the large positioning errors of the mobile base may cause the pick or place task to fail.

Beyond the conventional visual servo control methods referred to above, behavior-based methods for visual servo control have already been developed in the literature (Anglani et al., 1999; Kim et al., 2001; Wasik & Saffiotti, 2002, 2003). A behavior-based system has been proposed to perform grasping tasks in an unstructured environment, in cases in which the position of the targets is not already known (Wasik & Saffiotti, 2002, 2003). The controller maps input from the image space to the control values defined in the camera space. The control values are then transformed to joint controls by a Jacobain transformation, revealing that either a hand-eye calibration process has been implemented or the hand-eye relationship is known beforehand.

This study employs an uncalibrated eye-in-hand vision system to provide visual information for controlling the end-effector of the manipulator to approach and grasp the target workpiece. This work focuses on developing a novel behavior-based look-and-move control strategy to guide the manipulator to approach the object and accurately position its end-effector in the desired pose. Notably, the pose of the workpiece in relation to the end-effector of the manipulator is never numerically estimated. Also, the relationships between the deviations of the image features and the displacements of the end-effector are never determined. This strategy is based on six predefined image features. In the designed neural fuzzy controllers, each image feature is taken to generate intuitively one DOF motion command relative to the camera coordinate frame using fuzzy rules, which define a specific visual behavior. These behaviors are then combined and executed in turns to perform grasping tasks.

2. Image Processing

This section proposes the image processing method that we used to extract target object features in a visual servo system. In this study, the workpiece to be picked up by the

manipulator's end-effector is a rectangular parallelepiped. The images of the workpiece are captured from above by a CCD camera mounted on the end-effector, as the end-effector moves toward the workpiece. The image of the workpiece's top surface is a quadrangle. Only information about the quadrangle is of interest, so the captured image is firstly preprocessed to obtain a clear image of the quadrangle. Then, the selected image features can be calculated from the quadrangle.

Fig. 1 presents a complete image processing flowchart. The captured image is first preprocessed using the thresholding technique and opening and closing operations to yield a clear binary image of the workpiece's top surface, from which the area, the center of gravity, and the principal angle of the quadrangle in the image plane can be obtained. The Laplace operator is then applied to estimate quickly the approximate locations of the corners. Once the approximate corner locations are determined, four square windows can be obtained from the approximate locations of four corners, (x_i, y_i) , $i = 1, 2, 3, 4$, as shown in Fig. 2, to specify that all the edge points encircled by each window are on the same edge, to determine the precise location of the corners. Each square window can be established by the following procedure.

1. Select two adjacent corners, (x_1, y_1) and (x_2, y_2) .
2. Determine the width and length of the window, w and l .

$$w = |x_1 - x_2| \times 0.8 \text{ pixels}$$

$$l = |y_1 - y_2| \times 0.8 \text{ pixels}$$

Neither the width nor the length of the window are chosen as the full coordinate differences between two adjacent corners' locations to prevent incorrect encirclement due to the error in quickly estimating the approximate locations of the corners. However, if $|x_1 - x_2| < 5$ pixels, then the width is defined as $w = 10$ pixels, to ensure that at least a few edge points are encircled by the window, and if $|y_1 - y_2| < 5$ pixels, the length is defined as $l = 10$ pixels, to ensure that at least a few edge points are encircled by the window.

3. Calculate the coordinates of the window's upper left corner and the window's lower right corner, $w_{\text{left}}^{\text{upper}} = (x_{\text{left}}^{\text{upper}}, y_{\text{left}}^{\text{upper}})$ and $w_{\text{right}}^{\text{lower}} = (x_{\text{right}}^{\text{lower}}, y_{\text{right}}^{\text{lower}})$, from the following equations.

$$x_{\text{left}}^{\text{upper}} = (x_1 + x_2) / 2 - w / 2$$

$$y_{\text{left}}^{\text{upper}} = (y_1 + y_2) / 2 - l / 2$$

$$x_{\text{right}}^{\text{lower}} = (x_1 + x_2) / 2 + w / 2$$

$$y_{\text{right}}^{\text{lower}} = (y_1 + y_2) / 2 + l / 2$$

The coordinates of points on a single edge can then be fitted into a line equation using least-error-squares. The precise location of each corner can then be obtained from the point of intersection of two adjacent lines.

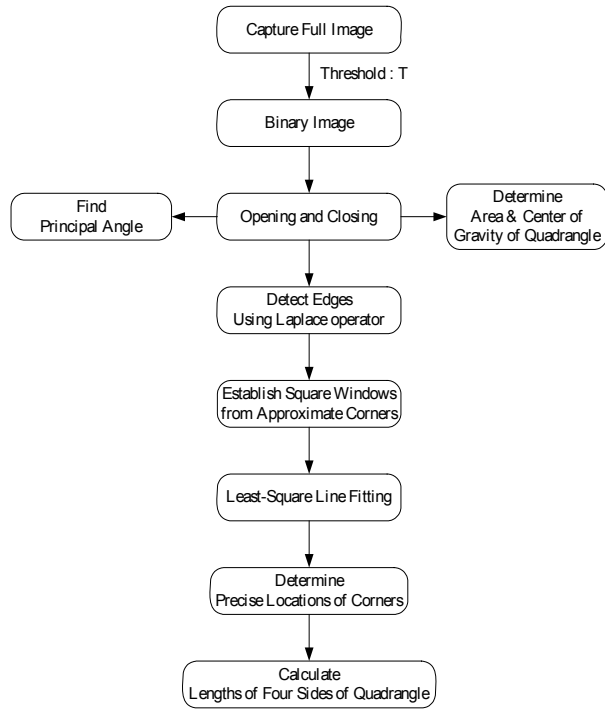


Fig. 1. Complete image processing flowchart.

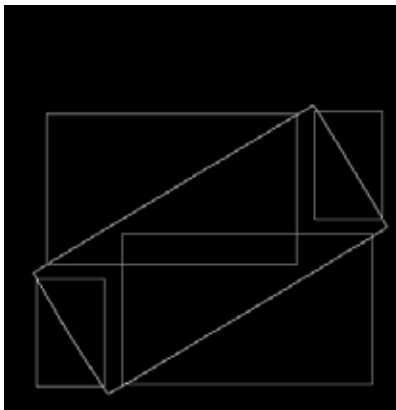


Fig. 2. Using corners' approximate locations to specify four windows to yield four line equations.

3. Control Strategy for Approaching the Target

Six designed neural fuzzy controllers map image features in the image space onto motion commands in the camera space. Also, each mapping from image space to camera's Cartesian space is mediated by fuzzy rules, and defines a particular vision-based behavior. Notably, these behaviors are defined from the perspective of the camera but not from that of the end-effector or external observer. This section defines a decomposition of the manipulation task into a set of basic behaviors, which are combined so that the eye-in-hand camera can gradually reach the desired position and orientation. The camera is fixed on the end-effector, so when the end-effector reaches the desired pose, the gripper can be commanded to grasp the workpiece.

3.1 Selection of Image Features

Six image features are adopted to determine the translational and orientational motion in 3D. Each image feature can uniquely direct one D.O.F. of motion relative to the camera frame. Before the image features are defined, the coordinate symbols are explicated. Specifically, as shown in Fig. 3, let $\delta^E X_1$, $\delta^E X_2$ and $\delta^E X_3$, represent differential changes in translation along the ${}^E X$, ${}^E Y$ and ${}^E Z$ axes of the end-effector frame. Let $\delta^E X_4$, $\delta^E X_5$ and $\delta^E X_6$ denote the differential changes in orientation about the ${}^E X$, ${}^E Y$ and ${}^E Z$ axes of the end-effector frame. Additionally, the camera frame must be considered. The origin of the camera frame is at the center of the camera, while its z axis is the optical axis of the camera. Similarly, $\delta^C X_i$ for $i=1, 2, \dots, 6$, represent the differential movements with respect to the camera frame.

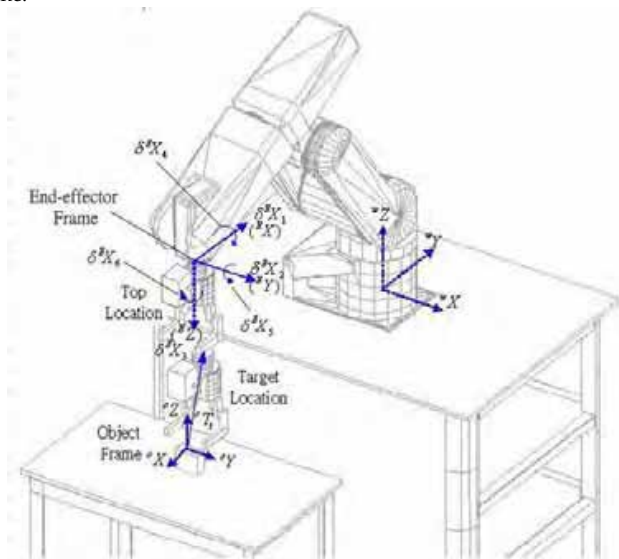


Fig. 3. End-effector and object frames.

In this study, the workpiece to be picked up by the manipulator's end-effector is a rectangular parallelepiped with a quadrangular image, as illustrated in Fig. 4. Six image features are extracted. Five image features, F_1 , F_2 , F_4 , F_5 and F_6 used in (Suh & Kim, 1994) are adopted. The other image feature called relative distance (RD) used in (Bien & Park, 1993) is taken as F_3 . Clearly, the image features are defined as follows.

$F_1 = V$, coordinate of the center of gravity of the quadrangle in the image plane. It directs camera motion cX_1 ;

$F_2 = U$, coordinate of the center of gravity of the quadrangle in the image plane. It directs camera motion cX_2 ;

$F_3 = RD = (A_1 - A_2) / A_1$, where A_1 is the area of the quadrangle in the desired pose, and A_2 represents the area of the quadrangle in the present pose. F_3 guides camera motion cX_3 ;

$F_4 = L_1 / L_2$, ratio of lengths of two opposite sides of the quadrangle about the axis, which passes through the center of gravity of the quadrangle and is parallel to the V axis in the image plane. F_4 directs camera motion cX_4 ;

$F_5 = L_3 / L_4$, ratio of lengths of two opposite sides of the quadrangle about the axis, which passes through the center of gravity of the quadrangle and is parallel to the U axis in the image plane. F_5 directs camera motion cX_5 , and

$F_6 = \text{angle between the } U \text{ axis and the axis of minimum moment of inertia of the quadrangle in the image plane. It guides camera motion } {}^cX_6$.

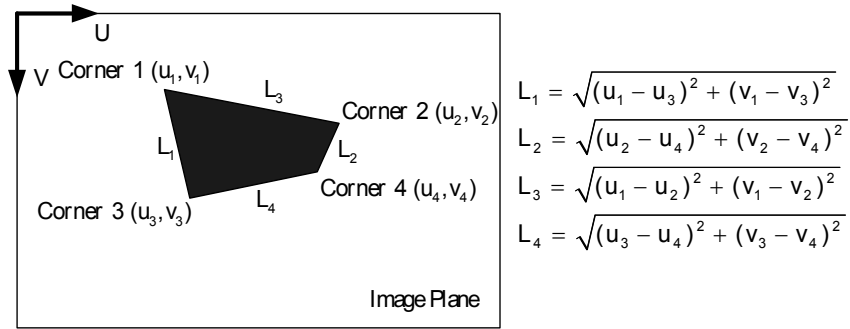


Fig. 4. Quadrangle in the workpiece image.

Each δF_i , for $i=1, 2, \dots, 6$, is defined as the difference between F_i and the feature value of the reference image at the target location, F_i' . F_i' is the value of F_i measured by the teach-by-showing method. This reference image is captured by the vision system when the end-effector is driven by a teaching box to the target location, cT_i , relative to the object frame, as shown in Fig. 3. The target location is defined as follows. [The end-effector is initially driven by a teaching box to a location that allows the gripper to grasp the workpiece. Then, the

end-effector is driven to another location, which is a safe distance from the preceding location (in this case, 10 cm above it). This “another location” is called the “target location”.] The reference features that correspond to the reference image at the target location are F_1^r , F_2^r , F_3^r , F_4^r , F_5^r and F_6^r .

3.2 Motion Planning Based on Behavior Design

In this work, fuzzy rules are used to enable the controllers to map image features in the image space onto motion commands in the camera space. These are then transformed to the commands in relation to the end-effector frame, which eventually control the manipulator. The final control values are relative motion commands sent to the position controller of the manipulator. The primary motivation for this process is that, after the notion of the camera frame has been introduced, the control rules for implementing basic vision-based motions such as “Center” or “Zoom” can be very easily written. No analytical model of the system is required.

1) Approach and Surround

The complete manipulation task involved in implementing a human-like visual servoing method is first divided into two complex behaviors - Approach and Surround - and one basic operation, Catch. The Approach behavior is the translational motion of the camera toward the workpiece, which is further divided into two basic behaviors - Center and Zoom. The Surround behavior is the orientational motion of the camera to keep the workpiece in the gripper, and is further divided into three basic behaviors - Yaw, Pitch and Roll. The Catch is a non vision-based operation. Only when the end-effector has reached the target location is the Catch activated and moves the end-effector 10 cm forward. The gripper then closes to grasp the workpiece. Fig. 5 displays the hierarchical composition of the behaviors, which are defined as follows.

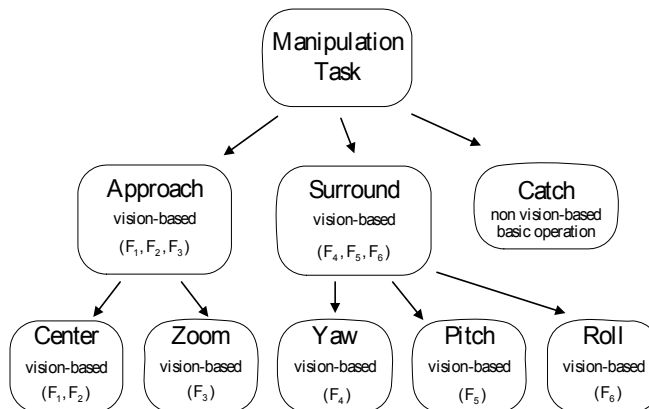


Fig. 5. Hierarchical composition of the behaviors.

Center is based on the first two image features, F_1 , and F_2 . This behavior translates the camera along the cX and cY axes of the camera frame to keep the center of gravity of the quadrangular image at the desired pixel in the image plane.

Zoom is based on F_3 ; it moves the camera along the cZ axis of the camera frame to keep the size of the object as a predefined value.

Yaw is based on F_4 ; it rotates the camera about cX to keep the ratio of the lengths of the two short sides equal to F_4^r .

Pitch is based on F_5 ; it rotates the camera about cY to keep the ratio of the lengths of the two long sides equal to F_5^r .

Roll is based on F_6 ; it rotates the camera about cZ so that the principal angle equals that in the reference image, in which the gripper's two fingers are arranged parallel to the short sides of target.

Vision-based behaviors are defined from the perspective of an eye-in-hand camera, so movements are performed relative to the camera frame.

2) Neural Fuzzy Controller

The main shortcoming of traditional fuzzy controllers is that they are unable to learn. The best control law or membership functions can be determined by experience. However, the manipulation tasks are non-linear and coupled. None set of membership functions is good for the entire work environment. With respect to learning capacity of the artificial neural network, back-propagation architecture is the most popular and effective for solving complex and ill-defined problems. Therefore, six simple neural fuzzy controllers (NFCs), employing back-propagation (Kim et al., 1995; Nomura et al., 1992) are designed herein. One image feature is input to each controller, which changes one D.O.F. of the camera motion as the output. The back-propagation algorithm is used only to adjust the consequents of fuzzy rules for each neural fuzzy controller at each iteration during the manipulation. Restated, the camera is guided intuitively according to the image features on the image plane. For example, if the area of the quadrangular image is smaller than that in the reference image, the camera appears to be far from the workpiece, and so the camera is moved forward. Otherwise, the camera is moved backward. The other five D.O.F. of motion of the camera are controlled in the same manner.

Fuzzy singleton rules are adopted to simplify the neural fuzzy controllers; they are defined as follows;

$$\text{if } \delta F_i \text{ is } A_i^j, \text{ then } \delta^c X_i^* \text{ is } w_i^j \quad (1)$$

where input variable δF_i is an image feature error; output variable $\delta^c X_i^*$ denotes a relative motion command in the camera frame; A_i^j are linguistic terms of the precondition part with membership functions $\mu_{A_i^j}(\delta F_i)$, and w_i^j represent the real numbers of the consequent part, $i = 1, 2, \dots, 6$ and $j = 1, 2, \dots, 7$. That is, each $i = 1, 2, \dots, 6$, can be regarded as a neural fuzzy controller with seven rules to control one D.O.F. of motion relative to the camera frame. Such a neural fuzzy system has a network structure as presented in Fig. 6. Herein, a simplified defuzzifier is used. The final output $\delta^c X_i$ of the neural fuzzy system is calculated by,

$$\delta^c X_i = \frac{\sum_{j=1}^7 \mu_{A_i^j} w_i^j}{\sum_{j=1}^7 \mu_{A_i^j}} \quad (2)$$

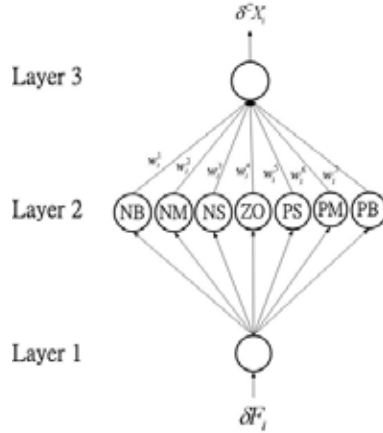


Fig. 6. Structure of the neural fuzzy controller with fuzzy singleton rules.

The parameter learning of NFC with fuzzy singleton rules can be the tuning of the real numbers W_j^i and the input Gaussian membership functions $\mu_{A_i}(\delta F_i)$, including the mean-points and the standard deviations (Lin & Lee, 1996). In this investigation, the mean-points and the standard deviations of the input membership functions are fixed to simplify parameter tuning. Only real numbers W_j^i are tuned on-line. Accordingly, the error function to be minimized is defined by

$$E_1 = \frac{1}{2} (F_1' - F_1)^2 = \frac{1}{2} (\delta F_1)^2 \quad (3)$$

The derivative of an error function E_1 with respect to the j -th consequent can be obtained as

$$\frac{\partial E_1}{\partial W_j^i} = \frac{\partial E_1}{\partial (\delta^c X_i)} \frac{\partial (\delta^c X_i)}{\partial W_j^i} \quad (4)$$

However, unfortunately, determining $\partial E_1 / \partial (\delta^c X_i)$ is difficult since the dynamics should be taken into account to see how the change in control input influences E_1 . Trickily, however, $\partial E_1 / \partial (\delta^c X_i)$ can be approximately computed from the difference ratio.

$$\frac{\partial E_1}{\partial (\delta^c X_i)} \approx \frac{dE_1}{d(\delta^c X_i)} \approx \frac{E_1(t) - E_1(t-1)}{\delta^c X_i(t) - \delta^c X_i(t-1)} \quad (5)$$

To decrease E_1 with respect to w_j^i , the consequent changes at time stage t , and $\Delta W_j^i(t)$ can be chosen to be proportional to $-\partial E_1 / \partial W_j^i$.

$$\Delta W_j^i(t) = -\eta \frac{\partial E_1}{\partial W_j^i} \quad (6)$$

where η is the learning-rate parameter. Hence, a learning rule for adapting the consequent at time stage t can be given as

$$w^l(t+1) = w^l(t) + \Delta w^l(t) \quad (7)$$

3.3 Rough Motion Transformation

In a manufacturing environment, hand-eye calibration process is generally time consuming. In this study, the pose of the camera in relation to the end-effector is invariant, as shown in Fig. 7, so the camera and the end-effector can be treated as a rigid body. After the motion of the rigid body has been analyzed, the transformation from the output values in relation to the camera frame to the motion commands with respect to the end-effector frame is obtained.

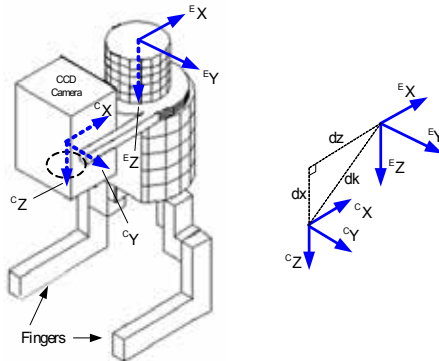


Fig. 7. End-effector and camera frames.

Fig. 8 reveals that if the controller output is only a rotation $\delta^c X_6$ about $^c Z$ but the command sent to the manipulator is a rotation about $^e Z$, then the unexpected camera displacements in the $^c X$ and $^c Y$ directions are $\delta^c X_1^+$ and $\delta^c X_2^+$, respectively. Similar situations occur in $\delta^c X_4$ and $\delta^c X_5$. Therefore, the motion command sent to the manipulator should be transformed as,

$$\begin{aligned} \delta^e X_1 &= \delta^c X_1 - dz \cdot \sin(\delta^c X_6) \cdot \tan(\delta^c X_6 / 2) - \\ &\quad 2dk \cdot \sin(\delta^c X_5 / 2) \cdot \cos(90^\circ - \delta^c X_5 / 2 - \tan^{-1}(dx/dz)) \\ \delta^e X_2 &= \delta^c X_2 + dz \cdot \sin(\delta^c X_6) + dx \cdot \sin(\delta^c X_4) \\ \delta^e X_3 &= \delta^c X_3 + dx \cdot \sin(\delta^c X_4) \cdot \tan(\delta^c X_4 / 2) - \\ &\quad 2dk \cdot \sin(\delta^c X_5 / 2) \cdot \sin(90^\circ - \delta^c X_5 / 2 - \tan^{-1}(dx/dz)) \\ \delta^e X_4 &= \delta^c X_4 \\ \delta^e X_5 &= \delta^c X_5 \\ \delta^e X_6 &= \delta^c X_6 \end{aligned} \quad (8)$$

where dx represents the distance between axes $^c X$ and $^e X$, and dz is the distance between axes $^c Z$ and $^e Z$. The two lines associated with dx and dz are assumed to be mutually perpendicular. They are measured roughly using a ruler and the naked eye. Accordingly,

$dk = \sqrt{(dx)^2 + (dz)^2}$ is assumed. Apparently, this hand-eye configuration is inaccurate. However, the designed neural fuzzy controllers handle the inaccuracy by tuning the consequents of the fuzzy rules according to the back-propagation algorithm. This process saves considerable time without the intensive computation associated with hand-eye calibration.

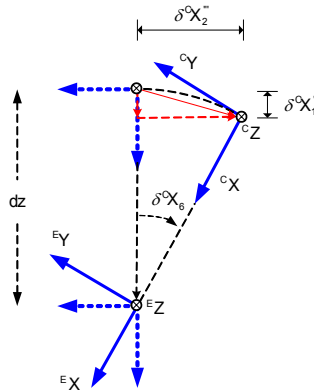


Fig. 8. Unexpected camera displacement caused by a rotation about E_Z .

3.4 Control Strategy

When the camera is far from the workpiece, the image features F_4 and F_5 are unstable because they are sensitive to the corner detection in image processing. Consequently, the Yaw and Pitch behaviors are activated only when the camera is close to the workpiece. On the contrary, the influence of distance and illumination on F_1 , F_2 , F_3 and F_6 is negligible.

Given the above restriction, the control strategy is as follows. Initially, the Approach behavior and the Roll behavior occur simultaneously in the approaching stage, in which the camera is moved toward the target. This process is iteratively performed until $|\delta F_1|$, $|\delta F_2|$, $|\delta F_3|$ and $|\delta F_6|$ are below the specified limiting values, ε_1 , ε_2 , ε_3 and ε_6 , respectively. The fine positioning strategy is then activated. In the fine positioning stage, Yaw and Pitch run concurrently to adjust the orientation of the camera. Approach and Roll are then executed again. These two processes iteratively run in turns until $|\delta F_i|$ is less than ε_i , $i = 1, 2, \dots, 6$. Finally, the basic operation Catch is inspired. Fig. 9 shows the behavior-based look-and-move control structure with rough motion transformation. Fig. 10 presents the control strategy flowchart.

For each step of motion, the motion command relative to the camera frame is transformed to a command relative to the end-effector frame using the proposed rough motion transformation. The inaccuracy of the hand-eye relationship can be neglected by adjusting on-line the singletons of the consequent parts in the fuzzy rules. The adjustment is implemented by the back-propagation algorithm that exploits a gradient descent method to reduce image feature errors.

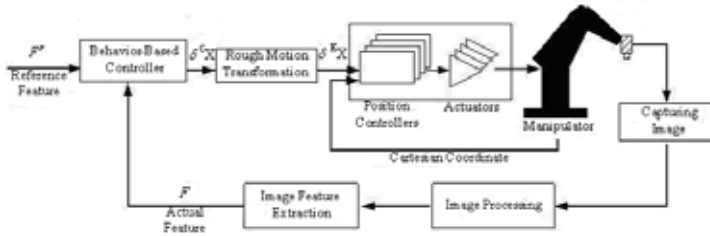


Fig. 9. Behavior-based look-and-move control structure.

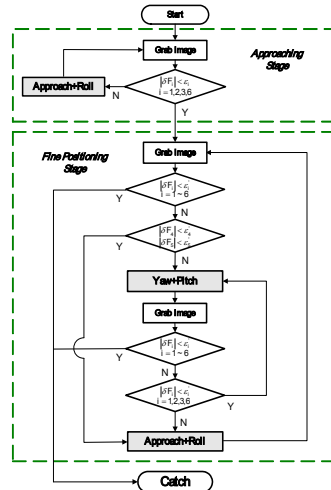


Fig. 10. Control strategy flowchart for manipulation.

4. Experimentation

4.1 Experimental Setup

As shown in Fig. 7, a CCD camera is mounted on the end-effector of the manipulator in such a way that each coordinate axis in the camera frame is approximately parallel to that in the end-effector frame. A ruler and the naked eyes reveal that the distance between ${}^c X$ and ${}^e X$, dx , is about 73 mm. dz is also estimated to be around 80 mm. Therefore, $dk = \sqrt{(dx)^2 + (dz)^2}$ is approximately 107 mm. The end-effector employed to grasp the workpiece is a two-finger gripper with parallel motion of the fingers, and is attached to the end of the manipulator. The gripper's fingers are installed approximately parallel to the ${}^e X$ axis of the end-effector frame. The gripper is always open, except when the end-effector is commanded to grasp the workpiece from the side of the workpiece. The manipulator's end-effector will pick up a workpiece that is a rectangular parallelepiped whose volume is $68\text{mm} \times 30\text{mm} \times 55\text{mm}$. The zoom of the camera is adjusted so that

horizontal angle of view is around 39° . However, gripper's fingers are still not in the field of vision. An auto-focus function is exploited to maintain the sharpness of the workpiece image.

The mobile base is stopped and fixed next to the workstation, which is roughly parallel to the surface of the workstation to verify the proposed control strategy. As presented in Fig. 11, the workpiece is placed in six different positions, which are separated by 15 cm, to simulate the possible position and orientation errors that arise in the application stage. In each position, the workpiece is pointed in three directions. It is placed at Pos2 and Pos5 in 0° and -45° directions, and is tilted by 3° and 6° to the station surface to simulate non-flat ground in the application stage. Before the manipulation is performed, one reference image is captured by applying the teach-by-showing method. The end-effector is first driven by a teaching box to a location so that the gripper can grasp the workpiece. Then, the end-effector is driven 10 cm above by a teaching box along the negative- Z axis to the target location. Notice that the target location of the end-effector in reference with the object frame is always fixed and is expected to reach. Then, the corresponding reference image features are extracted. Given the uncalibrated CCD camera and the imprecisely known hand-eye configuration, (F_1^r, F_2^r) does not necessarily correspond to the center pixel of the image plane, and F_6^r does not always equal zero degree. All of these three feature values depend on the captured reference image. However, the reference image feature F_3^r , which presents the distance between the camera and the workpiece, is set to zero at all times. The other two feature values, F_4^r and F_5^r , depend on the captured reference image.

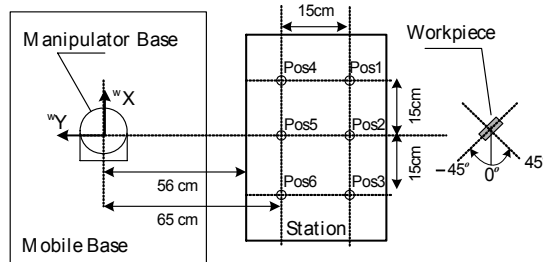


Fig. 11. Possible locations of the workpiece to be picked up.

The parameters of the above experimental setup to evaluate the positioning performance of the eye-in-hand manipulator are set as follows. Table 1 lists the initial parameters of the fuzzy membership functions, including the mean-points and the standard deviations of the Gaussian curves, and the learning rate of each DOF control in the designed neural fuzzy controllers. The developed control strategy is divided into two stages. The approaching stage continues until the errors in the image features δF_1 , δF_2 , δF_3 and δF_6 are below $\varepsilon_1 = 6$, $\varepsilon_2 = 6$, $\varepsilon_3 = 0.1$ and $\varepsilon_6 = 2$, respectively. In neural fuzzy controllers, the corresponding energy values E_1 , E_2 , E_3 and E_6 are less than 18, 18, 0.005 and 2, respectively. As to the following fine positioning stage, it never terminates unless the errors in the image features δF_1 , δF_2 , δF_3 , δF_4 , δF_5 and δF_6 are less than the specified values $\varepsilon_1 = 2$, $\varepsilon_2 = 2$, $\varepsilon_3 = 0.01$, $\varepsilon_4 = 0.01$, $\varepsilon_5 = 0.01$ and $\varepsilon_6 = 0$. The corresponding energy values $E_1 \sim E_6$ are below 2, 2, 0.00005, 0.00005, 0.00005 and 0, respectively.

Rule	Rule Parameter	NB (j=1)	NM (j=2)	NS (j=3)	ZO (j=4)	PS (j=5)	PM (j=6)	PB (j=7)	η_i
R_1^j (i=1)	C_1^j pixel	-90	-60	-30	0	30	60	90	0.0005
	σ_1^j pixel	11.339	11.339	11.339	11.339	11.339	11.339	11.339	
	W_1^j mm	-50	-10	-3	0	3	10	50	
R_2^j (i=2)	C_2^j pixel	-90	-60	-30	0	30	60	90	0.0005
	σ_2^j pixel	11.339	11.339	11.339	11.339	11.339	11.339	11.339	
	W_2^j mm	-50	-10	-3	0	3	10	50	
R_3^j (i=3)	C_3^j	-0.75	-0.5	-0.25	0	0.25	0.5	0.75	500
	W_3^j	0.0945	0.0945	0.0945	0.0945	0.0945	0.0945	0.0945	
	W_3^j mm	-150	-80	-25	0	25	80	150	
R_4^j (i=4)	C_4^j	-0.0276	-0.0207	-0.0138	0	0.0138	0.0207	0.0276	0.05
	σ_4^j	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	
	W_4^j degree	-1.5	-1	-0.5	0	0.5	1	1.5	
R_5^j (i=5)	C_5^j	-0.0081	-0.0054	-0.0027	0	0.0027	0.0054	0.0081	0.05
	σ_5^j	0.001	0.001	0.001	0.001	0.001	0.001	0.001	
	W_5^j degree	-1.5	-1	-0.5	0	0.5	1	1.5	
R_6^j (i=6)	C_6^j degree	-90	-60	-30	0	30	60	90	0.01
	σ_6^j degree	11.339	11.339	11.339	11.339	11.339	11.339	11.339	
	W_6^j degree	-90	-60	-30	0	30	60	90	

Table 1. Initial parameters of neural fuzzy controllers.

4.2 Results

In experiments performed to evaluate the positioning performance of the eye-in-hand manipulator, the end-effector of the manipulator is firstly driven to the top location, presented as ${}^w[-80, -750, 470, 180, 0, 90]$ in the world frame. Then, the end-effector is visually guided to grasp the workpiece according to the proposed control strategy with the preset parameters. Fig. 12 displays the images in the course of approaching the workpiece.

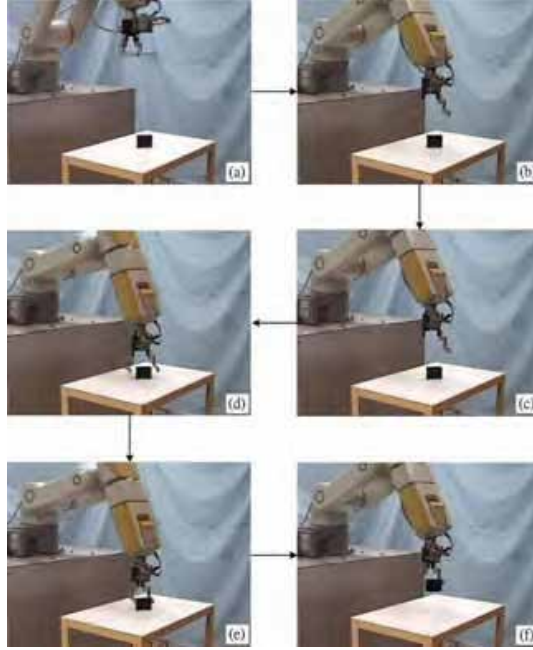


Fig. 12. (a)~(f) Images in the course of approaching the workpiece, based on the proposed control strategy; (a)~(b) approaching stage; (b)~(c) fine positioning stage; (d)~(f) catch operation.

The position and orientation errors are defined below to describe the positioning performance of the manipulator numerically. The location of the end-effector in each step, T_o^{eStep} , is recorded during the task manipulation. The coordinate transformation matrix between each step and desired locations of the end-effector can be written as follows.

$$T_{eDesired}^{eStep} = T_o^{eStep} (T_o^{eDesired})^{-1} = \begin{bmatrix} R_{eDesired}^{eStep} & t_{eDesired}^{eStep} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow T_{eDesired}^{eStep} = \begin{bmatrix} n_x & t_x & b_x & X \\ n_y & t_y & b_y & Y \\ n_z & t_z & b_z & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

where T_o^{eStep} corresponds to the location of the end-effector with respect to the world frame in each step, and $T_o^{eDesired}$ corresponds to the desired location for the end-effector with respect to the world frame. In each test, (10) gives the position error and (11) specifies the orientation error as the angle of rotation between the two coordinate frames about the principle axis.

$$Error_{position} = \sqrt{X^2 + Y^2 + Z^2} \quad (10)$$

$$\text{Error}_{\text{orientation}} = \tan^{-1} \left(\frac{\sqrt{(t_z - b_y)^2 + (b_x - n_z)^2 + (n_y - t_x)^2}}{n_x + t_y + b_z - 1} \right) \quad (11)$$

Table 2 presents the resulting positioning errors and the number of execution steps in each test. The number of steps is that required for the end-effector to travel from the top location to the target location. Executing this behavior-based control strategy takes a minimum of around 11 steps and a maximum of 17 steps in cases in which the workpiece is arranged without a tilt. The initial orientation of the workpiece only weakly influences the total execution time. In the cases in which the tilt angle is 3° or 6°, about 22 steps to 35 steps are required. In all of the tests, the final position error is less than 2.9 mm, and the final orientation error is less than 1.5°. Additionally, as the tilt angles are increased, more steps are needed for positioning the end-effector.

Workpiece Locations	Error _{Position} (mm)	Error _{Orientation} (degree)	Number of Steps
Pos1 No Rotation	1.0531	0.1384	14
Pos1 Rotation 45°	0.7573	0.1909	16
Pos1 Rotation -45°	1.7142	0.2180	11
Pos2 No Rotation	1.0337	0.1160	13
Pos2 Rotation 45°	0.5197	0.0767	15
Pos2 Rotation -45°	1.1542	0.0956	13
Pos3 No Rotation	1.2159	0.1732	16
Pos3 Rotation 45°	0.9185	0.1447	12
Pos3 Rotation -45°	1.0466	0.5370	12
Pos4 No Rotation	0.7296	0.0600	16
Pos4 Rotation 45°	1.4728	0.3718	14
Pos4 Rotation -45°	2.8471	0.5503	13
Pos5 No Rotation	0.6244	0.0591	16
Pos5 Rotation 45°	0.8570	1.1380	14
Pos5 Rotation -45°	1.5124	0.1721	16
Pos6 No Rotation	0.7469	0.0516	16
Pos6 Rotation 45°	1.4360	0.4218	13
Pos6 Rotation -45°	0.8015	0.0649	17
Pos2 No Rotation Tilted 3°	1.6559	1.1362	22
Pos2 No Rotation Tilted 6°	1.5903	0.3740	32
Pos2 Rotation -45° Tilted 3°	0.2003	0.1174	25
Pos2 Rotation -45° Tilted 6°	1.8498	0.3057	35
Pos5 No Rotation Tilted 3°	2.4739	0.4196	24
Pos5 No Rotation Tilted 6°	2.4876	0.3764	29
Pos5 Rotation -45° Tilted 3°	2.4000	1.4623	24
Pos5 Rotation -45° Tilted 6°	2.2709	0.3534	31

Table 2. Positioning errors of the eye-in-hand manipulator and the number of steps in each test.

4.3 Discussion

Although these experiments were performed with the mobile base fixed on the ground such that the base of the manipulator was approximately parallel to the surface of station, the workpiece was placed in six positions with various orientations and two tilt angles. These experimental data simulate the possible locations of the workpiece in relation to the base of the manipulator, when the mobile manipulator is driven to the station using a guidance control system of the mobile base in the application stage. Notably, a workpiece placed in various positions is used to simulate the position errors of the mobile base in the traveling and lateral directions. The workpiece points in different directions to simulate the orientation errors of the mobile base, and the setups with the different tilt angles of the workpiece are used to simulate the non-horizontality of the ground.

In the application stage, if the workpiece is placed in the middle position between Pos2 and Pos5 and pointed in the 0° direction, then the manipulator base is supposed to be driven by the mobile base to stop next to it. The experimental results confirm that the eye-in-hand manipulator can still pick up the workpiece, even when the position errors of the mobile base in the direction of travel and in the lateral direction are $\pm 150\text{mm}$ and $\pm 75\text{mm}$, respectively. Moreover, the manipulator can tolerate a workpiece rotation of $\pm 45^\circ$ and a non-horizontality angle of the ground of 6° .

In the other experiments, the workpiece is shifted by manually pushing the workstation in the approaching stage to test the advantages of the reactive arbitration. Nevertheless, the end-effector can correct its path to reach the desired pose without a loss, and pick up the workpiece. Furthermore, in all of the experimental tests, the workpiece was picked up successfully. Therefore, the probability of successful grasping was about 100%.

5. Conclusion

Mobile robots commonly need a guidance control system to navigate the mobile base and a method for performing the pick-and-place operations. The guidance control system and the non-planar ground inevitably cause the position and orientation errors of the mobile base. Therefore, this study proposes a behavior-based control strategy that employs an uncalibrated eye-in-hand vision system to control the end-effector of the manipulator to approach and grasp the target workpiece. All the designed behaviors are defined from the perspective of the camera and are mediated through fuzzy rules with a look-and-move control structure. The presented neural fuzzy controllers map image features in image space to relative motion commands in the camera frame. These motion commands are then transformed to the end-effector frame by the proposed rough motion transformation.

This work differs from the references (Wasik & Saffiotti, 2002, 2003) as follows. (1) A back-propagation algorithm is used to reduce image feature errors through the adjustment of the singletons of the consequent parts in the fuzzy singleton rules. (2) Two additional image features, the ratios of the lengths of the two pairs of opposite sides on the quadrangular image, are extracted to guide the rotation of the camera when the workpiece has a tilt angle. (3) The control values defined in the camera frame generated from the controller in (Wasik & Saffiotti, 2003) are transformed to joint controls by a Jacobain transformation, revealing that either a hand-eye calibration process has been implemented or the hand-eye relationship is known beforehand. However, this study proposes a rough motion transformation to replace the time-consuming hand-eye calibration. Only a general ruler and naked eye are required to estimate hand-eye configuration. The inaccuracy of this estimation can be ignored by adjusting on-line the singletons of the consequent parts in the fuzzy singleton rules. (4) A more human-like control strategy is

planned. The end-effector approaches the workpiece and the image is simultaneously maintained in the reference pixel of the image plane. The path from the initial location of the end-effector to the location where the workpiece can be grasped is shortened.

6. References

- Anglani, A.; Taurisano, F.; De Giuseppe, R. & Distante, C. (1999). Learning to Grasp by using Visual Information, *Proceedings of IEEE International Conference on Computational Intelligence in Robotics and Automation*, pp. 7-14, ISBN: 0-7803-5806-6, Monterey, CA, USA, Nov. 1999.
- Bien, Z. & Park, J.. (1993). Hybrid Fuzzy Self-organizing Controller for Visual Tracking, In: *Fuzzy Logic: State of the Art*, Lowen, R & Roubens, M., (Ed.), page numbers (569-578), Kluwer Academic Publishers, ISBN: 0-7923-2324-6, Netherlands.
- Hutchinson, S.; Hager, D. G. & Corke, P. I. (1996). A Tutorial on Visual Servo Control. *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 5, (Oct. 1996) page numbers (651-670), ISSN: 1042-296X.
- Kim, C. S.; Seo, W. H.; Han, S. H. & Khatib, O. (2001). Fuzzy Logic Control of a Robot Manipulator Based on Visual Servoing, *Proceedings of IEEE International Symposium on Industrial Electronics*, pp. 1597-1602, ISBN: 0-7803-7090-2, Pusan, Korea, June 2001.
- Kim, J. G.; Cha, D. H.; Cho, K. S. & Kim, S. H. (1995). An Auto Tuning Fuzzy Rule-Based Visual Servoing Algorithm for a Slave Arm, *Proceedings of IEEE International Symposium on Intelligent Control*, pp. 177-182, ISBN: 0-7803-2722-5, Monterey, CA, USA, Aug. 1995.
- Lin, C. T. & Lee, C. S. G. (1996). *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*, Prentice-Hall, Inc., ISBN: 0-13-261413-8, Upper Saddle River, NJ, U.S.A.
- Nomura, H.; Hayashi, I. & Wakami, N. (1992). A Learning Method of Fuzzy Inference Rules by Descent Method, *Proceedings of IEEE International Conference on Fuzzy Systems*, pp. 203-210, ISBN: 0-7803-0236-2, San Diego, CA, USA, Mar. 1992.
- Suh, I. H. & Kim, T. W. (1994). Fuzzy Membership Function Based Neural Networks with Applications to the Visual Servoing of robot Manipulators. *IEEE Transactions on Fuzzy Systems*, Vol. 2, No. 3, (Aug. 1994) page numbers (203-220), ISSN: 1063-6706.
- Ting, Y.; Chen, Y. H.; Lin, M.; Dai, S. C. & Kang, Y. (1997). A Study on the Inaccuracy of Vision System of Mobile Robots Causing the Failure of Pick-and-place Tasks, *Proceedings of 1997 Florida Conference on Recent Advances in Robotics*, pp. 43-46, Miami, FL, USA, Apr. 1997.
- Wasik, Z. & Saffiotti, A. (2002). A Fuzzy Behavior-Based Control System for Manipulation, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1596-1601, ISBN: 0-7803-7398-7, EPFL, Switzerland, Oct. 2002.
- Wasik, Z. & Saffiotti, A. (2003). A Hierarchical Behavior-Based Approach to Manipulation Tasks, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2780-2785, ISBN: 0-7803-7736-2, Taipei, Taiwan, Sept. 2003.

A Fuzzy Logic Controller for Autonomous Wheeled Vehicles

Mohamed B. Trabia

University of Nevada, Las Vegas
U.S.A.

Linda Z. Shi

University of California, San Diego
U.S.A.

Neil E. Hodge

University of California, Berkeley
U.S.A.

1. Introduction

Autonomous vehicles have potential applications in many fields, such as replacing humans in hazardous environments, conducting military missions, and performing routine tasks for industry. Driving ground vehicles is an area where human performance has proven to be reliable. Drivers typically respond quickly to sudden changes in their environment. While other control techniques may be used to control a vehicle, fuzzy logic has certain advantages in this area; one of them is its ability to incorporate human knowledge and experience, via language, into relationships among the given quantities. Fuzzy logic controllers for autonomous vehicles have been successfully applied to address various (and sometimes simultaneous) navigational issues, including:

- reaching a static target (Baturone, et al., 2004, Boada, et al., 2005, El Hajjaji & Bentalba, 2003, Maeda et al., 1991, Chen & Ozguner, 2005),
- tracking moving targets (Ollero, et al., 2001),
- maintaining stable vehicular velocity (Holzmann et al., 1998, Nobe & Wang, 2001),
- following a lane or a wall (Rosa & Garcia-Alegre, 1990, Hessburg & Tomizuka, 1994, Peng & Tomizuka, 1993) and,
- avoiding collision with static and dynamic obstacles (Baturone, et al., 2004, Murphy, 2001, Godjevac, et al., 2001, Seraji, 2005, Lee & Wang, 1994, Wheeler & Shoureshi, 1994, Ye & Wang, 2001).

Several researchers combined fuzzy logic controllers with various learning techniques, such as:

- supervised learning method (Godjevac, 2001),
- evolutionary method (Hoffman, 2001, Kim, et al., 2001),
- neural network (Pasquier, et al., 2001, Cang, et al., 2003),
- reinforcement learning (Dai, et al., 2005) and,
- optimization methods (Hong, 1997, Sanchez, et al., 1999).

To accomplish most tasks of a mobile robot, the controller must be able to adjust the steering angle and the velocity of the vehicle simultaneously. Most of the presented work below considers one or two aspects of the problem. For example, (Rosa & Garcia-Alegre, 1990) proposed a fuzzy logic controller that can modify the speed and the steering of a mobile robot to steer it at a fixed distance along a wall. Using a kinematic model and a variety of wall shapes, including cubic splines and line segments with ninety-degree intersections, they designed a fuzzy logic controller with a minimum number of rules to accomplish the task. (Maeda, et al., 1991) implemented fuzzy logic for the steering and the speed control of a two-wheel drive robot. The environment in which the control scheme was tested was limited to straight-line paths with a minimum number of ninety-degree turns. Fuzzy logic was incorporated in lateral vehicle control to solve the lane following problem (Hessburg & Tomizuka, 1994). They tested their design on a dynamic model first, and then implemented it in an actual vehicle to evaluate it. The experiment was to drive the vehicle on a road with multiple smooth curves interspersed by straight segments. (Peng & Tomizuka, 1993) described an engine throttle fuzzy logic controller for accelerating and decelerating the vehicle during typical driving conditions. The proposed controller had no provisions for obstacle avoidance. (Lee & Wang, 1994) proposed the use of fuzzy logic to assist in obstacle avoidance. Their simulation included both static and moving obstacles. (Wheeler & Shoureshi, 1994) controlled a vehicle on handling track using fuzzy logic. Their model included the vehicle's dynamic behavior. The handling track consisted of a set of pylons (known static obstacles) that the vehicle must navigate at high speeds (i.e., around 50 miles per hour). (Holzmann et al., 1998) combined a conventional vehicle acceleration controller with a fuzzy logic controller for vehicle velocity and inter-vehicle distance. (Sanchez et al., 1999) presented an off-line two-level fuzzy logic controller to track a path previously recorded or computed by means of a path planning program. The number of fuzzy rules was optimized to improve performance. (Ye & Wang, 2001) presented a novel navigation method for an autonomous vehicle in unknown environments. Their navigator consisted of an Obstacle Avoider (OA), a Goal Seeker (GS), a Navigation Supervisor (NS) and an Environment Evaluator (EE). The fuzzy actions inferred by the OA and the GS were weighted by the NS using the local and global environmental information and fused through fuzzy set operation to produce a command action. (Nobe and Wang, 2001) reviewed recent developments in advanced vehicle control systems (AVCS) including lateral steering, longitudinal throttle control, and integration of these controls for vehicles. Autonomous intelligent cruise control (AICC) and cooperative intelligent cruise control (CICC) were considered for a platoon of two or more closely spaced vehicles travelling with same velocity in a same lane. Look-down and look-ahead systems for automated steering were presented for the lateral vehicle control. (Kodagoda et al., 2002) developed and implemented fuzzy proportional derivative-proportional integral (PD-PI) controllers for the steering and the speed control of an autonomous guided vehicle. (Ohara & Murakami, 2004) proposed a model for a compact tractor-trailer using an electrical vehicle that estimated the cornering force and friction and changed speed and steering angle based on the inputs. The proposed PD controller can avoid the jackknife phenomenon of the vehicle. (Chen and Ozguner, 2005) presented a real-time fuzzy navigation algorithm for off-road autonomous ground vehicles. The controller's goal was to direct the vehicle safely, continuously and smoothly across natural terrain to reach a goal. The proposed navigator consisted of two fuzzy controllers, the steering

controller and the speed controller. These two collaborative controllers were designed separately by mimicking human performance.

The objective of this chapter is to describe a fuzzy logic controller for the steering and the velocity of an automobile. Observation indicates that drivers tend to separate the various driving tasks. For example, most drivers conceptualize velocity and direction separately. This separation of objectives is the basis of the proposed distributed fuzzy logic controller for the vehicle. A controller for the steering of an autonomous vehicle needs to achieve these objectives:

- steering the vehicle toward the target,
- steering the vehicle around any obstacle to avoid a collision,
- avoid being trapped in maze or cluttered environment,
- steering the vehicle to stop at a desired orientation.

The velocity controller should emulate the following human behaviors:

- starting the vehicle from a complete stop, and stopping it when it reaches the target,
- slowing down the vehicle when it approaches an obstacle and speeding it up as it moves beyond the obstacle,
- slowing down the vehicle when its turning radius decreases (i.e., the tighter the turn, the lower the velocity).

A fuzzy logic controller module, of the Mamdani-type, is created for each of these objectives.

The following is a brief summary for the remainder of this chapter. The second section briefly describes the nonlinear model of the vehicle dynamics. The third section details the first two modules of the steering controller, which meet the fundamental driving requirements: the Target Steering Fuzzy module, and the Collision Avoidance Steering Fuzzy module. The fourth section describes the third and the fourth modules of the steering controller, which deal with some special driving configurations and requirements: the Modified Bug Steering Fuzzy module, and the Final Orientation module. The fifth section introduces the three modules of the velocity controller: the Target Throttle Fuzzy module, the Cornering Throttle Fuzzy module, and the Collision Avoidance Throttle Fuzzy module. The sixth section proposes a tuning of the Target Throttle Fuzzy module to maintain a smooth velocity profile when the vehicle reaches its target position. The seventh section depicts two examples to show the operation of the two proposed fuzzy controllers, and comparison with the results of (Lee and Wang, 1994) are also included. The final section presents conclusions and recommendations for further work.

2. Vehicle Model

Since an automobile is a complex dynamic system, the details of its model are crucial to the accuracy of the simulation. The principal work in this field is (Wong, 1993) who developed the equations of motion for various types of vehicles. The model that is used in this paper depicts a two-axle, four-wheel vehicle, which is commonly known as the “bicycle” model since it uses only two wheels to represent a four-wheel vehicle; it neglects the lateral variations in the tire-road interface forces. It is further modified according to the suggestions of (Wheeler & Shoureshi, 1994) whose vehicle model was made more realistic by limiting the total tire force vector. They also derived expressions for calculating the longitudinal braking and accelerating forces. Figure 1 shows free body diagram of the vehicle. The equations of motion are

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \begin{Bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{Bmatrix} + \begin{Bmatrix} -m\dot{y}\dot{\theta} \\ m\dot{x}\dot{\theta} \\ 0 \end{Bmatrix} = \begin{Bmatrix} F_{xf} \cos \delta_f + F_{xr} - F_{yf} \sin \delta_f \\ F_{yr} + F_{yf} \cos \delta_f + F_{xf} \sin \delta_f \\ L_1 F_{yf} \cos \delta_f - L_2 F_{yr} + L_1 F_{xf} \sin \delta_f \end{Bmatrix} \quad (1)$$

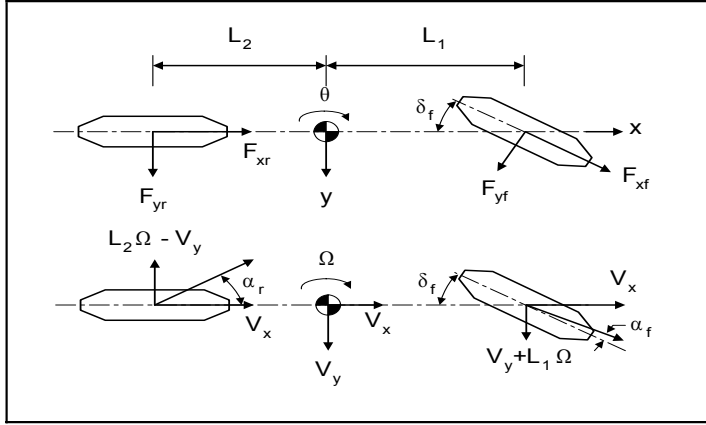


Fig. 1. Free Body Diagram of Vehicle Model.

The tires, which are modeled as nonlinear springs, are described as

$$\bar{F}_f = \frac{\tilde{F}_f}{\|\tilde{F}_f\|} \min(\|\tilde{F}_f\|, F_{f \max}) \quad (2)$$

$$\bar{F}_r = \frac{\tilde{F}_r}{\|\tilde{F}_r\|} \min(\|\tilde{F}_r\|, F_{r \max}) \quad (3)$$

The above forces can be resolved into x and y components to describe the motion. The lateral tire forces, F_{yf} and F_{yr} , are functions of each tire slip angle α and the cornering stiffness C_{α} . The lateral tire forces can be calculated using the following expressions:

$$\tilde{F}_{yf} = 2C_{\alpha f} \left(\delta_f - \tan^{-1} \left(\frac{L_1 \dot{\theta} + \dot{y}}{\dot{x}} \right) \right) \quad (4)$$

$$\tilde{F}_{yr} = 2C_{\alpha r} \left(\tan^{-1} \left(\frac{L_2 \dot{\theta} - \dot{y}}{\dot{x}} \right) \right) \quad (5)$$

The axial tire forces, F_{xf} and F_{xr} , are dependent on the angle of the gas pedal, δ_{gb} . This model uses the convention that a positive gas pedal angle represents the driver pushing on the gas pedal, and a negative gas pedal angle represents the driver pressing on the brake pedal. As such, there are two sets of axial tire force equations. For $\delta_{gb} < 0$,

$$\tilde{F}_{xf} = 0.7K_b \delta_{gb} \quad (6)$$

$$\tilde{F}_{xr} = 0.3K_b\delta_{gb}. \quad (7)$$

The power train is assumed to correspond to a rear wheel drive. Thus, for $\delta_{gb} > 0$,

$$\tilde{F}_{xf} = 0, \quad (8)$$

$$\tau_g \tilde{F}_{xr} = K_g \delta_{gb} - \tilde{F}_{xr} \quad (9)$$

Many of these parameters, which are particular to each vehicle, are usually determined experimentally. For this model, the parameters for a typical sports utility vehicle are used. Some of these parameters were found in (Byrne & Abdallah, 1995). The parameters K_g , K_b , and τ_g , are determined by varying their values and comparing the performance of the model during starting from rest, various peak velocities, and braking with experimental data.

It is assumed that the target has a beacon to guide the vehicle. It is also assumed that the vehicle is equipped with a proximity sensing system to determine the distance and direction of obstacles. This sensing system may be a single sensor mounted on a rotating platform on the front of the vehicle or a battery of sensors arrayed around the front of the vehicle and pointed along regular intervals. The configuration of the buffer zone created by the sensors is shown in Figure 2. To simplify obstacle detection, the sensor outputs the minimum measured distance to the nearest obstacle d_o and the direction of d_o , $\Delta\phi_o$. While d_o is not guaranteed to be the minimum distance to obstacle, sampling many points at a high frequency will increase the accuracy of the proposed technique.

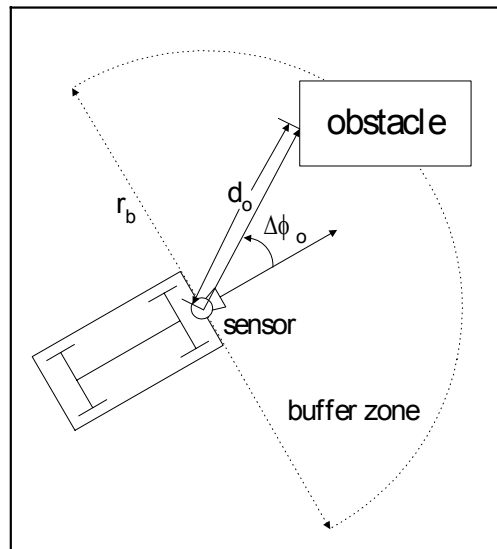


Fig. 2. Sensing Field Configuration.

3. Basic Steering Fuzzy Controller

In an effort to incorporate human knowledge and experience most efficiently in the design of the controller, the driving is divided into several tasks and a fuzzy controller was designed for each task. The basic driving tasks are to drive the vehicle toward the target and to avoid collision with obstacles. Two fuzzy modules, Target Steering Fuzzy module and Collision Avoidance Steering Fuzzy module, are designed to fulfill those two tasks. Two additional modules are discussed in the next section that meet special configurations or requirements.

Figure 3 shows a schematic of the inputs and outputs of Target Steering Fuzzy module and Collision Avoidance Steering Fuzzy module. The total steering angle is a summation of the outputs of two fuzzy modules. When the vehicle is near an obstacle, the output of Collision Avoidance Steering Fuzzy module is given a higher weight than that of Target Steering Fuzzy module, so that Collision Avoidance Steering Fuzzy module will be able to significantly affect the behavior of the vehicle in a short period of time.

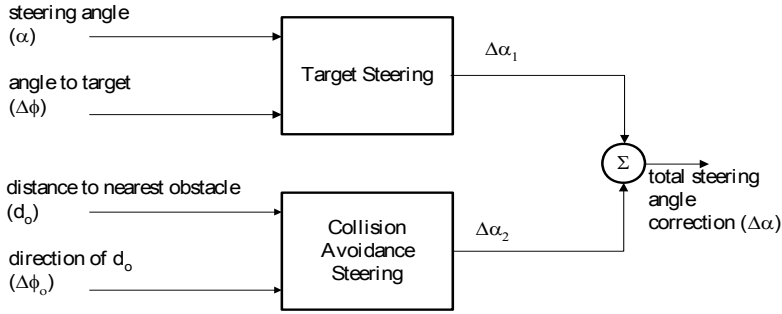


Fig. 3. Target Steering Fuzzy and Collision Avoidance Steering Fuzzy modules.

The membership functions in all of the fuzzy modules in this paper are either *sigmoid* or *product of sigmoid* types. The *sigmoid* membership function is defined as

$$f(x) = \frac{1}{1 + e^{-a(x-c)}}. \quad (10)$$

The *product of sigmoid* membership function is the result of multiplying two sigmoid membership functions together,

$$\mu(x) = \frac{1}{1 + e^{-a_L(x-c_L)}} \frac{1}{1 + e^{a_R(x-c_R)}} \quad (11)$$

Membership functions and the rules of the controller's modules that are used throughout the paper are based on common sense and observation of drivers' behavior and description of the variables used. These functions and rules are the result of tuning by running the simulation through various environments and by altering the initial vehicle position and orientation, the target vehicle position and orientation, the number and configuration of static obstacles, and the path and speed of dynamic obstacles.

3.1 Target Steering Fuzzy Module

The objective of this module is to steer the vehicle toward a target from its current location. The inputs to this module are the steering angle α and the angle to the target $\Delta\phi$, Figure 4. The output of this module is the change of steering angle $\Delta\alpha$. This module has the following goals:

- i. If the target is on the right side of the vehicle, it should turn to this direction. The bigger the angle to the target, the sharper the change of steering angle should be.
- ii. The change of steering angle depends on the current value of the steering angle.

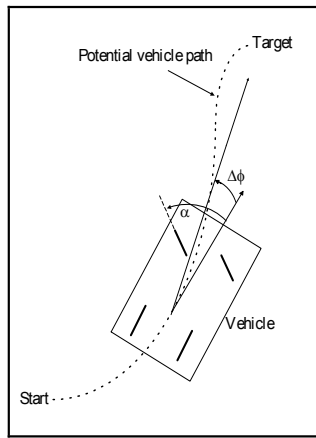


Fig. 4. Input Variables of the Target Steering Module.

Five membership functions: Negative Big (NB), Negative Medium (NM), Zero (Z), Positive Medium (PM) and Positive Big (PB), as shown in Figures 5 and 6, are used to describe the inputs and output of this fuzzy module. The same membership functions were used for α and $\Delta\phi$ for simplicity. The inner bounds of the PM and NM are all at zero due to the gradual tuning that was performed on them. This tuning also accounts for the lack of a “small” membership set, which was eliminated during the fine tuning phase.

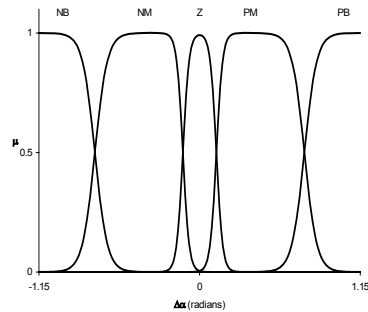
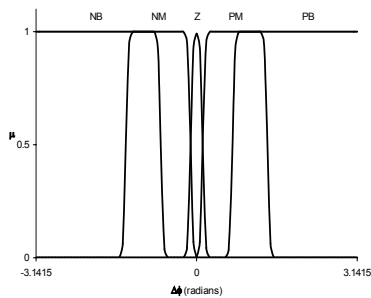


Fig. 5. Membership Functions for α and $\Delta\phi$ Fig. 6. Membership Functions for $\Delta\alpha$.

The rationale behind several of the rules of this module is presented here. Larger correction is usually not used to allow for the time delay in the system.

- If (α is Z) and ($\Delta\phi$ is NB) then ($\Delta\alpha_1$ is NM): If the vehicle is moving along a straight path and the target is to the right of the vehicle, then the correction to the steering angle should be of medium magnitude and to the right.
- If (α is PB) and ($\Delta\phi$ is Z) then ($\Delta\alpha_1$ is NM): If the vehicle is turning to the left and the target is straight ahead, then the correction to the steering angle should be of medium magnitude and to the right.
- If (α is Z) and ($\Delta\phi$ is Z) then ($\Delta\alpha_1$ is Z): If the current steering angle is zero and the target is straight in front of vehicle, then no correction to the steering angle is necessary.

The full rule base of this module is given in Table 1.

$\alpha \Rightarrow$ $\Delta\phi \Downarrow$	NB	NM	Z	PM	PB
NB	Z	Z	NM	NB	NB
NM	Z	Z	NM	NM	NM
Z	PM	Z	Z	Z	NM
PM	PB	PB	PM	Z	Z
PB	PB	PB	PM	Z	Z

Table 1. Rules for Target Steering Fuzzy Module

3.2 Collision Avoidance Steering Fuzzy Module

The objective of this module is to steer the vehicle away from obstacles, both static and dynamic. It is assumed that the vehicle is equipped with a sensing system that can determine the distance and direction of obstacles. The sensing system may be a single sensor mounted on a rotating platform on the front of the vehicle or may be a battery of sensors arrayed around the vehicle's front section along regular intervals. The buffer zone radius will be denoted by r_b , and is assigned a value of twenty meters. The configuration of the buffer zone is shown in Figure 2. Sensors input only two bits of information to the controller to simplify the analysis. The inputs are the distance and angle to the nearest obstacle, d_0 and $\Delta\phi_0$, respectively. As Figure 2 illustrates, the sensor may not return information about the closest point and the corresponding angle since the proposed sensing system used is not continuous. The sensor takes n_s samples over its 180-degree sweep. The output of the module is the steering correction $\Delta\alpha_2$. Since the locations of the obstacles are not known before the vehicle starts traversing the environment, the module uses a right-hand turning rule whenever it confronts an obstacle. The two basic goals of this module are as follows:

- The closer the vehicle is to the obstacle, the more extreme the evasive maneuver is, i.e., the larger the steering correction.
- The direction of the obstacle with respect to the front of the vehicle determines the magnitude and direction of the steering correction.

Three membership functions, Big (B), Small (S) and Zero (Z) are used to describe the first input variable, d_0 as shown in Figure 7. Five membership functions, Negative Medium (NM), Negative Small (NS), Zero (Z), Positive Small (PS) and Positive Medium (PM) are used to describe the second input variable, $\Delta\phi_0$ and the output variable, $\Delta\alpha_2$ as shown in Figure 8. The membership functions of $\Delta\alpha_2$ in this module have the same shape and relative

size as those of $\Delta\alpha_1$ in the previous module, Figure 9. The only difference is that all of the membership functions have been scaled down.

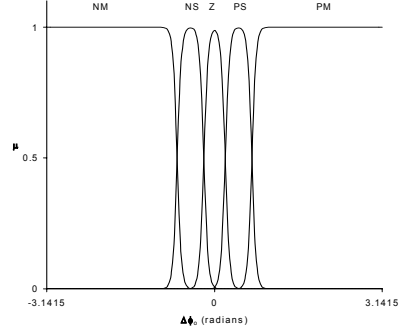
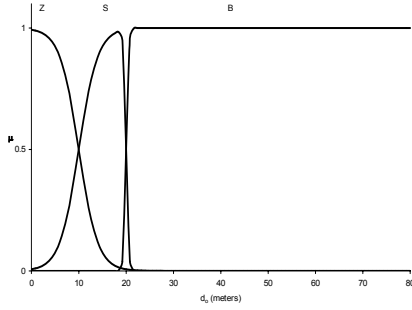


Fig. 7. Membership Functions for d_0 .

Fig. 8. Membership Functions for $\Delta\phi_0$.

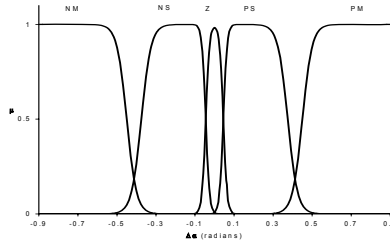


Fig. 9. Membership Functions for Output Variable $\Delta\alpha_2$.

At this point, it would be useful to look at a few of the rules to observe the relation between the rules and the goals of this fuzzy module:

- If (d_0 is S) and ($\Delta\phi_0$ is PS), then ($\Delta\alpha_2$ is NM): If the obstacle is between ten and twenty meters away, and is between zero to sixty degrees off to the left of the longitudinal axis of the vehicle, a sharp steering correction to the right will be needed.
- If (d_0 is B) then ($\Delta\alpha_2$ is Z): If the obstacle is more than twenty meters away from the vehicle, then no steering correction is made, regardless of its direction.

The rules of this module are shown in Table 2.

$d_0 \Rightarrow$ $\Delta\phi_0 \Downarrow$	Z	S	B
NM	PM	PM	Z
NS	PM	PS	Z
Z	NM	NM	Z
PS	NM	NM	Z
PM	NM	NM	Z

Table 2. Rules for Collision Avoidance Steering Fuzzy Module.

4. Extended Steering Fuzzy Controller

The basic steering fuzzy controller described in the previous section works well in most driving conditions. However, it may get stuck in a loop and never reach the target under certain special configurations. It may also fail to determine that the target is unreachable. In some applications, the vehicle is required to park at the target position with a specific orientation. Thus, two modules are added in the steering fuzzy controller: Modified Bug Steering Fuzzy module and Final Orientation module. The first module steers the vehicle within a maze and the other helps steer the vehicle toward its target position at a desired orientation.

4.1 Modified Bug Steering Fuzzy Module

As a backup to the primary steering fuzzy controller, which usually regulates obstacle avoidance steering, the Bug module is implemented. This module is not designed to emulate human behavior, but rather to reach the target when the steering fuzzy module is not able to. The Bug module is based on the Bug2 algorithm proposed by (Lumelsky & Stepanov ,1987, Lumelsky & Skewis, 1988, Lumelsky, 1991) whose research focused on the application of maze theory to the path planning of a “point automaton”. His research concluded that Bug2 had “unbounded worst-case performance”, i.e. in very rare cases, Bug2 can still drive the vehicle in an unending loop. (Sauerberger & Trabia, 1996) proposed a modified form of this algorithm for autonomous omnidirectional vehicles. Their algorithm, which produces shorter paths in many cases, triggered the Bug algorithm when the orientation of the vehicle was more than 360 degrees away, in either direction, from the angle of the ST line, Figure 10. This can be expressed as the following:

$$|\theta_v - \theta_{ST}| > 2\pi \quad (12)$$

For example, if the ST line was at 45 degrees, the vehicle would have to traverse a closed loop, and have an orientation of 405 degrees, before the Bug2 algorithm was activated.

The controller presented here presents another modification of the Bug2 algorithm. The modified algorithm is activated once the trigger condition is met. At this stage, the algorithm performs the following tasks:

- 1) Initially, the algorithm guides the vehicle straight toward the target. If the finishing criteria (discussed below) are satisfied, the vehicle turns off the Bug2 algorithm and goes straight to the target. If the vehicle gets to within r_o meters of an obstacle, the vehicle will turn to the right, putting the obstacle on the left of the vehicle.
- 2) The vehicle proceeds to follow the boundary of the obstacle, always keeping the obstacle on its left until one of the following conditions is met:
 - a) If the finishing criteria are satisfied, the vehicle goes straight to the target.
 - b) When the vehicle approaches the ST line, it treats it as an obstacle, turning to the right and keeping the line on its left if it moves closer to the target. Therefore, the algorithm causes the vehicle to stay on one side of the ST line, which will aid the vehicle in reaching the target quickly.
 - c) If the vehicle approaches the ST line and following it will drive the vehicle away from the target, it behaves as if the line is not an obstacle and crosses it.

The following finishing criteria must be simultaneously satisfied for the controller to allow the vehicle to go to the target:

- The vehicle must be within a distance r_b to the target. Larger distances may indicate that an obstacle lies between the vehicle and the target and may be undetected by the vehicle’s sensors.

- The vehicle must have a clear “line of sight” to the target. This condition is implemented by comparing the magnitude of the angle to the target to the magnitude of the angle to the nearest obstacle.

Schematics of the paths generated by the Bug module are shown in the two examples of Figure 10 and Figure 11.

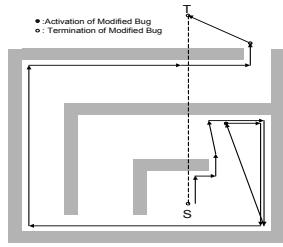


Fig. 10. Path 1 by Bug Steering Module.

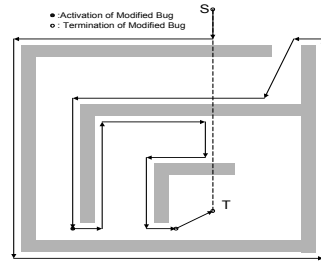


Fig. 11. Path 2 by Bug Steering Module.

The Bug Steering fuzzy module is designed to accommodate four-wheel vehicles, which cannot make the zero-radius turns that omnidirectional vehicles can. Thus, the module begins the appropriate steering adjustments ahead of time to allow the vehicle to make the necessary turns without colliding with obstacles or crossing the ST line. The inputs to the Bug module are the distance to the obstacle d_o , and the angle to the obstacle $\Delta\phi_o$. The output of this module is the correction in the steering angle $\Delta\alpha_3$.

Four membership functions, Big (B), Medium (M), Small (S) and Zero (Z) are used to describe the first input variable, d_o as shown in Figure 12. Seven membership functions, Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (Z), Positive Small (PS), Positive Medium (PM) and Positive Big (PB) are used to describe the second input variable, $\Delta\phi_o$, as shown in Figure 13. Five membership functions, Negative Medium (NM), Negative Small (NS), Zero (Z), Positive Small (PS) and Positive Medium (PM) are used to describe the output variable, $\Delta\alpha_3$, as shown in Figure 14. Note that the membership functions in this module are different from those of Figure 7 through Figure 9. This modification was necessary to improve the module response.

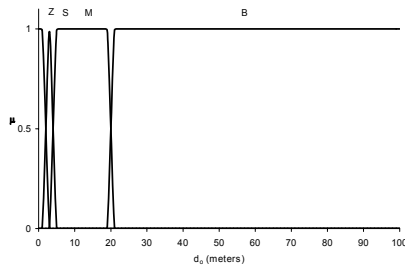


Fig. 12. Membership Functions for d_o .

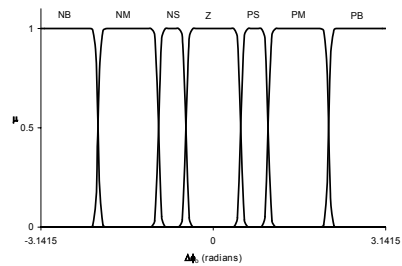


Fig. 13. Membership Functions for $\Delta\phi_o$.

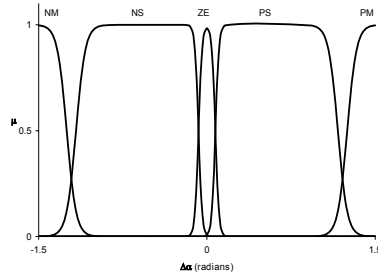


Fig. 14. Membership Functions for Output Variable $\Delta\alpha_3$.

As the rules in this module allow the vehicle to track obstacles, they are not very different from those of the Collision Avoidance Steering Fuzzy module. The rules of this module are shown in Table 3.

$d_0 \Rightarrow$ $\Delta\phi_0 \Downarrow$	Z	S	M	B
NB	PM	PM	PM	Z
NM	PM	PM	PM	Z
NS	PM	PM	PM	Z
Z	NM	NM	NS	Z
PS	NM	NS	NS	Z
PM	NS	Z	PS	Z
PB	PS	PS	PS	Z

Table 3. Rules for Modified Bug Steering Fuzzy Module.

4.2 Final Orientation Module

Orienting the vehicle in a particular direction at the target point (similar to parking a car in a parking lot) presents a challenging problem. This paper presents a simple solution for it by adding the fourth module in the steering controller. The final orientation consists of setting up “virtual” targets that are based on the desired final orientation. The controller guides the vehicle to the correct orientation by passing the vehicle through the virtual targets, which gradually orients the vehicle in the right direction. The coordinates of these target points are shown in Table 4. X_t and Y_t are the actual target coordinates and θ_t is the desired final orientation. The final orientation module is independent of the other three fuzzy steering modules. The steering fuzzy modules continue to control the vehicle as they otherwise would while the final orientation algorithm is operational.

Target	X coordinate (m)	Y coordinate (m)
T_{v1}	$X_t + 20\cos(\theta_t - \pi)$	$Y_t + 20\sin(\theta_t - \pi)$
T_{v2}	$X_t + 10\cos(\theta_t - \pi)$	$Y_t + 10\sin(\theta_t - \pi)$
T	X_t	Y_t

Table 4. Target Coordinates and Tracking Order.

Figure 15 shows a vehicle approaching the target. The final orientation algorithm first creates T_{v1} and drives the vehicle toward it. The algorithm then creates T_{v2} , and again drives the vehicle toward it. Finally, the algorithm allows the vehicle to detect the actual target, and the vehicle goes toward it.

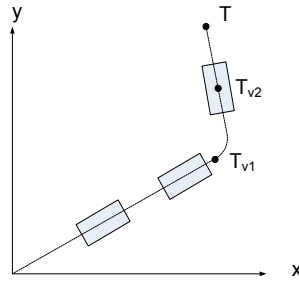


Fig. 15. Vehicle Approaching Target from a Specified Direction.

5. Velocity Fuzzy Control

To use human knowledge and experience efficiently in controlling the velocity of the vehicle, the problem is separated into several tasks. A fuzzy controller module is designed for each task. These tasks are target throttle, cornering throttle, and collision avoidance throttle. Figure 16 shows a schematic of the inputs and outputs of the three fuzzy modules to achieve these tasks. The total throttle/brake angle is the summation of the outputs of these modules.

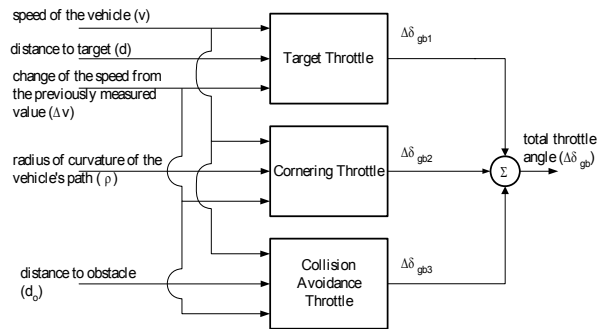


Fig. 16. Schematic Diagram of the Velocity Fuzzy Controller Modules.

5.1 Target Throttle Fuzzy Module

The objective of this module is to speed up the vehicle to reach the target, to slow it down when approaching the target and to stop it at the target position. The inputs to this module are the velocity of the vehicle v , distance to target d , and the change of velocity from the previously measured value Δv . The module has one output, which is the change in the gas pedal/brake angle $\Delta\delta_{gb1}$.

Four membership functions, Big (B), Medium (M), Small (S) and Zero (Z) are used to describe the first and the second input variables, v and d as shown in Figure 17 and 18 respectively. Note that the upper bound on the B membership set in Figure 17 is *twenty-five* meters per second, which indicates that the maximum velocity of the vehicle can be up to *ninety* kilometers per hour. Since the controller is designed for a totally autonomous vehicle, it is conservative; thus, the reduction of velocity due to the distance to the target, d , begins early to avoid the potential for collision. Three membership functions, Negative (N), Zero (Z), Positive (P) are used to describe the third input variable, Δv , as shown in Figure 19. These membership functions are selected to only indicate whether the velocity is decreasing, constant, or increasing. Finally, six membership functions, Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (Z), Positive Small (PS), and Positive Big (PB) are used to describe the output variable, $\Delta\delta_{gb1}$, as shown in Figure 20. Note that the membership functions for $\Delta\delta_{gb1}$ are not symmetric around zero since the vehicle should stop in less time than that it takes to accelerate the vehicle. The figure also shows that there are more negative functions than positive ones, which allows finer control over braking. The following is a sample of the rules of this module,

- If (v is B), (d is Z) and (Δv is P), then ($\Delta\delta_{gb1}$ is NB): The rule states that if the velocity is *big*, the distance to target is *zero*, and the change of velocity is *positive*, the controller should supply a *big* brake angle.
- If (v is Z), (d is B) and (Δv is N), then ($\Delta\delta_{gb1}$ is PB): This rule states that if the velocity is *zero*, the distance to target is *big*, and the change of velocity is *negative*, the controller should supply a *big* gas pedal angle.

The rules of this module are listed in Table 5.

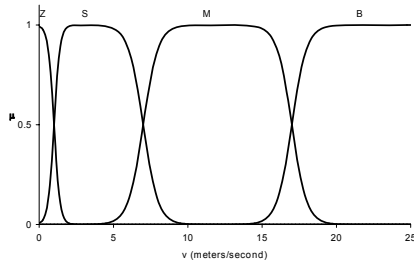


Fig. 17. Membership Functions for v .

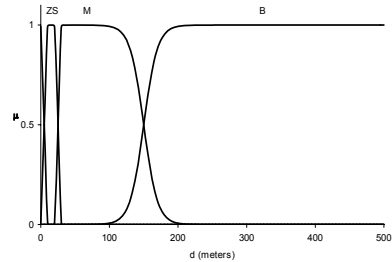


Fig. 18. Membership Functions for d .

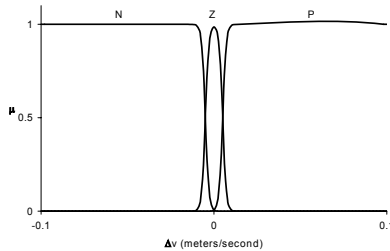


Fig. 19. Membership Functions for Δv .

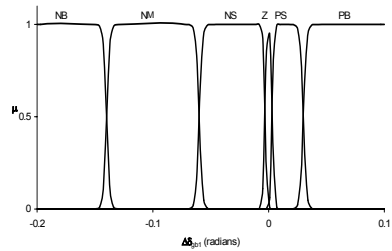


Fig. 20. Membership Functions for $\Delta\delta_{gb1}$.

$v \downarrow$	$d \Rightarrow \Delta v \downarrow$	Z	S	M	B
Z	N	PS	PS	PB	PB
	Z	PS	PS	PB	PB
	P	NS	Z	PB	PB
S	N	NS	Z	PS	PB
	Z	NS	NS	Z	PB
	P	NS	NM	NS	PB
M	N	NB	NS	PS	PS
	Z	NB	NM	Z	PS
	P	NB	NB	NM	Z
B	N	NS	NS	Z	PS
	Z	NB	NS	NS	Z
	P	NB	NB	NS	NB

Table 5. Rules for Target Throttle Fuzzy Module.

5.2. Cornering Throttle Fuzzy Module

The objective of this module is to slow down the vehicle when it is turning to ensure its stability. The inputs to this module are the velocity of the vehicle v , the radius of curvature of the vehicle’s path ρ (Figure 21), and the change of velocity from the previous measured value Δv . The module has one output, which is the change in the gas pedal/brake angle $\Delta\delta_{gb2}$. The membership functions of v , Δv , and $\Delta\delta_{gb2}$ are the same as those used in the target throttle module. However, $\Delta\delta_{gb2}$ uses only NS and Z membership functions since cornering module primarily reduces the velocity of the vehicle at sharp corners. Three membership functions, Z, S, B are used to describe the radius of curvature ρ , as shown in Figure 22. Note that these membership functions are selected to activate this module at sharp corners only.

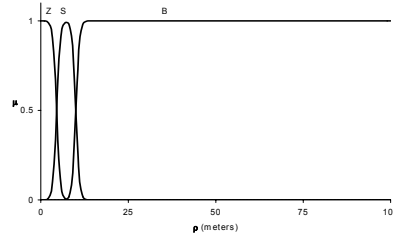
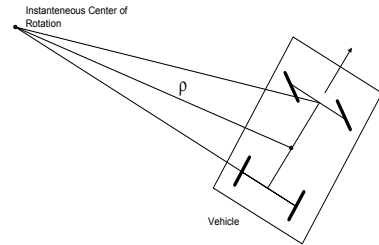


Fig. 21. Instantaneous Radius of Curvature. Fig. 22. Membership Functions for ρ .

The following is a sample of the rules of this module,

- If (v is B), (ρ is Z), and (Δv is P), then ($\Delta\delta_{gb2}$ is NS): The first rule states that if the velocity is big, the radius of curvature is zero, and the change of velocity is positive, the controller should supply a small brake angle.
- If (v is S), (ρ is B), and (Δv is Z), then ($\Delta\delta_{gb2}$ is Z): This rule states that if the velocity is small, the radius of curvature is big and the change of velocity is zero, the controller should do nothing.

The rules of this module are shown in Table 6.

$v \downarrow$	$\rho \Rightarrow$ $\Delta v \downarrow$	Z	S	B
Z	N	Z	Z	Z
	Z	Z	Z	Z
	P	Z	Z	Z
S	N	Z	Z	Z
	Z	Z	Z	Z
	P	NS	Z	Z
M	N	Z	Z	Z
	Z	NS	Z	Z
	P	NS	Z	Z
B	N	NS	Z	Z
	Z	NS	Z	Z
	P	NS	NS	Z

Table 6. Rules for Cornering Throttle Fuzzy Module.

5.3. Collision Avoidance Throttle Fuzzy Module

The goal of this module is to slow down the vehicle as it moves near obstacles. The inputs to this module are the velocity of the vehicle v , minimum measured distance to the nearest obstacle d_o , and the change of velocity from the previous measured value Δv . The output of the module is the change in the gas pedal/brake angle $\Delta\delta_{gb3}$.

The membership functions of v and Δv are the same as those used in the target throttle module. Figure 23 and Figure 24 show membership functions of d_o and $\Delta\delta_{gb3}$ respectively. Figure 23 shows that this module is activated when the distance from the vehicle to the obstacle becomes less than twenty meters.

The membership functions for v and Δv in the previous module are used in this module. Three membership functions, Z, S, B (Figure 23) are used to describe d_o . This module uses five membership functions, NB, NS, Z, PS, PB (Figure 24) to describe $\Delta\delta_{gb3}$. The minimum value of range of $\Delta\delta_{gb3}$ is significantly less than the minimum value of range of $\Delta\delta_{gb1}$ to indicate the need to brake the vehicle faster near obstacles.

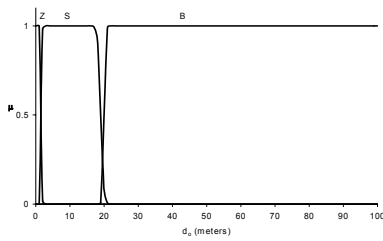


Fig. 23. Membership Functions for Input Variable, d_o .

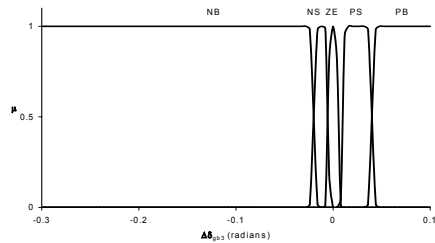


Fig. 24. Membership Functions for Output Variable, $\Delta\delta_{gb3}$.

The following is a sample of the rules of this module,

- If (v is B), (d_0 is Z), and (Δv is P), then ($\Delta\delta_{gb3}$ is NB): The rule states that if the velocity is *big*, the minimum measured distance to the nearest obstacle is *zero*, and the change of velocity is *positive*, the controller should supply a *big* brake angle.
- If (v is S), (d_0 is S), and (Δv is N), then ($\Delta\delta_{gb3}$ is Z): This rule states that if the velocity is *small*, the minimum measured distance to the nearest obstacle is *small*, and the change of velocity is *negative*, the controller should *do nothing*.

The full rule set of this module is shown in Table 7.

$v \Downarrow$	$d_0 \Rightarrow$ $\Delta v \Downarrow$	Z	S	B
Z	N	PS	PS	Z
	Z	PS	PS	Z
	P	NS	Z	Z
S	N	PS	Z	Z
	Z	PS	Z	Z
	P	NS	NS	Z
M	N	NS	Z	Z
	Z	NB	NS	Z
	P	NB	NB	Z
B	N	NS	NS	Z
	Z	NB	NS	Z
	P	NB	NB	Z

Table 7. Rules for Cornering Throttle Fuzzy Module.

6. Tuning Target Throttle Fuzzy Module

Tests show that cornering throttle and obstacle avoidance modules performed adequately after little manual tuning. However, the Target Throttle Fuzzy module consistently produces an oscillatory vehicle velocity profile as the vehicle approaches the target. Manual tuning of the membership functions and rules fails to improve the performance of this module. While the rules of this module generally fit within the driving patterns of most drivers, this oscillatory behavior can be easily explained by considering that two objectives of this module compete with each other near the target:

- Slowing down the vehicle as the vehicle approaches the target.
- Maintaining a non-zero velocity to ensure that the vehicle does not stop before reaching the target.

Usually drivers solve this problem by pressing the gas or brake pedals lightly when they approach their final target. Narrowing the ranges of the membership functions of the output variable of a fuzzy controller may simulate this behavior.

The design presented here proposes an adaptive tuning method to smooth the velocity of the vehicle as it approaches the target. Since the tuning is to be performed in real time, it is important to avoid intensive computations. Tuning the controller is based on the following decisions:

- i. The previous fuzzy rules of Section 5.1 are left unchanged.
- ii. The membership functions of the input variables are considered reasonable.
- iii. The membership functions of the output variable $\Delta\delta_{gb1}$, Figure 20, are used as an initial guess.
- iv. The Target Throttle Fuzzy module remains unchanged if the vehicle is far away from the target. This controller will be from now on labeled the *cruising throttle module*. Tuning becomes active only when the vehicle approaches the target. The tuned controller will be labeled the *slowing-down throttle module*.

The philosophy of tuning the membership functions can be still described using linguistic terms. The algorithm takes one of these actions:

- Case 1: If Δv is *positive* (accelerating), and $\Delta\delta_{gb}$ at the previous sample is also *positive*, attempt to decrease the velocity by moving the membership function of *positive small* (PS) of $\Delta\delta_{gb1}$ closer to zero.
- Case 2: If Δv is *negative* (decelerating), and $\Delta\delta_{gb}$ at the previous sample is also *negative*, attempt to increase the velocity by moving the membership function of *negative small* (NS) of $\Delta\delta_{gb1}$ closer to zero.
- Case 3: If Δv and $\Delta\delta_{gb}$ at the previous sample have different signs, it indicates that the controller is trying to stabilize the velocity. In this case, all the membership functions of $\Delta\delta_{gb1}$ should remain unchanged.

The product of sigmoid membership functions, given by Equation (11), offers the advantage of changing only one side of the membership function while keeping the other unchanged. To maintain smooth output of the controller, moving the right side of PS membership function requires a corresponding movement of the left side of PB membership function. Similarly, moving the left side of NS membership function requires a corresponding movement of the right side of NM membership function. That is, *boundaries* of membership functions must be moved such that there are no gaps created between membership functions. As such, the proposed scheme maintains the initial amount of overlap (or gap) between any adjacent membership functions. The controller incorporates the tuning system, Figure 25, which has these components:

- *Trigger*: If the distance is small, the tuning scheme is triggered. In this paper, a crisp trigger of comparing the distance to the target with D_{small} is used.
- *Cruising throttle module*: The module is the controller of Section 6.1.
- *Tuner*: The trainer tunes the membership functions of the output of target throttle module if the trigger is fired. Its decision is based on the three cases discussed above. The new membership functions are

$$\mu_{PB}(i) = \frac{1}{1 + e^{-a_{PBL}(X - \phi(i) C_{PBL}(i-1))}} \quad , \quad (13)$$

$$\mu_{PS}(i) = \frac{1}{1 + e^{-a_{PSL}(X - C_{PSL})}} \frac{1}{1 + e^{a_{PSR}(X - \phi(i) C_{PSR}(i-1))}} \quad , \quad (14)$$

$$\mu_{NS}(i) = \frac{1}{1 + e^{-a_{NSL}(X - \phi(i) C_{NSL}(i-1))}} \frac{1}{1 + e^{a_{NSR}(X - C_{NSR})}} \quad , \quad (15)$$

$$\mu_{NM}(i) = \frac{1}{1 + e^{-a_{NML}(x - c_{NML})}} \frac{1}{1 + e^{a_{NMR}(x - \phi(i)c_{NMR}(i-1))}} \quad (16)$$

When $|\Delta v|$ is significantly close to zero, the original membership functions should be used. Therefore, the value of the scaling factor $\phi(i)$ is equal to one. As the value of $|\Delta v|$ increases, the original membership functions are modified by reducing the value of $\phi(i)$. $\phi(i)$ is described using the following equation:

$$\begin{aligned} \phi(i) &= 0.25 - 300(|\Delta v(i)| - 0.0025) & \text{if } |\Delta v(i)| < 0.0025 \\ \phi(i) &= 0.25 & \text{if } |\Delta v(i)| \geq 0.0025 \end{aligned} \quad (17)$$

Product of sigmoid membership functions require that the center of the right side be of higher value than that of the left side. To maintain this characteristic, the minimum value of $\phi(i)$ is limited to 0.25. The *Slowing-down module* is similar to the *Cruising throttle module*. The only exception is that it uses the tuned membership functions produced by the *Tuner* for the output variable.

The parameter PI (i.e., the *Performance Index*) is used to assess the performance of the controller with and without the tuning scheme:

$$PI = \sum_{i=k} \Delta v(i)^2 \quad t(k) = t(D_{small}) \quad (18)$$

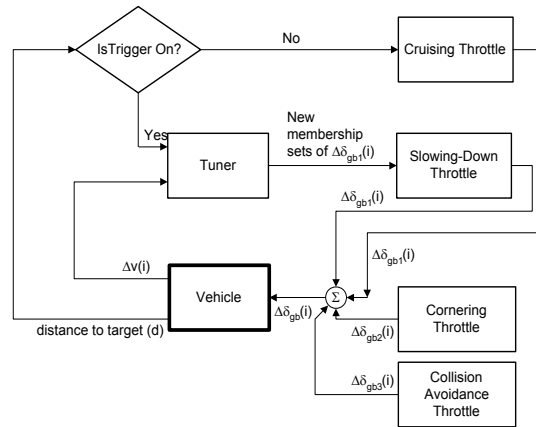


Fig. 25. Schematic of Target Throttle Tuning System.

7. Examples

The seven modules of the two proposed controllers are used to guide the vehicle around obstacles and reach the target in several examples to check the validity of the ideas proposed in this paper. In each example, the vehicle and the obstacle configurations are chosen to display certain characteristics of the controller or to contrast various modules of it.

7.1 Velocity Fuzzy Control Example

The following example, shown in Figure 26, demonstrates the abilities of the velocity fuzzy controller. The starting position and orientation are (0.0, 0.0) meters and zero degree respectively. The target position and orientation are (100.0, 60.0) meters and ninety degrees respectively. The buffer zone radius will be denoted by r_{bf} and is generally assigned a value of 20 meters. The sensors are sampled every 0.1 seconds while the time step for the simulation is 0.01 second. The example has both static obstacles and a dynamic obstacle that moves at a constant velocity. D_{small} is equal to 40 meters. The markers of Figure 26 correspond to $t_0 = 0$ seconds, $t_1 = 10$ seconds, $t_2 = 15$ seconds, $t_3 = 20$ seconds, and $t_4 = 45$ seconds. t_5 is equal to 99.5 seconds using the original target throttle module and 91.9 seconds using the target throttle tuning system. Figure 27 shows the velocity versus time of the vehicle at both cases. If no tuning is used, the velocity of the vehicle experiences an oscillatory pattern until it reaches the target. If the proposed tuning scheme is used, the velocity profile becomes smooth. The total path traversal time is also reduced. PI is equal to 0.0068 when the tuned controller is used, while PI is equal to 0.0566 when using the original cruising controller.

Figures 28 through 30 show the output of the modules, $\Delta\delta_{gb1}$ (with tuning), $\Delta\delta_{gb2}$, and $\Delta\delta_{gb3}$, respectively, when tuning is used. The target throttle controller (cruising controller) initially accelerates the vehicle. This controller later reduces its input to keep the velocity of the vehicle bounded. In the meantime, the cornering throttle controller attempts to slow the vehicle as it turns around the static obstacle. The obstacle avoidance controller has similar output at this stage. The effect of these controllers at t_1 becomes apparent as the vehicle starts to slow down. This trend continues until t_4 as the vehicle turns and tries to avoid both obstacles. The contribution of $\Delta\delta_{gb1}$ diminishes at this stage as this module slows the vehicle down while it moves closer to the target. The value of $\Delta\delta_{gb2}$ also approaches zero at this stage since the turning radii of the vehicle are reasonably larger. However, the obstacle avoidance controller is also active at this stage. This phase continues until the vehicle leaves the obstacles behind. The tuning trigger is activated directly after that point at around 46 seconds. The effect is clear in Figure 28. The reduction of $\Delta\delta_{gb1}$ is counteracted by the increase in $\Delta\delta_{gb3}$, which increases as the effects of the obstacles diminish. Eventually, the slowing-down module, which is the only active one at the end of the motion, moves the vehicle toward the target.

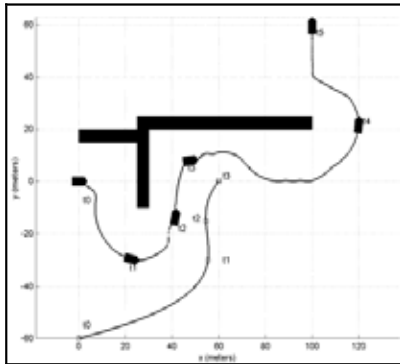


Fig. 26. Trajectory of Velocity Fuzzy Controller.

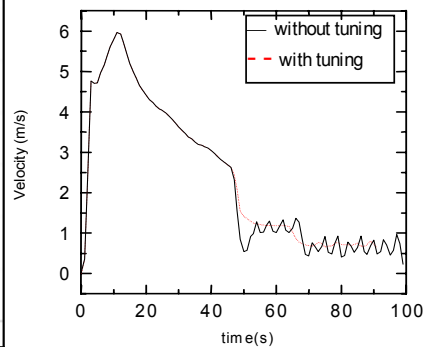


Fig. 27. Velocity Profiles.

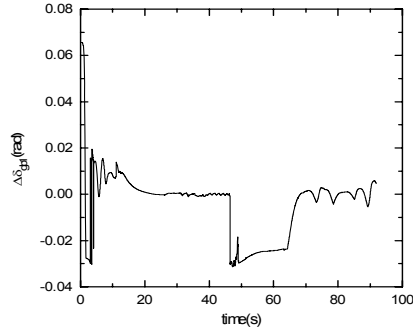


Fig. 28. Output of $\Delta\delta_{gb1}$.

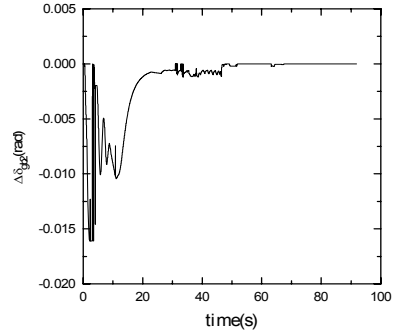


Fig. 29. Output of $\Delta\delta_{gb2}$.

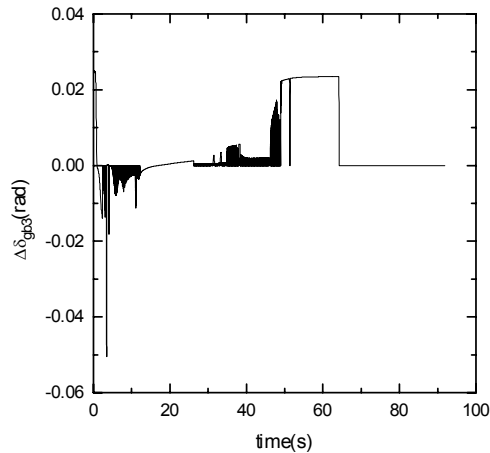


Fig. 30. Output of the Collision Avoidance Throttle Module, $\Delta\delta_{gb3}$.

7.2. Bug Steering Module Example

The conditions for the simulation are an initial position of (0.0, 0.0) meters, an initial orientation of zero degrees, a final position of (100.0, 100.0) meters and a final orientation of zero degrees. The results for this example are shown in Figure 31. The times depicted in the figures are $t_0 = 0$ seconds, $t_1 = 15$ seconds, $t_2 = 25$ seconds, $t_3 = 75$ seconds, $t_4 = 140$ seconds, and $t_5 = 145$ seconds, $t_6 = 170$ seconds, $t_7 = 198$ seconds, and $t_8 = 217.8$ seconds.

Initially, when there are no obstacles in its vicinity, the vehicle attempts to go straight to the target. Once the vehicle gets closer to the obstacle, it goes to the right, attempting to avoid it. When the vehicle senses an opening in the obstacle, it attempts to enter; however, as the opening is only 10 meters wide; thus, the controller will not allow the vehicle to enter since r_b is set to twenty meters. The vehicle then continues moving around the obstacles, attempting to find a clear path to the target. Once its orientation has deviated from the ST

angle by more than 360 degrees, the Modified Bug Fuzzy module is triggered. This occurs between times t_4 and t_5 . At that point, the vehicle attempts again to go straight to the target. Once it senses the obstacle, it turns right, keeping the obstacle at a distance of about three meters to the left of the vehicle. It follows the wall of the maze until about t_7 . At this point, the vehicle is within twenty meters of the target and the vehicle has an unobstructed line of sight to the target. The final orientation controller is switched on here. Thus, the vehicle goes straight to the target while maintaining a course that results in a final orientation of zero degrees. The controller of (Lee and Wang, 1994) starts in the same direction as the proposed controller but fails to enter the maze as it does not have a module for such a task, as shown in Figure 32.

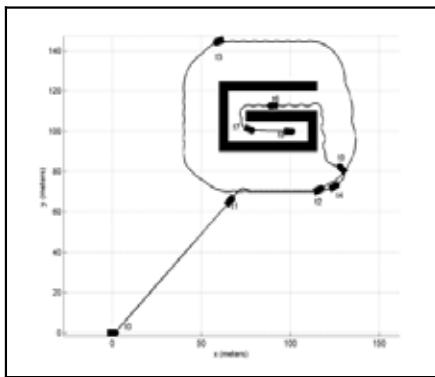


Fig. 31. Results of Using the Proposed Controller in Maze Tracking.

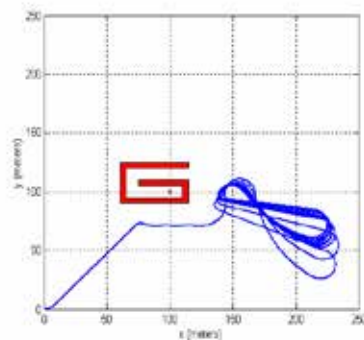


Fig. 32. Results of Using the Controller of (Lee and Wang, 1994) in the Maze of Figure 31 .

8. Conclusions

This chapter presents a fuzzy logic control system for steering a two-axle vehicle. Two controllers are presented individually: the steering controller and the velocity controller. Each fuzzy controller is divided into several modules to represent the distributed way in which humans deal with different driving tasks. All of these modules are of the Mamdani-type and use sigmoid or product of sigmoid membership functions. The outputs of the various modules are added together to control the steering angle and the speed of the vehicle, respectively.

Two fuzzy modules in the steering controller are designed to meet the basic driving requirement: Target Steering Fuzzy module and Collision Avoidance Steering Fuzzy module. The first module steers the vehicle toward the target by monitoring the steering angle and the angle to the target. The objective of the second module is to avoid collisions with static and dynamic obstacles. Its inputs are the distance and angle to the nearest obstacle. When the vehicle is near an obstacle, the output of this module is given a higher weight than that of the first module, so that it will be able to significantly affect the behavior of the vehicle.

The Modified Bug Steering Fuzzy module is proposed as a backup for the Target Steering Fuzzy module in the event that it cannot guide the vehicle to the target. The criterion used

to turn it on consists of monitoring the vehicle's global orientation and making sure that the vehicle's orientation does not change by more than 360 degrees. This module includes a start to target line as a fictitious obstacle. The vehicle follows this line if it encounters it, unless it drives away from the target, which reduces path length in many cases. The criteria used to turn this module off consist of making sure that i) the vehicle is within 20 meters of the target, and ii) the vehicle has an unobstructed line of sight to the target. Finally, the rules and membership functions of the Modified Bug Fuzzy module account for the physical properties of a two-axle vehicle by allowing it enough time and space to make the necessary turns.

A separate module in the steering controller adjusts the vehicle to the desired final orientation by introducing two intermediate target points. The module drives the vehicle through these two points and adjusts the vehicle's orientation accordingly.

The velocity controller is divided into several modules, each dealing with a separate objective, to mimic human behavior. These modules are:

- i. Target Throttle Fuzzy module: The objectives of the module are to start the vehicle moving from a complete stop, to speed it up, and to stop it when it gets sufficiently close to the target. The inputs to this module are the velocity of the vehicle, distance to target, and the change of velocity from the previously measured value. The module has one output, which is the change in the gas pedal/brake angle ($\Delta\delta_{gb1}$).
- ii. Cornering Throttle Fuzzy module: The objective of this module is to reduce the speed of the vehicle when its turning radius decreases (i.e., the tighter the turn, the lower the velocity). The inputs to this module are the velocity of the vehicle, the radius of curvature of the vehicle's path, and the change of velocity from the previous measured value. The module has one output, which is the change in the gas pedal/brake angle ($\Delta\delta_{gb2}$).
- iii. Collision Avoidance Throttle Fuzzy module: The objective of this module is to reduce the speed of the vehicle as it approaches an obstacle and increase the speed as the vehicle continues past the obstacle. The inputs to this module are the vehicle velocity, the minimum measured distance to the nearest obstacle, and the change of velocity from the previous measured value. The output of the module is the change in the gas pedal/brake angle ($\Delta\delta_{gb3}$).

The Cornering Throttle and Collision Avoidance Throttle Fuzzy modules performed satisfactorily after some manual tuning. The initial Target Throttle Fuzzy module, which produces an oscillatory velocity pattern when the vehicle approaches the target, may be explained by the fact that two objectives of this controller are in conflict at this stage:

- Decelerating the vehicle to stop at the target point.
- Accelerating the vehicle if the velocity approaches zero before the vehicle reaches the target point.

The Target Throttle Fuzzy module is tuned to eliminate these velocity oscillations near the target. This tuned controller is composed of two fuzzy logic controllers: *cruising module* and *slowing-down module*. The *cruising module* is similar to the Target Throttle Fuzzy module. It drives the vehicle while it is away from the target. The *slowing-down module* is triggered if the distance to the target is small. This controller is similar to the cruising controller. However, the ranges of the membership functions of its output are continuously varied based on two inputs: the current change in velocity and the change in the gas/brake angle at

the previous sample. This arrangement succeeds in driving the vehicle to its target without exciting oscillations in the velocity response.

Simulation examples show that the fuzzy controllers successfully guide the vehicle toward the target, while avoiding all obstacles placed in its path. Future work will focus on extending the proposed fuzzy logic controllers to other types of autonomous vehicles, such as autonomous underwater or unmanned aerial vehicles. An intelligent algorithm that is able to generate fuzzy rules and tune the parameters of the fuzzy logic controllers of a vehicle can reduce the implementing time in different systems. Real-time calculation of the throttle angle during cornering could be modeled, but that would require a four wheel model, with calculations of the force at each tire. Criteria to indicate loss of traction on the tires toward the inside of a turn could also be used as inputs for the Cornering Throttle Fuzzy Module.

9. References

- Baturone, I. ; Moreno-Velo, F. J., Sanchez-Solano, S. & Ollero, A. (2004). Automatic design of fuzzy controller for car-like autonomous robots, *IEEE Trans. on Fuzzy Systems*, Vol. 12, No. 4, 447-465
- Boada, M. J. L.; Boada, B. L. & Diaz, V. (2006). Integrated control of front-wheel steering and front braking forces on the basis of fuzzy logic, *Journal of Automobile Engineering*, 253-267
- Byrne R. & Abdallah, C. (1995). Design of a Model Reference Adaptive Controller for Vehicle Road Following, *Mathl. Comput. Modelling*, Vol. 22, No. 4, 343-354
- Cang, Y., Yung, N.H.C., Wang, D. W. (2003). A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance," *IEEE Trans. on Systems, Man and Cybernetics*, Part B, Vol.33, 17 -27
- Chen, Q. & Ozguner, U. (2005). Real-time navigation for autonomous vehicles: A fuzzy obstacle avoidance and goal approach algorithm, *Proceedings of the American Control Conference 3*, pp. 2153-2158, Oregon, June 2005, Portland
- Dai, X. ; Li, C. K. & Rad, A. B. (2005). An approach to tune fuzzy controller based on reinforcement learning for autonomous vehicle control, *IEEE Trans. on Intelligent Transportation systems*, Vol. 6, No. 3, 285-293
- El Hajjaji, A. & Bentalba, S. (2003). Fuzzy path tracking control for automatic steering of vehicles, *Robotics and Autonomous Syst.*, Vol. 43, 203-213
- Godjevac J. & Steele, N. (2001). Neuro-fuzzy control for basic mobile robot behaviour, In: *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Driankov D. & Saffotti, A., (Eds.), 97-117, Physica-Verlag, A Springer-Verlag Company, New York
- Hessburg T. & Tomizuka, M. (1994). Fuzzy Logic Control for Lateral Vehicle Guidance, *IEEE Trans. On Control Systems*, 55-63
- Hoffmann, F. (2001). The role of fuzzy logic control in evolutionary robotics, In: *Fuzzy Logic Techniques for autonomous vehicle navigation*, Driankov D. & Saffotti, A., (Eds.), 119-147, Physica-Verlag, A Springer-Verlag Company, New York
- Hong, C. (1997). Tuning the Fuzzy Control Autopilot Strategy for Driving Pattern Simulation of Road Vehicles, *International Journal of Vehicle Design*, 35-52
- Holzmann, H.; Germann, S., Halfmann, C. & Isermann, R. (1998). Intelligent Fuzzy Distance and Cruise Control for Passenger Cars, *Journal of Intelligent and Fuzzy Systems*, 315-327

- Kim, S. H.; Park, C. & Harashima, F. (2001). A self-organized fuzzy controller for wheeled mobile robot using an evolutionary algorithm, *IEEE Trans. on Industrial Electronics*, Vol. 48, 467-474
- Kodagoda, K. R. S.; Wijesoma, W. S. & Teoh, E. K. (2002). Fuzzy speed and steering control of an AGV, *IEEE Transactions on Control Systems Technology*, Vol. 10, 112-120
- Lee P. & Wang, L. (1994). Collision Avoidance by Fuzzy Logic Control for Automated Guided Vehicle Navigation, *Journal of Robotic Systems*, 743-760
- Lumelsky, V. & Stepanov, A. (1987). Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape, *Algoritmica*, Vol. 2, 403-430
- Lumelsky, V. & Skewis, T. (1988). A Paradigm for Incorporating Vision in the Robot Navigation Function, *IEEE Conference on Robotics and Automation*, 734-739, Philadelphia, Pennsylvania, April 1988
- Lumelsky, V. (1991). A Comparative Study on the Path Length Performance of Maze-Searching and Robot Motion Planning Algorithms, *IEEE Trans. on Robotics and Automation*, Vol. 7, No. 1, 57-66
- Maeda, M.; Maeda, Y., & Murakami, S. (1991). Fuzzy Drive Control of an Autonomous Mobile Robot, *Fuzzy Sets and Systems*, 195-204
- Murphy, R. R. (2001). Fuzzy Logic for fusion of practical influences on vehicle speed control, In: *Fuzzy Logic Techniques for autonomous vehicle navigation*, Driankov D. & Saffotti, A., (Eds.), 73-96, Physica-Verlag, A Springer-Verlag Company, New York
- Nobe, S.A. & Wang, F. Y. (2001). An overview of recent developments in automated lateral and longitudinal vehicle controls, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics 5*, pp. 3447-3452, Nashville, October, 2000
- Ohara, H. & Murakami, T. (2004). Tracking control of a compact electrical vehicle trailer based on equivalent dynamics model, *International Workshop on Advanced Motion Control*, AMC, pp. 183-186, Kawasaki, Japan, March 2004
- Ollero, A.; Ferrus, J., Sanchez, O. & Heredia, G. (2001). Mobile robot path tracking and visual target tracking using fuzzy logic, In: *Fuzzy Logic Techniques for autonomous vehicle navigation*, Driankov D. & Saffotti, A., (Eds.), 51-72, Physica-Verlag, A Springer-Verlag Company, New York
- Pasquier, M.; Quek, C. & Toh, M. (2001) Fuzzylot: a novel self-organising fuzzy-neural rule-based pilot system for automated vehicles, *Neural networks*, vol. 14, 1099-1112
- Peng H. & Tomizuka, M. (1993). Preview Control for Vehicle Lateral Guidance in Highway Automation, *Journal of Dynamic Systems, Measurement, and Control*, 679-686
- Rosa, R.; & Garcia-Alegre, M. (1991). Fuzzy Logic Strategies to Control an Autonomous Mobile Robot, *Cybernetics and Systems*, 267-276
- Sanchez, O.; Ollero, A. & Heredia, G. (1999). Hierarchical fuzzy path tracking and velocity control of autonomous vehicles, *Integrated Computer-Aided Engineering*, Vol. 6, 289-301
- Sauerberger M. & Trabia, M. (1996). Design and Implementation of an Autonomous Control Algorithm for an Omnidirectional Vehicle, *Proceedings of the IASTED International Conference on Robotics and Manufacturing*, 131-134
- Seraji, H. (2005). SmartNav: A rule-free fuzzy approach to rover navigation, *Journal of Robotic Systems*, Vol. 22, No. 12, 795-808

- Wheeler M. & Shoureshi, R. (1994). A Fuzzy Driver on the Handling Track, *Dynamic Systems and Control*, vol. 1, 431-439
- Wong, J. (1993). *Theory of Ground Vehicles*, 2nd ed., John Wiley and Sons, Inc., New York, NY
- Ye, C. & Wang, D.W. (2001). A novel navigation method for autonomous mobile vehicles, *Journal of Intelligent & Robotic Systems*, Vol. 32, 361-388

Real-Time Optimization Approach for Mobile Robot

Hiroki Takeuchi

University of Michigan

FXB Building 1320 Beal Avenue Ann Arbor, MI 48109-214

U.S.A.

1. Introduction

Optimization technique has been applied at many engineering aspects. Especially, it was used as a trajectory analysis of aircraft or spacecraft at the beginning. As computer environment grows up, such application comes to focus on more complicated and dynamic problem. One of them is mobile robot.

On another front, optimization algorithm itself has been developed for on-line use. Classic method like as gradient method has been applied for off-line computing. Off-line technique can not be applied to real-time use like as robot control. However, after new method such as receding horizon control was emerged, the ability for real-time control use comes to be practical.

What the brain of human or animal executes is, as it were, "optimization". As the result, it generates very smooth motion. Our goal is to realize such optimization and make mobile robot motion more smart and intelligent.

Receding Horizon Control is one of potential optimization technique. This algorithm is oriented for on-line use and practical optimization. This chapter provides some results of, (1) RHC formulation including ZMP balance condition, (2) RHC formulation including ZMP balance condition and Swing leg state variable constraint. Generally, theoretical treatment of Zero Moment Point is hard to solve. ZMP variable has an aspect of intermediate variable and it is somewhat mysterious. The technique can give a deal for such formulation. The numerical model is defined as simple one in the formulation of (1), then the proposed method can be applied to any legged robot which has to handle ZMP variable. Simulation results are also provided.

2. Real-Time Optimization

Generally, Gradient Method is most popular method in optimization technique. However, its long calculation time has made hurdle to use at real-time optimization. The procedure which arbitrary trajectory for whole time converges into optimal solution using gradient makes a defect. This procedure also may cause the trajectory sink to local minimum.

There are several real-time optimization algorithms called Receding Horizon Control. Those have good and bad points. In this chapter, two methods are introduced. The one is Receding Horizon Gradient Method, the other is backward sweep method which use transversality condition and it is used for the application in section 3 and section 4.

2.1 Receding Horizon Gradient Method

2.1.1 Continuation Method

Gradient Method has conventionally eliminated a problem that the initial condition of state equation and terminal condition of co-state equation are known although terminal condition of co-state equation and initial condition of co-state equation is unknown in TPBVP. Gradient Method is the method that the initial trajectory which is assigned as whole time ($t=0 \rightarrow t=t_f$) on real-time axis converges to optimal solution along its gradient. Defects of this method are;

- (1) It could converge to minimum solution
- (2) It takes a good amount of time to converge.
- (3) This initial trajectory must be considered for immediate convergence each time.

Then a Continuation Method eliminated these problems in place of Gradient Method (Ohtsuka,1997). If the interval time in Euler-Lagrange equations set 0, the solution comes to be trivial. The optimal solution can be chased to extend the time little by little based on this trivial solution. Generally, Predictor-Corrector Method is used for pursuit in Continuation Method. However it is too slow to execute on real-time control sequence.

Then a method which transverse condition is applied to balance to 0 eliminated this problem (ohtsuka,1997). The process to handle matrix operation needs longer calculation time if the numerical model is larger scale. In this research, a proposed method eliminates formula operation without Euler-Lagrange equations as much as possible. Such attempt may be also said that it returns to its basic focus on.

2.1.2 Gradient

When the terminal time T on performance index interval perturbs $T+dT$, the trajectory also perturbs. The extended performance index is described as:

$$J^* = \varphi[x^*(t, \tau)] + \int_t^{t+T} L + \lambda^T(t, \tau) \cdot \{ f[x(t, \tau), u(t, \tau)] - \dot{x}(t, \tau) \} d\tau \quad (2.1)$$

Then the perturbation is described as:

$$\delta J^* = \left[\frac{\partial \varphi}{\partial x} + H \right]_{\tau=T} dT + [\varphi_x[x(t, \tau)] - \lambda^T(t, \tau)]_{\tau=T} \delta x(t, T) + \int_t^{t+T} \{ H_x + \lambda^T(t, \tau) \} \cdot \delta x + H_u \delta u + \delta \lambda^T (f - \dot{x}) d\tau \quad (2.2)$$

When the terminal time T on performance index interval perturbs $T+dT$, it causes $\delta x, \delta \lambda, \delta u$. δu dominates $\delta x, \delta \lambda$ by state equation and co-state equation.

If the trajectory satisfies Euler-Lagrange equations, the gradient must be 0. If the trajectory perturb from the optimal solution, it cause the generation of gradient. In Continuation Method, the gradient of the initial trajectory equals 0, then it perturbs $T+dT$, causes a bit of gradient, and it is recovered.

2.1.3 Sampling Interval

The movement of performance index interval along real-time is executed each sampling interval. Although the time length of the performance index interval is extended, it is little and changes smoothly. This enables the time length re-scaled along the extension.

In reference (Ohtsuka, 1997), the number of the sampling interval arrays is fixed. In this case the sampling interval is growing longer along real-time, and it must be considered how to get re-scaled initial state value, co-state value, and input value. In this paper, the time length of the array is fixed; the number of these arrays is growing along real-time. $d\tau = \text{constant} = dt$. This aims for practical use. It can replace the initial array of x, λ, u on performance index interval as the next step array of x, λ, u on real-time directly. The simulation results confirm that this is feasible.

Therefore the algorithm proposed here is:

- (1) $t=0, \tau=0$, trivial solution
- (2) $T_{i+1} = T_i + d\tau$
- (3) Rescale trajectory array of input variable on τ axis
- (4) Calculate trajectory of state variable with state equation
- (5) Calculate trajectory of co-state variable with co-state equation
- (6) Calculate the gradient H_u
- (7) If the gradient sufficiently closed to 0, then go to (10)
- (8) Update the trajectory of input variable on τ axis $u_{\text{new}} = u_{\text{old}} + \alpha \cdot \text{gradient}$
- (9) $\alpha = \alpha + \text{step}$; Go to (4)
- (10) The initial array of input variable on τ axis replaces the next step array of input variable on t axis.
- (11) Calculate state value and co-state value from (10)

Various rescaling could be considered at (3). One of them is to use the input variable trajectory one step ago and rescale it like as Fig. 2. Somewhat gradient is invoked by rescaled input variable trajectory, however, such gradient could be expected sufficiently small from the viewpoint of Continuation Method.

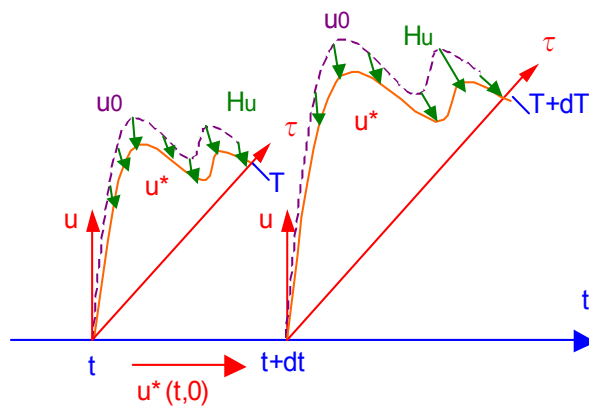


Fig. 1. Differential changes in the terminal time.

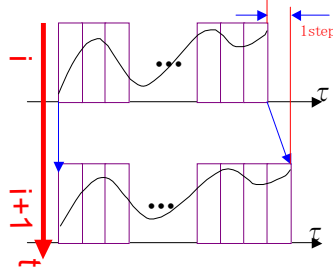


Fig. 2. Scaling for input variable.

A simple example is arranged here. The state equation is described as:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} (1 - x_1^2(t) - x_2^2(t))x_1(t) - x_2(t) + u(t) \\ x_1(t) \end{bmatrix} \quad (2.3)$$

The state variables are $x_1(t)$, $x_2(t)$, and input variable is $u(t)$. Performance index is described as:

$$J = 2(x_1^2(t_f) + x_2^2(t_f)) + \int_0^{t_f} (x_1^2(t) + x_2^2(t) + u^2(t))dt \quad (2.4)$$

The evaluated interval is moving and extended along real-time in Receding Horizon Control procedure, then it is described as:

$$J = 2(x_1^2(t, T) + x_2^2(t, T)) + \int_t^{t+T} (x_1^2(t, \tau) + x_2^2(t, \tau) + u^2(t, \tau))d\tau \quad (2.5)$$

Fig. 3 shows the comparison between conventional gradient method (Fletcher-Reeves Method) and proposed method. The trajectories of state variables and control inputs are matched each other. Fig. 3 (f), (g) error values transition described as Equation (2.6) endorse this fact. The each error is closed to 0 sufficiently.

$$\text{error}(t) = \lambda^*(t, T) - \varphi[x^*(t, T)] \quad (2.6)$$

Mathematica3.0 on Windows OS executes this calculation.

	RHGM	Gradient Method
Simulation Time	5.0 s	5.0 s
dt	10.0 ms	50.0ms
Continuation Terminal Time	$T_{i+1}=T_i+10.0\text{ms}$	-
Maximum of Terminal Time	0.5 s	-
Maximum number of iteration in a step	40	30
Initial Condition x	{0.0, 2.0}	{0.0, 2.0}
Reference x_f	{0.0, 0.0}	{0.0, 0.0}

Table 1. Simulation Data.

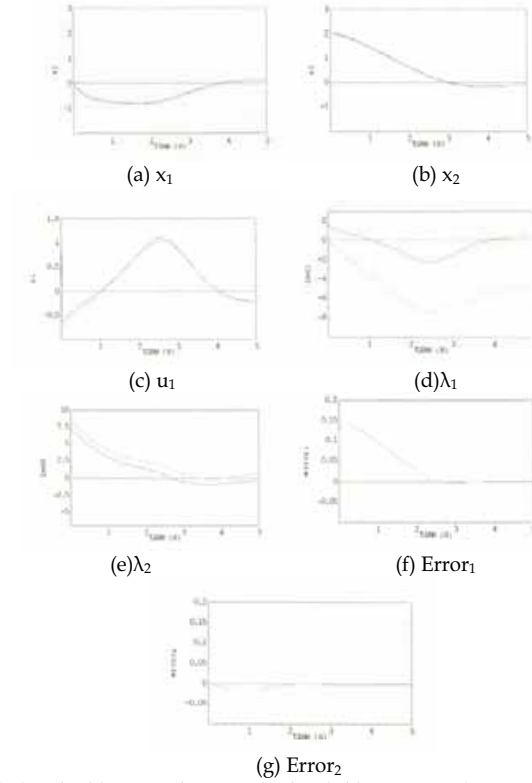


Fig. 3. Example (Dashed line: gradient method, solid line: RHGM).

A. Nonlinear link system

Nonlinear three-link system can also be solved by proposed method. The state equation is described as:

$$\begin{bmatrix} \theta_1(t) \\ \theta_2(t) \\ \theta_3(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ \dot{\theta}_3(t) \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ \dot{\theta}_3(t) \\ \mathbf{M}^{-1}(\Theta) \cdot (\mathbf{u}(t) - \mathbf{V}(\Theta, \dot{\Theta}) - \mathbf{G}(\Theta)) \end{bmatrix} \quad (2.7)$$

$$\Theta = \begin{bmatrix} \theta_1(t) \\ \theta_2(t) \\ \theta_3(t) \end{bmatrix}$$

The equation involves nonlinearity of vertical three-link system. M means inertial matrix, V means corioles term, G means gravity term. Performance index is defined as:

$$J = (x_f(t,T) - x(t,T)) \cdot S_f \cdot (x_f(t,T) - x(t,T)) + \int_t^{t+T} (u \cdot R \cdot u + x \cdot Q \cdot x) d\tau \quad (2.8)$$

Fig. 5 shows the result of this simulation. Table 2 describes calculation parameters and physical parameters of the links. Fig. 5 (q)-(v) shows the error of the transverse condition (Equation(2.6)). Simulation was done on Mathematica 5.0. One of the advantages of RHC is that it can treat with singular target point. RHC algorithm does not contain Jacobian matrix, then it can treat with such problem. The simulation result shows that the three link manipulator reaches to the singular target attitude.

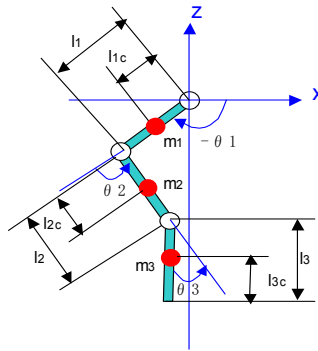
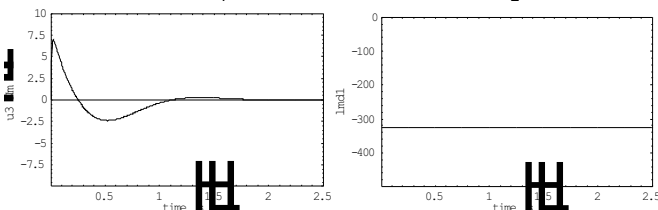
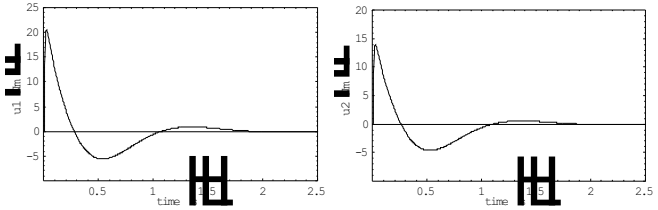
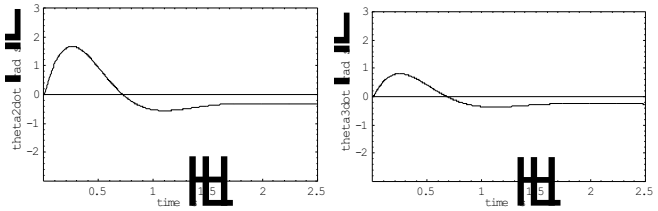
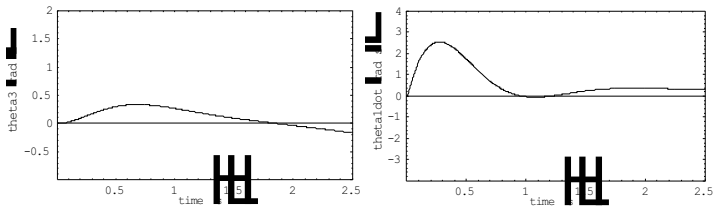
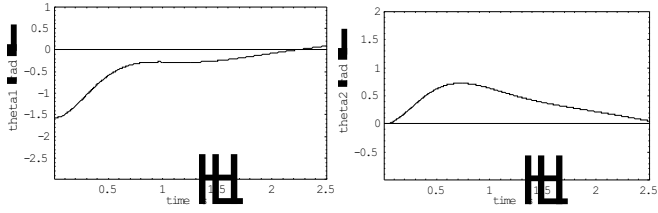
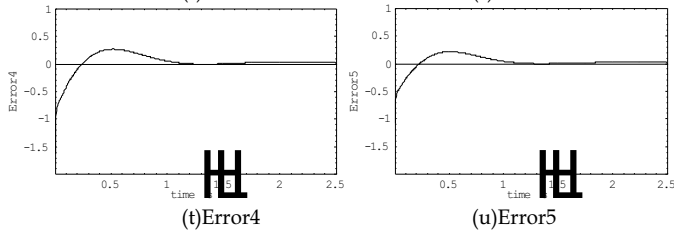
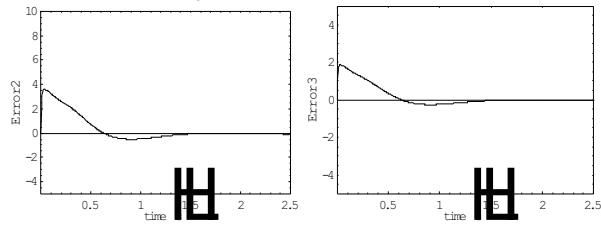
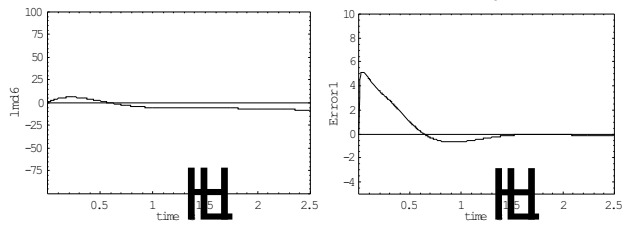
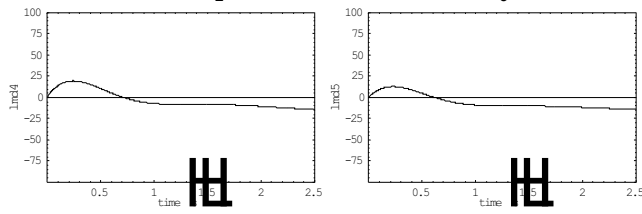
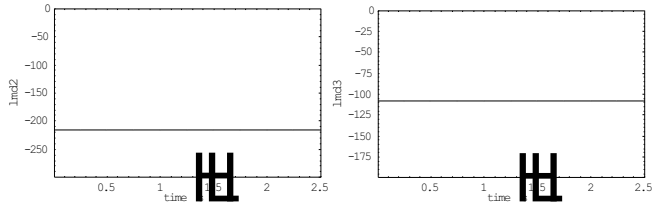


Fig. 4. Nonlinear link system.

Simulation Time	2.5 s
dt	5.0 ms
Continuation Terminal Time	$T_{i+1}=T_i+dt$
Maximum of Terminal Time	0.5 s
Maximum number of iteration in a step	40
Initial Condition	$\{-1.57,0,0,0,0,0,0,0,0\}$
Reference Condition	$\{0,0,0,0,0,0,0,0,0\}$
S_f	$\{1,0,1,0,1,0,0,1,0,1,0,1\}$
R	$\{1,0,1,0,1,0\}$
Q	$\{1,0,1,0,1,0,1,0,1,0,1,0\}$
m_1	0.5kg
m_2	0.5kg
m_3	0.5kg
l_1	0.3m
l_{1c}	0.2m
l_2	0.3m
l_{2c}	0.2m
l_3	0.3m
l_{3c}	0.2m

Table 2. Simulation Data





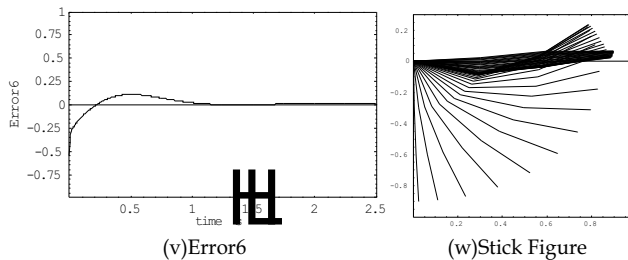


Fig. 5. Nonlinear 2- link system.

3. Particular Condition for Legged Robot – ZMP Balance Condition –

Legged robot has specific balance condition attributable to the unstable dynamics. It needs to contrive ways to involve such condition into formulation.

While the legged robot is standing on one foot, the point at which the center of gravity (C.G.) of the robot is projected onto the ground must be located on the sole plane to enable it static walk. While standing on two feet, there must be a point on the plane, which connects both the soles. While standing on four feet, there must be a point on the polygon, which consist of the four soles. While the robot moves, in order to be stabilized dynamically and to walk, the same concept is required. Generally, this is called ZMP (Zero Moment Point, (Vukobratovic,1969)). ZMP within sagittal plane can be expressed as follows from link $i=0$ to $i=n$.

ZMP is the point on the ground where ground reaction forces are applied.

$$x_{zmp} = \frac{-\sum_{i=0}^n m_i(-g+z_i)x + \sum_{i=0}^n m_i \ddot{x}_i z_i}{\sum_{i=0}^n m_i(-g+z_i)} \tag{3.1}$$

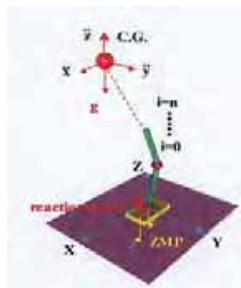


Fig. 6. Definition of Zero Moment Point.



Fig. 7. Definition of sagittal plane and frontal plane.

g is gravity acceleration and m is the mass of each link. If the “ M ” is represented for the whole mass,

$$x_{zmp} = \frac{-M(-g+\ddot{z})x + M\ddot{z}}{M(-g+\ddot{z})} \quad (3.2)$$

This equation means that the sum total of moment of the point-mass around the origin of the coordinate balances with the moment generated by the ZMP distance from the origin and the reaction force from the ground. If the ZMP is located in the polygon constituted by the soles as well as the point at which the C.G. projects itself onto the ground in a static walk, the robot is stabilized and a dynamic walk can be carried out. If the ZMP runs-over from this polygon, it will cause the robot to fall and it cannot continue to walk.

An attempt to converge the ZMP to a referenced ZMP trajectory by using feedback control in recent years has been performed (Hirai, K., Hirose, M., Haikawa, Y., Takenaka, T., 1998). Then, how a ZMP reference trajectory could be generated poses the next problem.

In the conventional research of legged robot, there are two variables “ x ” and “ z ” in Equation(3.2) and poses a problem in solving the ZMP variable. Because it is not solved uniquely. An optimization problem must be solved to obtain a solution. If the condition, which holds a center of gravity position at fixed height, is added, we can avoid this problem temporarily. Then, the variable \ddot{z} in the equation is set to 0, the equation could be described as follows, and a pseudo solution is uniquely obtained.

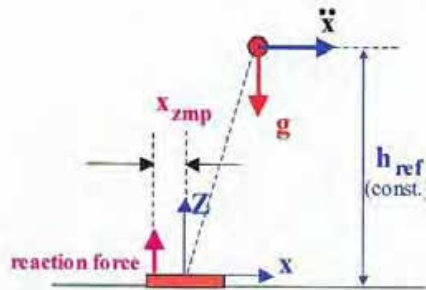


Fig. 8. Pseudo ZMP.

In the conventional legged robot research, one of big problems was to compute ZMP by Equation(3.2) since there are two variables of “ x ” and “ z ”. To avoid this problem, some treatments had been concerned (mitobe, 2002). The main concept was to make the numerical model unique. Nonlinear dynamic equation is linearized adding constraints and reduced to unique equation like as A constraint below is added:

$$z = k \cdot x + h_{ref} \quad (3.3)$$

h_{ref} denotes a fixed height. Then we have linear equation:

$$\ddot{x} = \frac{g}{h_{ref}} \cdot x + \frac{1}{m \cdot h_{ref}} \tau \quad (3.4)$$

τ denotes ankle joint torque. The value of ZMP can be obtained from τ and sensed value of floor reaction force. Then the ZMP could be in proportion with the acceleration of x . One of

the defects in this equation is that the necessary torque for whole the robot is collected on the ankle joint. Redundancy, which the robot possesses, could not be utilized effectively. Equation(3.3) is defined arbitrarily by designer.

- (1) Excessive torque for whole the robot is converged to the ankle joint
- (2) Robot motion is restricted because the constraint is adopted (The motion is on a linear line)
- (3) Solutions for another joints without the ankle joint could not be obtained

The second item implies that robot motion is not natural. What it takes to utilize redundancy of a robot is optimization. Since iterative calculation is needed in the optimization by the gradient method, such technique usually turns into off-line calculation. However, in a robot control that the real-time performance is required, off-line optimization is disadvantageous. When an unexpected situation appears, it could not be coped with.

Many engineering applications require real-time solutions of optimization problems. However, traditional algorithms for digital computers may not provide real-time optimization. An attractive and promising approach was introduced to real-time solutions for optimization problems known as Receding Horizon Control. This new optimization technique goes into the practical usage stage. Since RHC does not use a gradient method for optimization, it can carry out calculation processing of the optimal solution in short time such as a real-time control interval. Although much research has been conducted in respect to the theory, applying RHC to robotics still has no actual example. This paper describes ZMP control of the legged robot using RHC proving that real-time optimization is available. Furthermore, it proposes a method of generating the optimum ZMP reference (Takeuchi,2003).

3.1 Equality Constraint Formulation

RHC formulation without constraints has been performed backwards. In this section, RHC containing the equality constraint is focused on and explained.

The state equation to treat,

$$\dot{x}(t) = f[x(t), u(t)] \quad (3.5)$$

As equality constraint,

$$C[x(t), u(t)] = 0 \quad (3.6)$$

If equality constraint condition can be used, it is convenient when formulating a problem like real-time control of a robot.

The performance index is defined as,

$$J = \phi[x^*(t+T)] + \int_t^{t+T} L[x^*(t, \tau), u^*(t, \tau)] d\tau \quad (3.7)$$

RHC has added superscript * to the variable on the time-axis τ which moves, in order that the evaluation section may move with time. The left side of the bracket (t, τ) means the real time "t", and the right side of this bracket means the time on the τ axis. Hamiltonian is described as,

$$H = L + \lambda^{*T} \cdot f + \rho^{*T} \cdot C \quad (3.8)$$

λ, ρ are co-state variables. Euler-Lagrange equations are described as,

$$\dot{\mathbf{x}}^*(t, \tau) = \mathbf{H}_x^T \quad (3.9)$$

$$\dot{\lambda}^*(t, \tau) = -\mathbf{H}_x^T \quad (3.10)$$

$$\mathbf{H}_u = 0 \quad (3.11)$$

$$\mathbf{C}[\mathbf{x}^*(t, \tau), \mathbf{u}^*(t, \tau)] = 0 \quad (3.12)$$

$$\mathbf{x}^*(t, 0) = \mathbf{x}(t) \quad (3.13)$$

$$\lambda^*(t, T) = \phi_x^T[\mathbf{x}^*(t, T)] \quad (3.14)$$

It considers obtaining a solution using the continuation method (Ohtsuka, 1997). The perturbation from an optimal path is described as:

$$\delta \ddot{\mathbf{x}} = \mathbf{f}_x^T \cdot \delta \dot{\mathbf{x}} - \mathbf{f}_u^T \cdot \delta \mathbf{u} - \mathbf{f}_\rho^T \cdot \delta \rho \quad (3.15)$$

$$\delta \dot{\lambda} = -\mathbf{H}_{xx} \cdot \delta \dot{\mathbf{x}} - \mathbf{f}_x^T \cdot \delta \dot{\lambda} - \mathbf{H}_{xu} \cdot \delta \mathbf{u} - \mathbf{H}_{x\rho} \cdot \delta \rho \quad (3.16)$$

$$\mathbf{H}_{xu} \cdot \delta \dot{\mathbf{x}} + \mathbf{f}_u^T \cdot \delta \dot{\lambda} + \mathbf{H}_{uu} \cdot \delta \mathbf{u} - \mathbf{H}_{u\rho} \cdot \delta \rho = 0 \quad (3.17)$$

$$\mathbf{C}_x \cdot \delta \dot{\mathbf{x}} + \mathbf{C}_u \cdot \delta \mathbf{u} = 0 \quad (3.18)$$

Here, $\delta \mathbf{u}$ and $\delta \rho$ are eliminable if Equation(3.17), Equation(3.18) are solved as simultaneous equations. Then Equation(3.15), Equation(3.16) are described as:

$$\frac{d}{dt} \begin{bmatrix} \delta \dot{\mathbf{x}} \\ \delta \dot{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \delta \dot{\mathbf{x}} \\ \delta \dot{\lambda} \end{bmatrix} \quad (3.19)$$

$$\mathbf{A} = \mathbf{f}_x + \mathbf{f}_\rho \cdot \mathbf{H}_{\rho\rho}^{-1} \cdot \mathbf{H}_{\rho x} - \mathbf{f}_\rho \cdot \mathbf{H}_{\rho\rho}^{-1} \cdot \mathbf{H}_{\rho u} \cdot \mathbf{C}_u^{-1} \cdot \mathbf{C}_x - \mathbf{f}_u \cdot \mathbf{C}_u^{-1} \cdot \mathbf{C}_x \quad (3.20)$$

$$\mathbf{B} = \mathbf{f}_\rho \cdot \mathbf{H}_{\rho\rho}^{-1} \cdot \mathbf{f}_u^T \quad (3.21)$$

$$\mathbf{C} = -\mathbf{H}_{xx} + \mathbf{H}_{xu} \cdot \mathbf{C}_u^{-1} \cdot \mathbf{C}_x - \mathbf{H}_{x\rho} \cdot \mathbf{H}_{\rho\rho}^{-1} \cdot \mathbf{H}_{\rho x} - \mathbf{H}_{x\rho} \cdot \mathbf{H}_{\rho\rho}^{-1} \cdot \mathbf{H}_{\rho u} \cdot \mathbf{C}_u^{-1} \cdot \mathbf{C}_x \quad (3.22)$$

$$\mathbf{D} = -\mathbf{f}_x^T - \mathbf{H}_{x\rho} \cdot \mathbf{H}_{\rho\rho}^{-1} \cdot \mathbf{f}_u^T \quad (3.23)$$

The subsequent calculation method follows the continuation method which Ohtsuka and Fujii developed(Ohtsuka,1997). This is explained briefly below. This technique pursues the optimal solution so that the error F of the transversality conditions of an Euler-Lagrange equation is converged to 0.

$$\frac{d}{dt} F[\lambda(t), \mathbf{x}(t), T(t)] = \text{Coeff} \cdot F[\lambda(t), \mathbf{x}(t), T(t)] \quad (3.24)$$

Thus, F can be stabilized. The equation below is assumed here.

$$\delta \dot{\lambda}^*(t, \tau) = \delta \mathbf{S}^*(t, \tau) \cdot \delta \dot{\mathbf{x}}^*(t, \tau) + \mathbf{c}^*(t, \tau) dt \quad (3.25)$$

This is substituted for Equation(3.19).

$$\dot{S}_r = D \cdot S^* - S^* \cdot A - S^* \cdot B \cdot S^* + C \quad (3.26)$$

$$c_r^* = (D - S^* \cdot B) \cdot c^* \quad (3.27)$$

This terminal value is acquired from Equation(3.25). An $S^*(t, 0)$, $c^*(t, 0)$ will be acquired if it finds the integral from the terminal value along time reversely.

$$\dot{\lambda}(t) = S^*(t, 0) \cdot \dot{x}(t) + c^*(t, 0) \quad (3.28)$$

Then, optimized $\lambda(t)$ will be obtained if it integrates with the upper equation on real time. Also optimized $u(t)$ can be obtained from $H_u = 0$.

3.2 Model Expression as a Point Mass

Modeling of robot mechanics has been studied for many years. One of the ways is to describe nonlinearity of the robot precisely. However such dynamic equations leads to a result that the equation itself is too much complicated to treat. Such modeling also could not be applicable to even similar type robots. From this point of view, a concept of simple modeling has been emerged. Modeling simply leads to be basic and flexible to apply it to various type controlled objects. The fundamental treatment should be formulated at first, and then applied to more complicated modeling properly.

However, overdo of reduction and linearization in modeling leads to many defects mentioned in 5.1 Introduction. The modeling must be simple, but also, must be with wide application.

In this study, when a legged robot is modeled, the whole robot is treated as a point mass of most fundamental case. Treating the whole robot center of gravity as inverted pendulum is a technique generally performed. According to such simple modeling, a method applicable to a biped robot, a quadruped robot, and other multi-legged type robots can be proposed so that Chapter\ref{chap:application} may describe. First, in order to help understanding, it deals with a problem at a 2-dimensional plane. As an input of a system, it sets setting up the acceleration of axis "x" and "z", $u_x = \ddot{x}$, $u_z = \ddot{z}$. Gravity is applied to a perpendicular lower part.

$$\frac{d}{dt} \begin{bmatrix} x(t) \\ z(t) \\ \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ u_x(t) \\ u_x(t) - g \end{bmatrix} \quad (3.29)$$

Although this is considered on Sagittal plane, if it considers at Frontal plane,

$$\frac{d}{dt} \begin{bmatrix} y(t) \\ z(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{y}(t) \\ \dot{z}(t) \\ u_y(t) \\ u_y(t) - g \end{bmatrix} \quad (3.30)$$

This modeling does not include ankle joint torque explicitly. Then the problem mentioned in 5.1 Introduction could not be emerged. The potential of the redundancy of

the robot could be left and it produces extensive utility. Furthermore, we can obtain ZMP trajectory.

3.3 Equal Constraints

Difficulties in taking the ZMP variable into a state equation as a state variable complicates the problem in formulation. Because the right side of Equation(3.2) has the dimension of acceleration, it causes differentiation of acceleration. Then, the use of the equality constraint expressing ZMP eliminates this problem.

$$x_{zmp}(t) \cdot (u_z(t) - g) = x(t) \cdot (u_z(t) - g) - z(t) \cdot u_x(t) \quad (3.31)$$

This means that the total moment by the acceleration inputs and gravity balances with the moment by the ZMP distance and reaction force from the ground. ZMP is the point of satisfying Equation(3.31). If ZMP is located in the polygon constituted by the sole plane, it can support the reaction force without generating any moment. Furthermore, this paper proposes that the 3rd input u_{zmp} substitute for x_{zmp} , then this idea compliments using the ZMP variable in formulation.

It is not necessary for U_{zmp} to be included in the state equation.

$$u_{zmp}(t) \cdot (u_z(t) - g) = x(t) \cdot (u_z(t) - g) - z(t) \cdot u_x(t) \quad (3.32)$$

If ZMP is treated as one of the inputs, when an optimum solution is calculated, the optimum ZMP input will be obtained simultaneously. At this point, the equal constraint has an important role.

The performance index is created using norm of inputs. Here, features include that the term of u_{zmp} is added in the performance index. Thus, by setting up the solution, which minimizes the norm of each axial acceleration and the ZMP sway, will be calculated. Considering that the ZMP may not sway within the sole of the robot, this has lead to the design of the robot's sole to be as small as possible.

$$\begin{aligned} J &= X^T \cdot S_f \cdot X + \int_0^{t+T} (X^T \cdot Q \cdot X + U^T \cdot R \cdot U) d\tau \\ X &= [x_f - x(t, T), y_f - y(t, T), \dot{x}_f - \dot{x}(t, T), \dot{y}_f - \dot{y}(t, T)]^T \\ U &= [u_x^*(t, \tau), u_z^*(t, \tau), u_{zmp}^*(t, \tau)]^T \end{aligned} \quad (3.33)$$

S_f and R are diagonal weight matrix.

3.4 Application

A legged robot is generally a nonlinear mechanical multi-link system. In order to compensate such nonlinearity, using this technique together with the conventional nonlinear control technique is also worth consideration. The real-time optimum solution using RHC can be used as reference trajectories. On the other hand, nonlinear control in modern control theory performs control of the whole multi-link system of the robot to converge to the reference trajectories.

When we have a powerful CPU for real-time control, it is possible that it formulates everything in a nonlinear model using RHC alone.

Using the conventional control method for the legged robot, the optimum reference trajectory have to be prepared before the real-time control. If the robot encounters an unexpected situation, the robot cannot cope with it. In regards to this point, the technique proposed is practical.

The optimum ZMP input, which is obtained using RHC, can generate the reference trajectory. Then a control system block diagram could be constructed so that actual ZMP may follow this reference trajectory. Since $u_x, u_y,$ and u_z are obtained for optimum inputs, they are newly regarded as reference trajectories.

Thus, this simple method is applicable also to a biped robot, a quadruped robot, and other multipled robots.

3.4.1 In the Case of a Biped

In the case of the biped, the phase of leg behavior could be divided into a support phase and a swing phase. The ZMP must be respectively settled in the sole planes. The ZMP must transition in these planes, if the initial point and the terminal point are set as in Fig.9. Control of the nonlinear force is performed in the nonlinear control part as is shown in the Fig.10 control block diagram. Collecting the calculation results in which each parameter is changed makes it possible to predict the range of ZMP sway in a sole. This is a very interesting point. In design of a biped robot, an index can be obtained which has an influence on the design dimensions of the sole plane.

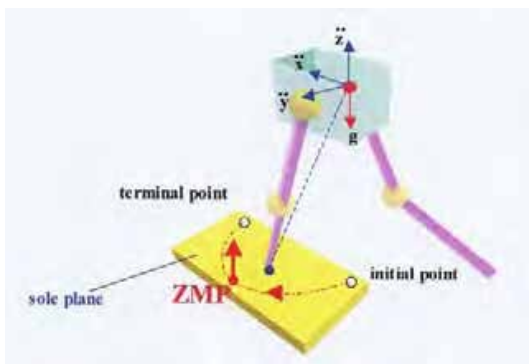


Fig. 9. Case of Biped.

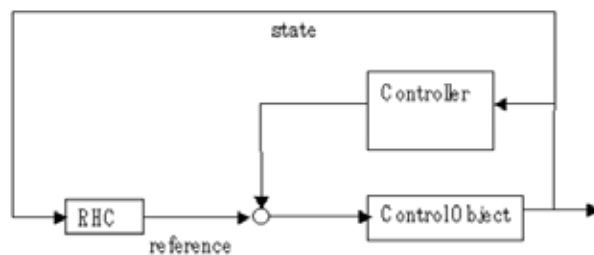


Fig. 10. Control Block Diagram.

3.4.2 In the Case of a Quadruped

The legs can be treated as the support legs pair and the swing legs pair by turns in ideal gaits. The polygon which is constituted by some sets of legs can secure a larger plane compared with a biped robot's case. For example, when a quadruped robot performs a "trot gait", it is necessary to repeat a support phase - swing phase for the diagonal leg entirely by turns. In Fig.11, step planes cross in the trot gait. If the starting point of ZMP is set at the tip of this plane, it can run to the tip of the following plane. Then a stable locomotion pattern will be achieved.



Fig. 11. Case of Quadruped.

4. Swing Leg Condition – State Variable Constraints –

The point mass modeling for formulation of Receding Horizon Control is introduced in the preceding section. That idea is to look at the robot model in perspective. We are in the next stage how the swing legs should be treated. The legged robot motion can be categorized into two phases support leg phase and swing leg phase depending on whether or not they are in contact with the floor while the legged robot is moving. The support legs support the robot against gravity while the swing legs are swung forward and then prepared for the subsequent support phase.

Since a real robot's legs behave nonlinearly, nonlinear forces interfere with the motion of the swing legs when they move. It is not practical to include all nonlinearities of the robot in a state equation. Therefore, this chapter proposes a simple formulation that reflects the nonlinearity of the motion of swing legs as far as possible. The root joint of the swing leg is connected to the center of gravity of the robot in sagittal plane and the acceleration is transferred to the root joint while the center of gravity satisfies the conditions of the ZMP constraint. This formulation can be easily applied to multi-legged robots such as biped, quadruped, and hexapod by simple addition of the equations of motion of the swing legs.

It is necessary to apply the constraint on the swing legs such that their position is always above floor level. The absence of this constraint could create a situation in which the legs would be positioned below the floor. Constraints such as these can be described as inequality state variable constraints. This section describes a method for solving the problem by means of the slack variable method.

4.1 Modeling of Swing Leg

Generally, a mathematical model of a legged robot increases in complexity if we try to describe the nonlinearity of its movement precisely, with the result that the equations of motion become

too complex for the robot control to apply. A mathematical model that requires less calculation is needed to shorten the control interval and allow real-time control of the swing legs. (Takeuchi,2003) describes the ZMP condition on the sagittal plane by using a constraint equation.

$$x_{zmp}(t) \cdot (u_z - g) = x(t) \cdot (u_z(t) - g) - z(t) \cdot u_x(t) \tag{4.1}$$

This indicates that the moment, which consists of X_{zmp} and the reaction force of the floor, balances with the gross moment around the origin of the coordinate system. Kinetic balance is maintained as long as X_{zmp} is in the area of the sole (biped), or in the area of the polygon described by the points where the toes of the grounding legs touch the floor (quadruped). The state equation of the center of gravity of a robot in sagittal plane is described as follows.

$$\frac{d}{dt} \begin{bmatrix} x(t) \\ z(t) \\ \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ u_x(t) \\ u_z(t) - g \end{bmatrix} \tag{4.2}$$

At practical control of legged robot, the control of swing legs must also be taken into account. We have therefore described the swing legs as a nonlinear two-link coordinate system and appended it to the state equation of the center of gravity of the robot. This approach is based on the assumption that the root joint of the swing leg will receive the acceleration of the center of gravity of the whole robot.

$$\frac{d}{dt} \begin{bmatrix} x(t) \\ z(t) \\ \theta_1(t) \\ \theta_2(t) \\ \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ u_x(t) \\ u_z(t) - g \\ M^{-1}(\Theta)(u(t) - V(\Theta, \dot{\Theta}) - G(\Theta)) \end{bmatrix} \tag{4.3}$$

$$\Theta = [\theta_1(t), \theta_2(t)]^T$$

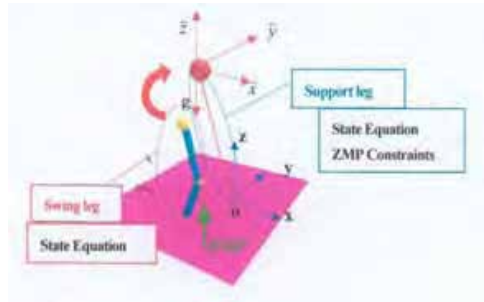


Fig. 12. Swing leg Formulation Model.

Where M is the inertia matrix, V is the Coriolis term, and G is the gravity term. The acceleration of inertia originating in the acceleration of the center of gravity of the whole robot is input to the acceleration of the root joint of the swing leg at the first step of the Newton-Euler method. We set it at $u_x, u_z - g$ in the Newton-Euler method for swing leg.

Equation(4.3) can be easily expanded to biped, quadruped, and another type of legged robot by appending nonlinear equations of motion of swing legs to the state equation of the center of gravity of the robot.

In this method,

- (1) The nonlinearity of swing legs can be taken into account.
- (2) The state equation is simple and practical.
- (3) The equations for motion of swing legs are easily appended to state equations.

Therefore, this formulation can be extended to multi-legged robots. It enable us to make feasible formation.

4.2 Performance Index

Equation (4.4) is used as an evaluation function.

$$J = \phi[x(t+T)] + \frac{1}{2} \int_0^{t+T} \{(\dot{x}'(\tau) - x_t)^\top \cdot Q \cdot (\dot{x}'(\tau) - x_t) + (u'(\tau) - u_t)^\top \cdot R \cdot (u'(\tau) - u_t)\} d\tau \quad (4.4)$$

$$\phi[x'(t+T)] = (x'(t+T) - x_t)^\top \cdot S_t \cdot (x'(t+T) - x_t)$$

$$(u'(\tau) - u_t)^\top \cdot R \cdot (u'(\tau) - u_t) = r_1 \cdot u_x'^2(\tau) + r_2 \cdot (u_z'(\tau) - g)^2 + r_3 \cdot u_{zmp}'^2(\tau) + r_4 \cdot u_{a_1}'^2(\tau) + r_5 \cdot u_{b_2}'^2(\tau)$$

The first term is a terminal constraint. Including U_{zmp} in the performance index is a novel approach which allows us to obtain a solution that minimizes input of acceleration in each axis direction, input of each joint torque of swing leg, and norm of acceleration in each axis direction, input of each joint torque of swing leg, and norm of ZMP. It reduces sway of ZMP in the area of the robot sole (biped) or in the area of the polygon whose vertexes are described by the toes of the grounding legs (quadruped).

4.3 Slack Variable Method

The slack variable $d(t)$ is introduced to make the control input explicit.

$$\text{height} + l_1 \cdot \sin \theta_1(t) + l_2 \cdot \sin(\theta_1(t) + \theta_2(t)) - d^2(t) = 0 \quad (4.5)$$

First, the inequality constraint is converted to an equal constraint equation. We described this as $S(x, t) = 0$, and differentiate it with respect to time until the control input appears in the equation.

$$\frac{dS}{dt} = \frac{\partial S}{\partial t} + \frac{\partial S}{\partial x} \dot{x} = \frac{\partial S}{\partial t} + \frac{\partial S}{\partial u} f(x, u) \quad (4.6)$$

In this case, the control input will appear by differentiating the equation twice.

$$l_1 \cdot \dot{\theta}_1(t) \cdot \cos \theta_1(t) + l_2 \cdot (\dot{\theta}_1(t) + \dot{\theta}_2(t)) \cdot \cos(\theta_1(t) + \theta_2(t)) - 2 \cdot d(t) \cdot \dot{d}(t) = 0 \quad (4.7)$$

$$\begin{aligned}
& l_1 \cdot u_{\theta_1}(t) \cdot \cos \theta_1(t) + l_2 \cdot (u_{\theta_1}(t) + u_{\theta_2}(t)) \cdot \cos(\theta_1(t) + \theta_2(t)) \\
& - l_1 \cdot \dot{\theta}_1^2(t) \cdot \sin \theta_1(t) - l_2 \cdot (\dot{\theta}_1(t) + \dot{\theta}_2(t))^2 \cdot \sin(\theta_1(t) + \theta_2(t)) \\
& - 2 \cdot \dot{d}^2(t) = 0
\end{aligned} \tag{4.8}$$

By introducing new state variables to the state equation,

$$\frac{d}{dt} d(t) = \dot{d}(t) \tag{4.9}$$

$$\frac{d}{dt} (\dot{d}(t)) = u_{\text{sack}}(t) \tag{4.10}$$

The equation can be written with the new input variable u_{sack} . The initial values $d(t)$ and $\dot{d}(t)$ which satisfy equations (4.9) and (4.10) need to be determined. In the case of Table 3, $d = 0.36$, and $\dot{d}(t) = 0$

$$\frac{d}{dt} \begin{bmatrix} x(t) \\ z(t) \\ \theta_1(t) \\ \theta_2(t) \\ \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ d(t) \\ \dot{d}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ u_x(t) \\ u_z(t) - g \\ u_{\theta_1}(t) \\ u_{\theta_2}(t) \\ \dot{d}(t) \\ u_{\text{sack}}(t) \end{bmatrix} \tag{4.11}$$

The number of state variables becomes ten on addition of two new variables, and the input variables become six by adding one. The ZMP condition can be described as follows.

$$u_{\text{zmp}}(t) \cdot (u_z(t) - g) = x(t) \cdot (u_z(t) - g) - z(t) \cdot u_x(t) \tag{4.12}$$

\subsection{Performance Index}

Equation(4.13) is used as an performance index.

$$\begin{aligned}
J &= \phi[x(t+T)] + \frac{1}{2} \int_0^{t+T} \{ (x(\tau) - x_f)^T \cdot Q \cdot (x(\tau) - x_f) + (u(\tau) - u_f)^T \cdot R \cdot (u(\tau) - u_f) \} d\tau \\
\phi[x(t+T)] &= (u(t+T) - u_f)^T \cdot S_f \cdot (u(t+T) - u_f)
\end{aligned} \tag{4.13}$$

The first term is a terminal constraint. After the state variables are converged to the reference states, the inputs will be 0. Solving by RHC, the input of the Z-axis component will start to be generated again when the altitude of the center of gravity begins to drop. Major transition in the altitude causes major transition in its ZMP variable. To decrease these, we devised a method in which the terms related to u_z in the function of performance index are replaced by $u_z - g$ and a constant value is previously allocated to u_z . This method is effective for problems including the gravity term.

4.4 Numerical Calculation (Nonlinear Case)

We then formulated the problem, including the nonlinear term of the swing legs.

$$\frac{d}{dt} \begin{bmatrix} x(t) \\ z(t) \\ \theta_1(t) \\ \theta_2(t) \\ \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ d(t) \\ \dot{d}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ u_x(t) \\ u_z(t) - g \\ M^{-1}(\Theta)(u(t) - V(\Theta, \dot{\Theta}) - G(\Theta)) \\ \dot{d}(t) \\ u_{\text{sack}}(t) \end{bmatrix} \quad (4.14)$$

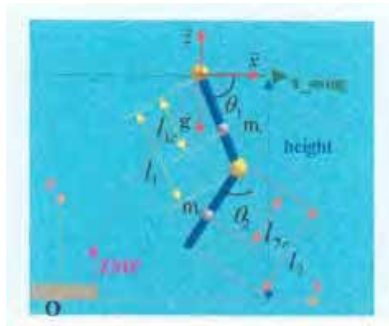
In this case, Equation(4.14) is substituted into $u_1(t)$ and $u_2(t)$ in Equation(4.15).

$$\begin{bmatrix} u_{\theta_1}(t) \\ u_{\theta_2}(t) \end{bmatrix} = M(\Theta) \cdot \ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) \quad (4.15)$$

The link parameters used in the calculation are listed in Fig. 13 and the parameters used in the optimization are shown in Table 3. The simulation time was set at 1.0s and its calculation took 0.9s. The simulation solution converged more rapidly than the case of linear. The control interval is set at 2.0ms. How much nonlinearity can be taken into account in the formulation depends on the capacity of the computer used. More nonlinearity can be included in the formulation if a higher performance computer is used.

S_f	diag[10.0,200.0,20.0,10.0,1.0 $\times 10^{-4}$,\$1.0 \times 10^{-4}\$,\$1.0 \times 10^{-4}\$,\$1.0 \times 10^{-4}\$,1.0,1.0]
Q	diag[1.0,4.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
R	diag[1.0,1.0,1.0,1.0,10.0,1.0]
x_0	[0.0,1.0,-1.7,-0.6,0.27,0.0,0.36,0.0] ^T ,
x_f	[0.2,1.0,-1.2,-1.0,0.27,0.0,0.0,0.0] ^T ,
u_f	[0.0,9.8,0.0,0.0,0.1,0.0] ^T ,
T_f	0.1s
Δt	2ms
Number of divide	5
ζ	450
height	1.0m

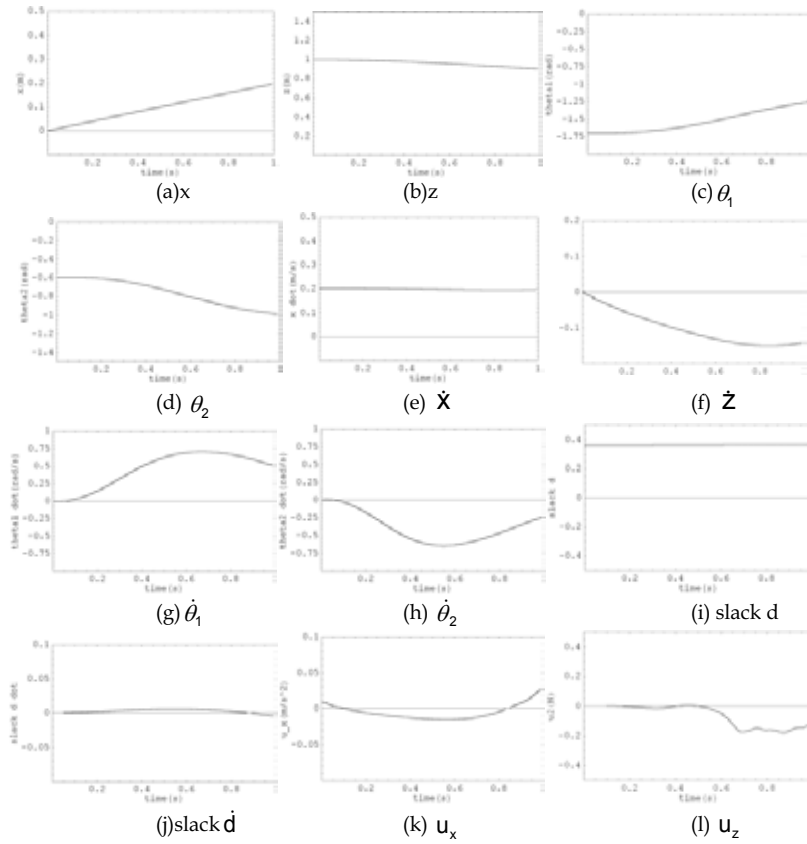
Table 3. Parameters for Simulation (Nonlinear Case).



link parameters

l_1	0.5m
l_2	0.5m
l_{1c}	0.5m
l_{2c}	0.5m
m_1	0.5kg
m_2	0.5kg

Fig. 13. The link model.



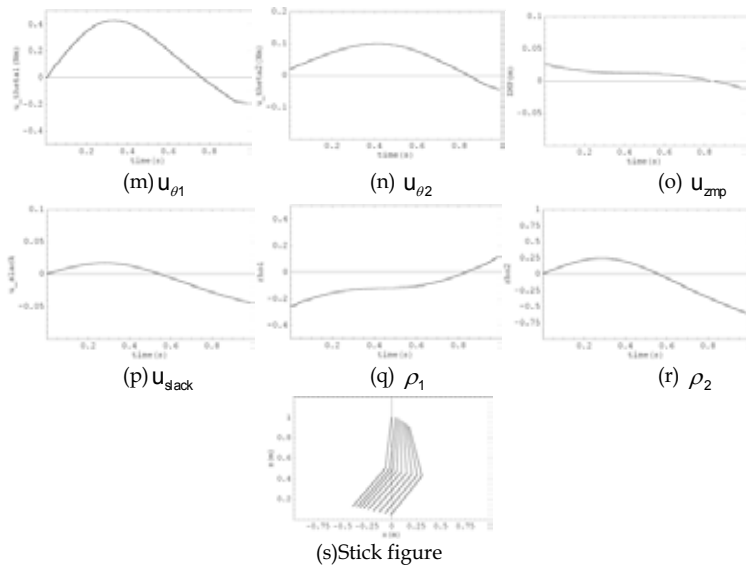


Fig. 14. Simulation(Slack Variable Method).

5. References

- Bryson Jr, A. E., Ho, Y.C., Applied Optimal Control, Hemisphere, 1975
- Hirai, K., Hirose, M., Haikawa, Y., Takenaka, T. (1998). The Development of Honda humanoid Robot, Proc. IEEE Int. Conf. Robotics and Automation, pp. 1321-1326
- Mitobe, K., Masuyama, A., Shibata, T., Yamano, M., Nasu, Y. (2002). A ZMP Manipulation Method for Walking Robots and its Application to Angular Momentum Control, Journal of Robot Society of Japan Vol. 20, No. 5, pp. 53-58
- Ohtsuka, T., Fujii, H. (1997). Real-Time Optimization Algorithm for Nonlinear Receding Horizon Control, Automatica, 33-6, p.p. 1147-1154
- Vukobratovic, M., Juricic, D. (1969), Contributions to the synthesis of biped gait. IEEE Trans on Biomedical Engineering, BME-16:1-6
- Vukobratovic, M., Frank, M. A. A., Juricic, D. (1970). On the Stability of Biped Locomotion, IEEE Trans on Biomedical Engineering, BME-17, No.1, pp. 25-36
- D. Q. Mayne, H. Michalska, Receding Horizon Control of Nonlinear Systems, IEEE Trans, AC-35-7, pp. 814-824, 1990
- Mayne, D. Q., Michalska, H., Robust receding horizon control of constrained nonlinear systems, IEEE Trans. on AC, Vol. 38, No. 11, pp. 1623-1633, 1993
- J. L. Speyer and A. E. Bryson Optimal Programming Problems with a Bounded State Space, AIAA journal, vol. 6, p.p. 1488-1492, 1968
- Takeuchi, H. (2003). Real Time Optimization for Robot Control using Receding Horizon Control with Equal Constraint, Journal of Robotic Systems, Vol. 20, No. 1, p.p. 3-13
- Takeuchi, H. (2004). Real Time Optimization and Control of Legged Robot - Formulation with Inequality State Constraint - , Journal of Robotic Systems, Vol. 21, No. 4, p.p. 153-166

Design and Control of an Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels

Jae-Bok Song*, Kyung-Seok Byun**

**Korea University, ** Mokpo National University
Republic of Korea*

1. Introduction

Applications of wheeled mobile robots have recently extended to service robots for the handicapped or the aged and industrial mobile robots working in various environments. The most popular wheeled mobile robots are equipped with two independent driving wheels. Since these robots possess 2 degrees-of-freedom (DOFs), they can rotate about any point, but cannot perform holonomic motion including sideways motion. To overcome this type of motion limitation, omnidirectional mobile robots (OMRs) were proposed. They can move in an arbitrary direction without changing the direction of the wheels, because they can achieve 3 DOF motion on a 2-dimensional plane. Various types of omnidirectional mobile robots have been proposed so far; universal wheels (Blumrich, 1974) (Ilou, 1975), ball wheels (West & Asada, 1997), off-centered wheels (Wada & Mory, 1996) are popular among them.

The omnidirectional mobile robots using omnidirectional wheels composed of passive rollers or balls usually have 3 or 4 wheels. The three-wheeled omnidirectional mobile robots are capable of achieving 3 DOF motions by driving 3 independent actuators (Carlisle, 1983) (Pin & Killough, 1999), but they may have stability problem due to the triangular contact area with the ground, especially when traveling on a ramp with the high center of gravity owing to the payload they carry. It is desirable, therefore, that four-wheeled vehicles be used when stability is of great concern (Muir & Neuman, 1987). However, independent drive of four wheels creates one extra DOF. To cope with such a redundancy problem, the mechanism capable of driving four omnidirectional wheels using three actuators was suggested (Asama et al., 1995).

Another approach to a redundant DOF is to devise some mechanism which uses this redundancy to change wheel arrangements (Wada & Asada, 1999) (Tahboub & Asada, 2000). It is called a variable footprint mechanism (VFM). Since the relationship between the robot velocity and the wheel velocities depends on wheel arrangement, varying wheel arrangement can function as a transmission. Furthermore, it can be considered as a continuously-variable transmission (CVT), because the robot velocity can change continuously by adjustment of wheel arrangements without employing a gear train. The CVT is useful to most mobile robots which have electric motors as actuators and a battery as a power source. Energy efficiency is of great importance in mobile robots because it is directly related to the operating time without

recharging. Some mobile robots are equipped with a transmission system, but most mobile robots use gear trains with a fixed gear ratio, because the transmission is heavy, bulky, and expensive. Therefore, transmission based on wheel arrangement provides possibility of energy efficient drive. The CVT can provide more efficient motor driving capability as its range of velocity ratio gets wider. The mobile robot proposed by (Wada & Asada, 1999), however, has a limited range to ensure stability of the vehicle.

In this research, an omnidirectional mobile robot with steerable omnidirectional wheels (OMR-SOW) shown in Fig. 1 is proposed to improve CVT performance in which robot stability is guaranteed regardless of wheel arrangement and thus the range of velocity ratio is greatly extended. The OMR-SOW is an omnidirectional mobile robot with 3 DOF motion and 1 DOF in steering. The steering DOF can be achieved by synchronously steerable omnidirectional wheels. While the VFM has a common steering axis for all four wheels, the OMR-SOW has an independent steering axis for each wheel. Therefore, the OMR-SOW possesses a wider range of velocity ratio without stability degradation. The four-wheeled omnidirectional mobile robot involving this mechanism combined with the continuous alternate wheels developed in our laboratory (Byun & Song, 2003) has been developed.



Fig. 1. Photo of OMR-SOW.

The OMR-SOW has some drawbacks. When the omnidirectional capability is not required especially, in normal straight-line driving, the omnidirectional mechanism tends to prevent the robot from driving efficiently. In this case, the wheel arrangement used in the automobile (i.e., 4 wheels in parallel) is preferred to the omnidirectional mechanism. Furthermore, the maximum height of a surmountable bump for OMR is limited by the radius of the passive roller of the omnidirectional wheel, which is much smaller than the radius of the wheel for the ordinary mobile robot. To overcome these drawbacks, the robot should function as an ordinary mobile robot unless its task requires omnidirectional capability. In this research, a new mechanism, which can be used as a differential drive mechanism as well as an omnidirectional one, is proposed.

The remainder of this chapter is organized as follows. In Section 2, the structure of a variable wheel arrangement mechanism is introduced and the kinematics and dynamics of the OMR-SOW are presented. Section 3 explains how the CVT function is achieved in the OMR-SOW. Section 4 discusses construction of the OMR-SOW and some experimental results. Conclusions are drawn in Section 5.

2. Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels

In this section, a new type of omnidirectional mobile robot, an omnidirectional mobile robot with steerable omnidirectional wheels (OMR-SOW), is introduced. Since four wheels of a robot can be independently driven, the OMR-SOW is of 4 DOFs: 2 DOFs for translation, 1 DOF rotation and 1 DOF for steering. The steering DOF can function as a continuously variable transmission (CVT). In the following subsections, steerable omnidirectional wheels are introduced and the features of the OMR-SOW are discussed in detail.

2.1 Steerable Omnidirectional Wheels

Nontrivial wheeled mobile robots are classified into five categories according to degree of mobility and degree of steerability (Campion et al., 1996). They did not mention steerable omnidirectional wheels since most omnidirectional wheels did not have steering capability. However, steerable omnidirectional wheels have an additional DOF which can be used as a continuously variable transmission.

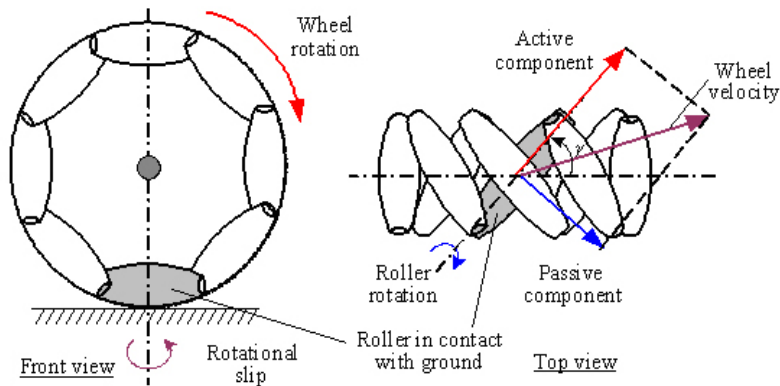


Fig. 2. DOFs in a general omnidirectional wheel.

An omnidirectional wheel has 3 DOFs composed of wheel rotation, roller rotation and rotational slip about the vertical axis passing through the point of contact (Muir & Neuman, 1987). Fig. 2 shows a typical omnidirectional wheel in which the roller axes have an inclination angle γ with the wheel plane. Note that the wheel shown in the figure represents a general omnidirectional wheel and several different wheel mechanisms are available depending on roller types and inclination angles (e.g., universal wheel (Blumrich, 1974) with $\gamma = 0^\circ$ and Mecanum wheel (Ilou, 1975) with $\gamma = 45^\circ$). In the omnidirectional wheel, the wheel velocity can be divided into the components in the active direction and in the passive direction. The active component is directed along the axis of the roller in contact with the ground, while the passive one is perpendicular to the roller axis.

In most cases, omnidirectional wheels are fixed relative to the robot body and do not rotate for steering since steering can be performed by a combination of wheel velocities in these types of mechanisms. Omnidirectional wheels, however, are able to be combined with the steering mechanism as shown in Fig. 3. Since the steering mechanism provides an additional

DOF, this type of steerable omnidirectional wheel module has one more DOF in addition to 3 DOFs defined in Fig. 2. In Fig. 3, the origin o represents the center of a robot and C the steering axis for the wheel shown. In the figure, ϕ is the steering angle, l is the offset distance, L_0 is the distance from the robot center to steering axis, and γ is the angle between the roller axis and the wheel plane, respectively.

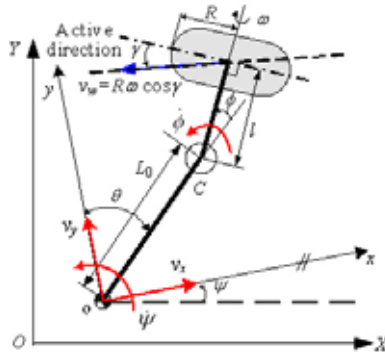


Fig. 3. Coordinate systems and parameters for a steerable omnidirectional wheel.

As mentioned earlier, a steerable omnidirectional wheel has 4 DOFs. Since roller rotation and rotational slip do not impose any constraint on the wheel motion, the constraints can be obtained by the relationship between the robot velocity and the active wheel velocity (i.e., wheel velocity in the active direction). The active wheel velocity is given by

$$v_w = R\omega \cos \gamma \quad (1)$$

where R is the wheel radius and ω is the angular velocity of a wheel.

Let the robot velocity vector be given as $v_r = [v_x, v_y, \dot{\psi}, \dot{\phi}]^T$, where v_x and v_y are the translational velocities of the robot center, $\dot{\psi}$ is the angular velocity about the robot center, and $\dot{\phi}$ is the derivative of the steering angle, respectively. If the velocity of a robot is given, the velocity of each wheel can be obtained as a function of steering angle by

$$\omega = [-v_x \cos(\theta - \phi - \gamma) + v_y \sin(\theta - \phi - \gamma) + \dot{\psi} \{l \cos \gamma + L_0 \cos(\gamma + \phi)\} + l \dot{\phi} \cos \gamma] / (R \cos \gamma) \quad (2)$$

The derivation of Eq. (2) can be referred to Song and Byun (2004). Note that a change in steering angle causes the relationship between the robot velocity and the wheel velocities to change, which enables a steerable omnidirectional wheel mechanism to function as a CVT.

Fig. 4 shows the continuous alternate wheel (CAW) (Byun & Song, 2003) which was used in the OMR-SOW. Note that this wheel has the same feature as a general omnidirectional wheel shown in Fig. 2, except for an inclination angle $\gamma = 0^\circ$. Many types of omnidirectional wheels with passive rollers have gaps between rollers. Since these gaps cause a wheel to make discontinuous contact with the ground, they lead to vertical and/or horizontal vibrations during wheel operation. However, the CAW makes continuous contact with the ground with alternating large and small rollers around the wheel, so virtually no vibration is created during operation.

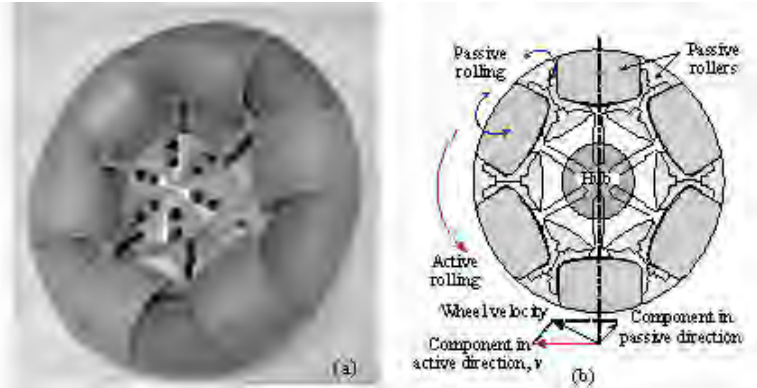


Fig. 4. Continuous alternate wheel; (a) photo and (b) active and passive rolling of CAW.

2.2 OMR-SOW with 4 Offset Steerable Omnidirectional Wheels

In this section, the structure and operational principle of the proposed OMR-SOW will be presented. The coordinate systems for the OMR-SOW are illustrated in Fig. 5. The frame $O-XY$ is assigned as the reference frame for robot motion in the plane, and the moving frame $o-xy$ is attached to the robot center. The angle θ between the y -axis and the diagonal line of the robot body depends on the shape of the body (i.e., $\theta = 45^\circ$ for a square body). The four wheel modules can rotate about each pivot point C_1, \dots, C_4 located at the corners of the robot body, but they are constrained to execute a synchronized steering motion of a single DOF by the mechanism composed of connecting links and a linear guide. In Fig. 5, the steering angle ϕ is defined as the angle from the zero position which coincides with the diagonal lines (i.e., C_1C_3 or C_2C_4) of the robot body. Although four wheel modules are steered at each steering axis, the steering angle ϕ for each wheel is identical. Note that steering is indirectly determined by the vector sum of each wheel velocity vector (not by an independent steering motor).

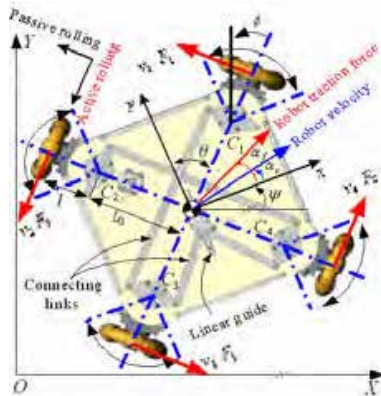


Fig. 5. Coordinate systems for OMR-SOW.

The CAW discussed in section 2.1 can perform differential drive as well as omnidirectional drive. That is, the steering angle is changed in the range of -30° to $+30^\circ$ in the omnidirectional drive mode, but maintained at $+45^\circ$ or -45° in the differential drive mode as shown in Fig. 6. Conventional wheels used in the differential drive have two advantages over omnidirectional wheels. First, the differential drive is generally more efficient in energy than the omnidirectional drive when the omnidirectional drive is not required (e.g., straight-line driving). Second, the conventional wheels can go over a higher bump than the omnidirectional wheels can, because the maximum height of a surmountable bump for the omnidirectional wheels is limited by the radius of its passive roller, which is much smaller than the radius of a conventional wheel.

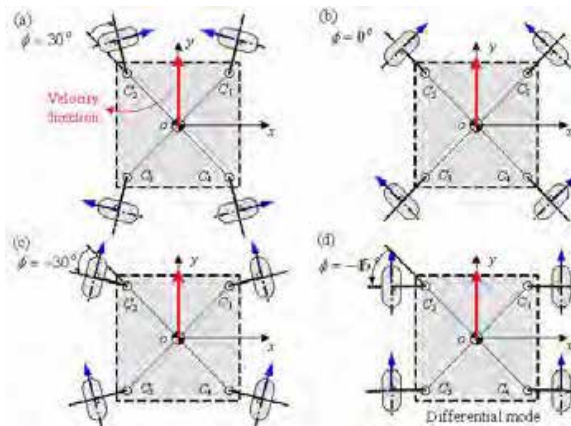


Fig. 6. Various wheel arrangements.

However, the differential drive mode cannot be set up by simply adjusting the steering angle of the omnidirectional wheels to $+45^\circ$ or -45° , because passive rollers cannot be constrained in this case. For example, in Fig. 6(d), pushing the robot in the x direction causes the robot to move in this direction, which does not happen in a robot with conventional wheels which resist sideways motion. That is, passive rolling must not be allowed in the differential drive mode.

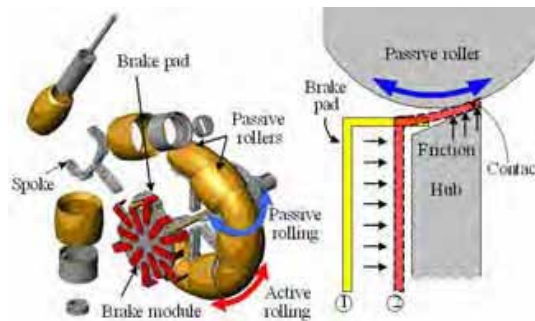


Fig. 7. Disassembled appearance of CAW with brake module.

2.3 Kinematic Analysis of OMR-SOW

The 4-wheeled omnidirectional mobile robot was illustrated in Fig. 5. Using Eq. (2), the relationship between the wheel velocity vector and the vehicle velocity vector can be given by

$$\begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{Bmatrix} = \begin{bmatrix} -C & S & L & l \\ -C & -S & L & -l \\ C & -S & L & l \\ C & S & L & -l \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ \dot{\psi} \\ \dot{\phi} \end{Bmatrix} \quad (3)$$

where $C = \cos(\theta - \phi)$, $S = \sin(\theta - \phi)$, $L = L_0 \cos \phi + l$, and v_1, v_2, v_3, v_4 are the wheel velocities in the active direction. The matrix in Eq. (3) is invertible, since $C \neq 0$ and $S \neq 0$ for $0 < \theta - \phi < 90^\circ$. The inverse matrix is obtained by

$$J = \frac{1}{4} \begin{bmatrix} -1/C & -1/C & 1/C & 1/C \\ 1/S & -1/S & -1/S & 1/S \\ 1/L & 1/L & 1/L & 1/L \\ 1/l & -1/l & 1/l & -1/l \end{bmatrix} \quad (4)$$

which corresponds to the Jacobian matrix relating the wheel velocity vector to the robot velocity vector as follows:

$$V_w = J^{-1}V_r \text{ or } V_r = JV_w \quad (5)$$

where $V_w = [v_1 \ v_2 \ v_3 \ v_4]^T$ and $V_r = [v_x \ v_y \ \dot{\psi} \ \dot{\phi}]^T$. It follows from Eq. (5) that the robot velocity and steering velocity of the OMR-SOW can be completely determined by control of four independent motors driving each wheel.

To help understanding the operational principle of the OMR-SOW, let us consider the following example. Suppose that the parameters are $\theta = 45^\circ$, $\phi = -15^\circ$, $L_0 = 2$, $l = 1$, as shown in Fig. 8. If the robot velocity is given by $v_r = [2 \ 0 \ 0 \ 0]^T$, C, S and L become

$$C = \cos(45^\circ + 15^\circ) = 1/2 \quad S = \sin(45^\circ + 15^\circ) = \sqrt{3}/2 \quad L = L_0 \cos(-15^\circ) + l = 2.932 \quad (6)$$

Then, Eq. (5) can be calculated as

$$V_w = J^{-1}V_r = \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{Bmatrix} = \begin{bmatrix} -1/2 & \sqrt{3}/2 & 2.932 & 1 \\ -1/2 & -\sqrt{3}/2 & 2.932 & -1 \\ 1/2 & -\sqrt{3}/2 & 2.932 & 1 \\ 1/2 & \sqrt{3}/2 & 2.932 & -1 \end{bmatrix} \begin{Bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{Bmatrix} \quad (7)$$

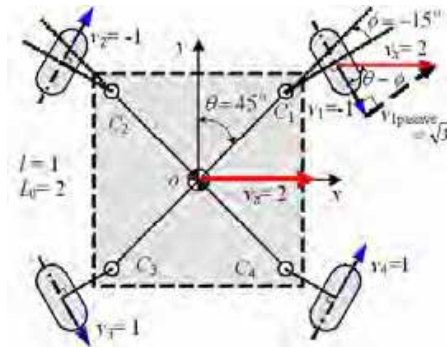


Fig. 8. Example in kinematics.

Fig. 8 shows the active wheel velocities to produce the desired robot velocity. Note that the desired robot velocity contains only the translational velocity in the x direction, so the resultant velocity of each wheel has the magnitude of 2 in the x direction. From this observation, it follows that the passive wheel velocity of wheel 1 has the magnitude of $\sqrt{3}$ to form the specified resultant wheel velocity, as shown in Fig. 8.

2.4 Dynamic Analysis of OMR-SOW

Consider the dynamic model of a robot shown in Fig. 5. The robot motion on the fixed coordinate system XY is described by

$$M_r \dot{V}_R = F_R \quad (8)$$

$$\text{where } M_r = \begin{bmatrix} M & 0 & 0 & 0 \\ 0 & M & 0 & 0 \\ 0 & 0 & I_z & 0 \\ 0 & 0 & 0 & I_\phi \end{bmatrix}, \quad V_R = \begin{bmatrix} v_X \\ v_Y \\ \dot{\psi} \\ \dot{\phi} \end{bmatrix}, \quad \text{and } F_R = \begin{bmatrix} F_X \\ F_Y \\ T_z \\ T_\phi \end{bmatrix},$$

where M is the mass of a robot, I_z the moment of inertia about the z axis passing through the robot center and I_ϕ the moment of inertia about the steering axis of the wheel modules. Note that the subscript R represents the fixed reference frame.

The transformation matrix from the moving robot frame $\{r\}$ to the fixed reference frame $\{R\}$ is given by

$$R = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

The relationship between the fixed reference frame and the moving robot frame is described by

$$V_R = R \cdot V_r, \quad F_R = R \cdot F_r \quad (10)$$

Differentiation of V_R is given by

$$\dot{V}_R = R \dot{V}_r + \dot{R} V_r \quad (11)$$

Therefore, the robot of Eq. (8) is described on the moving coordinate system by

$$M_r (R \dot{V}_r + \dot{R} V_r) = R F_r \quad \text{or} \quad R^{-1} M_r (R \dot{V}_r + \dot{R} V_r) = F_r \quad (12)$$

On the other hand, the force and moment of a robot can be expressed from the geometry in Fig. 5 by

$$F_x = -CF_1 - CF_2 + CF_3 + CF_4, \quad F_y = SF_1 - SF_2 - SF_3 + SF_4 \quad (13a)$$

$$T_z = LF_1 + LF_2 + LF_3 + LF_4, \quad T_\phi = IF_1 - IF_2 + IF_3 - IF_4 \quad (13b)$$

where F_x and F_y are the forces acting on the robot center in the x and y directions, T_z is the moment about the z axis passing through the robot center, and T_ϕ is the torque required to

rotate the wheel modules, respectively. Note that the force F_i ($i = 1, \dots, 4$) is the traction force acting on the wheel in the direction of active rolling. Using the Jacobian matrix defined in Eq. (4), the relationship between the wheel traction forces and the resultant forces acting on the robot body is given by

$$\mathbf{F}_r = \mathbf{J}^{-T} \mathbf{F}_w \text{ or } \mathbf{F}_w = \mathbf{J}^T \mathbf{F}_r \quad (14)$$

where $\mathbf{F}_w = [F_1 \ F_2 \ F_3 \ F_4]^T$ and $\mathbf{F}_r = [F_x \ F_y \ T_z \ T_\phi]^T$. It is noted that \mathbf{F}_r is given by a vectorial sum of traction forces. Varying a combination of the traction forces can generate arbitrary forces and moments for driving the vehicle and the moment for steering the wheel modules.

In addition, wheel forces are given by

$$R\mathbf{F}_w = \mathbf{U} - I_w \dot{\boldsymbol{\omega}}_w - c_w \boldsymbol{\omega}_w \text{ or } R\mathbf{F}_w = \mathbf{U} - \frac{I_w}{R} \dot{\mathbf{V}}_w - \frac{c_w}{R} \mathbf{V}_w \quad (15)$$

where R is the wheel radius, $\mathbf{U} = [u_1 \ u_2 \ u_3 \ u_4]^T$, where u_i is the motor torque of the i -th motor, I_w is the moment of inertia of the wheel about the drive axis, and c_w is the viscous friction factor of the wheel, and $\boldsymbol{\omega}_w = [\omega_1 \ \omega_2 \ \omega_3 \ \omega_4]^T$, where ω_i is the angular velocity of the i -th wheel. From Eq. (5), the wheel velocity and acceleration vectors are obtained by

$$\mathbf{V}_w = \mathbf{J}^{-1} \mathbf{V}_r, \quad \dot{\mathbf{V}}_w = \dot{\mathbf{J}}^{-1} \mathbf{V}_r + \mathbf{J}^{-1} \dot{\mathbf{V}}_r \quad (16)$$

After substitution of Eq. (14), (15) and (16) into (12), the following relation is obtained

$$\mathbf{R}^{-1} \mathbf{M}_r (\mathbf{R} \dot{\mathbf{V}}_r + \dot{\mathbf{R}} \mathbf{V}_r) = \mathbf{J}^{-T} \mathbf{F}_w = \mathbf{J}^{-T} \left(\frac{1}{R} \mathbf{U} - \frac{I_w}{R^2} (\dot{\mathbf{J}}^{-1} \mathbf{V}_r + \mathbf{J}^{-1} \dot{\mathbf{V}}_r) - \frac{c_w}{R^2} \mathbf{J}^{-1} \mathbf{V}_r \right) \quad (17)$$

This can be simplified by use of the relation $\mathbf{R}^{-1} \mathbf{M}_r \mathbf{R} = \mathbf{M}_r$ to

$$(\mathbf{R} \mathbf{J}^T \mathbf{M}_r + \frac{I_w}{R} \mathbf{J}^{-1}) \dot{\mathbf{V}}_r + (\mathbf{R} \mathbf{J}^T \mathbf{R}^{-1} \mathbf{M}_r \dot{\mathbf{R}} + \frac{I_w}{R} \dot{\mathbf{J}}^{-1} + \frac{c_w}{R} \mathbf{J}^{-1}) \mathbf{V}_r = \mathbf{U} \quad (18)$$

Eq. (18) represents the dynamic model of a robot.

3. CVT of OMR-SOW

As explained in Section 2, a change in the steering angle of OMR-SOW functions as a CVT. The CVT of an automobile can keep the engine running within the optimal range with respect to fuel efficiency or performance. Using the engine efficiency data, the CVT controls the engine operating points under various vehicle conditions. A CVT control algorithm for the OMR-SOW ought to include the effects of all four motors. A simple and effective algorithm for control of the CVT is proposed based on the analysis of the operating points of a motor.

3.1 Velocity and Force Ratios

Since the omnidirectional mobile robot has 3 DOFs in the 2-D plane, it is difficult to define the velocity ratio in terms of scalar velocities. Thus the velocity ratio is defined using the concept of norms as follows:

$$r_v = \frac{\|V_r\|}{\|V_w\|} = \frac{\|J V_w\|}{\|V_w\|} \quad (19)$$

Note that the velocity ratio for the identical wheel velocities varies depending on the steering angle. For example, suppose that a robot has a translational motion in the x axis. The robot velocity is then given by

$$V_r = [1 \ 0 \ 0 \ 0]^T \quad (20)$$

From Eq. (4) and (5), the wheel velocity is computed as

$$v_w = [-c \ -c \ c \ c]^T \quad (21)$$

From Eq. (19), the velocity ratio is obtained as

$$r_v = \frac{1}{2c} = \frac{1}{2\cos(\theta - \phi)} \quad (22)$$

Similarly, if a robot has a translational motion in the y axis, (i.e., $v_r = [0 \ 1 \ 0 \ 0]^T$), then the velocity ratio is given by

$$r_v = \frac{1}{2s} = \frac{1}{2\sin(\theta - \phi)} \quad (23)$$

For rotational motion given by $v_r = [0 \ 0 \ 1 \ 0]^T$, the velocity ratio is

$$r_v = \frac{1}{2L} = \frac{1}{2(L_0 \cos\phi + l)} \quad (24)$$

Figure 9 shows the velocity ratio profiles as a function of steering angle in case of $L_0 = 0.283\text{m}$, $l = 0.19\text{m}$, and $\theta = 45^\circ$. It is observed that the translational velocity ratios vary significantly in the range between 0.5 and infinity, while the rotational velocity ratio is kept nearly constant.

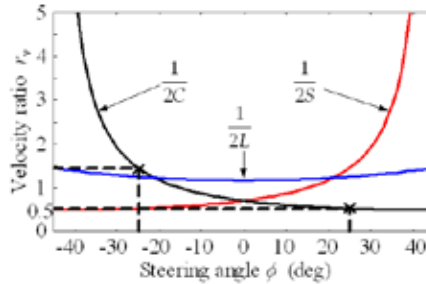


Fig. 9. Velocity ratios as a function of steering angle.

The force ratio of the force acting on the robot center to the wheel traction force can be defined in the same way as the velocity ratio in Eq. (19).

$$r_f = \frac{\|F_v\|}{\|F_w\|} = \frac{\|J^{-T} F_w\|}{\|F_w\|} = \frac{1}{r_v} \quad (25)$$

Note that the force ratio corresponds to the inverse of the velocity ratio.

3.2 Motion Control of OMR-SOW

The motion of a mobile robot can be controlled by wheel velocities. From Eq. (5), when the desired robot motion V_{rd} is given, the reference wheel velocity V_{wd} of each wheel can be computed by

$$V_{wd} = J^{-1}V_{rd} \tag{26}$$

As shown in Fig. 10 representing the block diagram of the control system for OMR-SOW, when the reference wheel velocity $V_{wd} = [v_{1d} \ v_{2d} \ v_{3d} \ v_{4d}]^T$ is given to each motor, the PI controller performs velocity control of each motor to generate the control signal u_i ($i = 1, \dots, 4$). If each wheel is controlled to follow the reference wheel velocity, then the robot can achieve the desired motion. Practically, all mobile robots have slip between the wheels and the ground to some extent. This slip causes the real motion to be different from the desired one. Since the robot does not have any sensor measuring the robot velocity, this error is somewhat inevitable.

Since four wheels are independently controlled in the OMR-SOW, a steering angle can be arbitrarily selected while the desired robot velocity (i.e., 2 translational DOF s and 1 rotational DOF) is achieved. That is, a wide range of steering angles can lead to the identical robot velocity. The steering control algorithm then determines the desired steering angle ϕ_d to achieve the maximum energy efficiency for the given robot velocity. Therefore, the desired steering velocity is computed by

$$\dot{\phi}_d = K_\phi(\phi_d - \phi) \tag{27}$$

where K_ϕ is the control gain of steering and ϕ is the actual steering angle measured by the encoder installed on one of the steering axes.

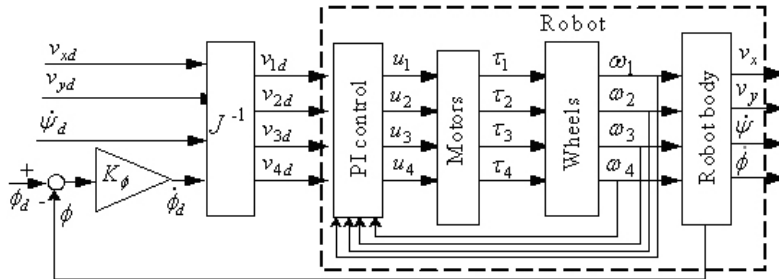


Fig. 10. Control system of OMR-SOW with steering angle control.

Figure 11 shows the operating points of a motor used in the mobile robot. In the figure T_{max} is the maximum continuous torque, and ω_{max} is the maximum permissible angular velocity. The solid lines represent the constant efficiency and the dashed lines denote the constant output power. The input power is obtained by the product of the input current and voltage, whereas the output power is measured by the product of the motor angular velocity and torque. The efficiency η is the ratio of the output to input power.

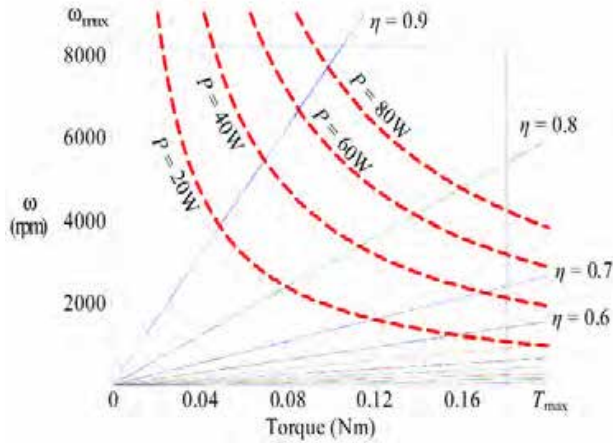


Fig. 11. Operating range of a motor.

As shown in this figure, the efficiency varies as the operating point moves on the constant output power line. The operating point of a motor can be varied by the CVT. For the same output power, a reduction in the force ratio of CVT leads to a decrease in velocity and an increase in torque, and in turn a decrease in efficiency. Therefore, the CVT should be controlled so that motors operate in the region of high velocity and low torque.

3.3 Steering Control Algorithm

As explained in Section 3.2, when the desired robot velocity \mathbf{V}_{rd} is given, each wheel is independently controlled. Any robot velocity can be achieved for a wide range of steering angles, but some steering angles can provide better energy efficiency than others. In this section, the steering control algorithm is proposed to determine such a steering angle that can result in maximum energy efficiency.

In Fig. 11, velocity is controlled by each motor controller. The current sensor at each motor drive measures the motor current and computes the motor torque $\tau = [\tau_1 \ \tau_2 \ \tau_3 \ \tau_4]^T$. The wheel traction force \mathbf{F}_w can then be computed by

$$\mathbf{F}_w = (\tau - I_w \dot{\omega}_w - c_w \omega_w) / r \quad (28)$$

where r is the wheel radius, I_w the moment of inertia of the wheel about the wheel axis, c_w the viscous friction factor of the wheel, and $\omega_w = [\omega_1 \ \omega_2 \ \omega_3 \ \omega_4]^T$ is the wheel angular velocity. By substituting (28) into (14), the robot traction force \mathbf{F}_r can be obtained by

$$\mathbf{F}_r = [\mathbf{F}_x \ \mathbf{F}_y \ \mathbf{T}_z \ \mathbf{T}_\phi]^T = \mathbf{J}^{-T} \mathbf{F}_w \quad (29)$$

In Eq. (29), the torque T_ϕ required to steer a wheel module is independent of the steering angles. Since the force ratio associated with rotation has little relation with steering angles, it is governed mostly by translational motions. The robot traction force direction α_f can then be given by

$$\alpha_f = \text{atan2}(F_x, F_y) \quad (30)$$

Figure 12 shows the force ratio r_f defined in (25) in terms of the robot traction force angle α_f and the steering angle ϕ . Identical wheel traction forces can generate substantially different robot traction forces depending on ϕ . The OMR-SOW capable of CVT has the maximum energy efficiency in the region with the highest force ratio (i.e., high velocity and low torque). For example, when $\alpha_f = 90^\circ$, the steering angle of -30° can generate maximum energy efficiency in the omnidirectional drive mode. In Fig. 13, curve 1 is obtained by connecting the steering angles corresponding to the maximum force ratio for each robot traction force angle α_f . However, a rapid change in steering angle from $+30^\circ$ to -30° is required to maintain the maximum force ratio around $\alpha_f = 90^\circ.n + 45^\circ$ ($n = 0, 1, \dots$). Such discontinuity in the steering angle due to a small change in traction force direction is not desirable. To overcome this problem, a sinusoidal profile (curve 2) is practically employed in control of the OMR-SOW.

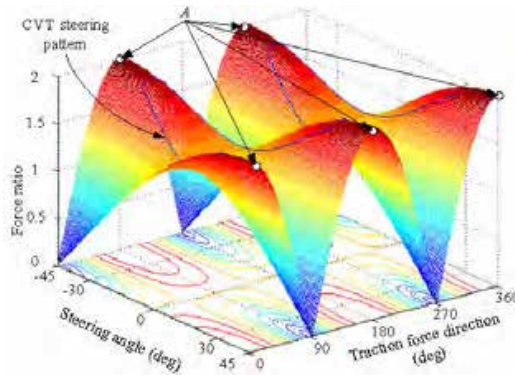


Fig. 12. Force ratio as a function of steering angle and force direction.

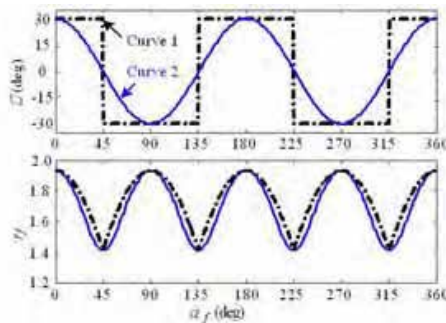


Fig. 13. Steering angle curve corresponding to maximum force ratio as a function of robot traction force angle.

If the steering angle ϕ is set to either $+45^\circ$ or -45° as shown in Fig. 6(d), the OMR-SOW can be driven in the differential drive mode. In this mode, the OMR-SOW has the maximum force ratio denoted as A as shown in Fig. 12, which leads to the higher energy efficiency

than in the omnidirectional drive mode. In conclusion, if the CVT is controlled in consideration of the steering pattern for each driving condition, the energy efficient driving can be achieved. However, the change from the omnidirectional to the differential drive mode cannot be conducted while the robot is moving, because the steering angle greater than $\pm 30^\circ$ usually brings about slip between the wheel and ground, and the passive rollers cannot be controlled. Hence, the robot should stop temporarily to conduct this conversion.

4. Experiments and Discussions

4.1 Construction of OMR-SOW

The Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels (OMR-SOW) developed, as shown in Fig. 1. This robot contains four wheel modules comprising the omnidirectional wheel connected to each motor, a synchronous steering mechanism, and a square platform whose side is 500mm. The height of the platform from the ground is 420mm. The robot can be used as a wheelchair since it was designed to carry a payload of more than 100kg. The drive mechanism uses four DC servo motors (150W), which are controlled by the DSP-based motor controllers having a sampling period of 1ms. As shown in Fig. 14, the DSP-based master controller performs kinematic analysis, plans the robot trajectory, and delivers the velocity commands to each wheel. The robot can move autonomously, but the PC is used to monitor the whole system, collect data, and display the robot's states. The suspension system, composed of a 4 bar linkage, a damper and a spring, is required to ensure that all wheels are in contact with the ground at all times, which is very important in this type of four-wheeled mechanism. This suspension can also absorb the shock transmitted to the wheels.

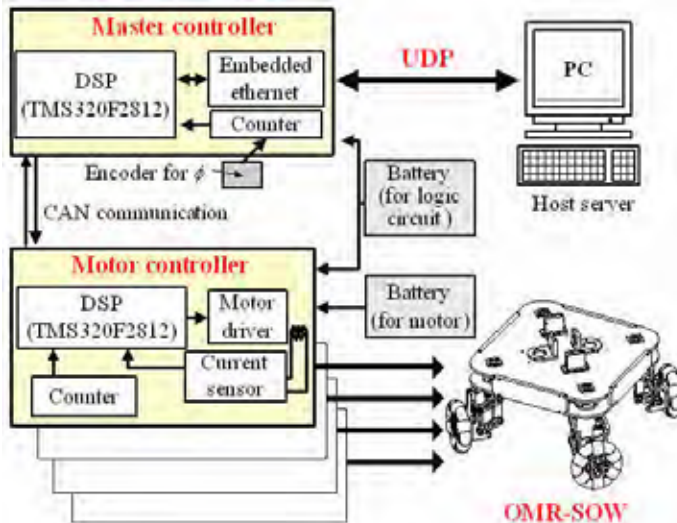


Fig. 14. Control systems for OMR-SOW.

4.2 Experiments of OMR-SOW

Some performance tests for the prototype vehicle have been conducted. Tracking performance of the vehicle with one person on it has been tested for various trajectories. Fig. 15 shows experimental results for a square trajectory. The vehicle control algorithm generates the required vehicle velocity and then computes the velocity of each wheel to achieve the desired motion through the Jacobian analysis given in Eq. (5). In Fig. 15(a), the solid line represents the actual trajectory of the robot and the dashed line is the reference trajectory. Triangles in the figure represent the position and orientation of the robot in every second and the triangle filled with gray color is the start location. Fig. 15(b) shows robot velocity and steering angle. The robot velocity follows the reference input faithfully. Since the accumulated position error is not compensated for, however, there exists a position error between the reference and actual trajectory. Fig. 15(c) shows each wheel velocity and motor currents.

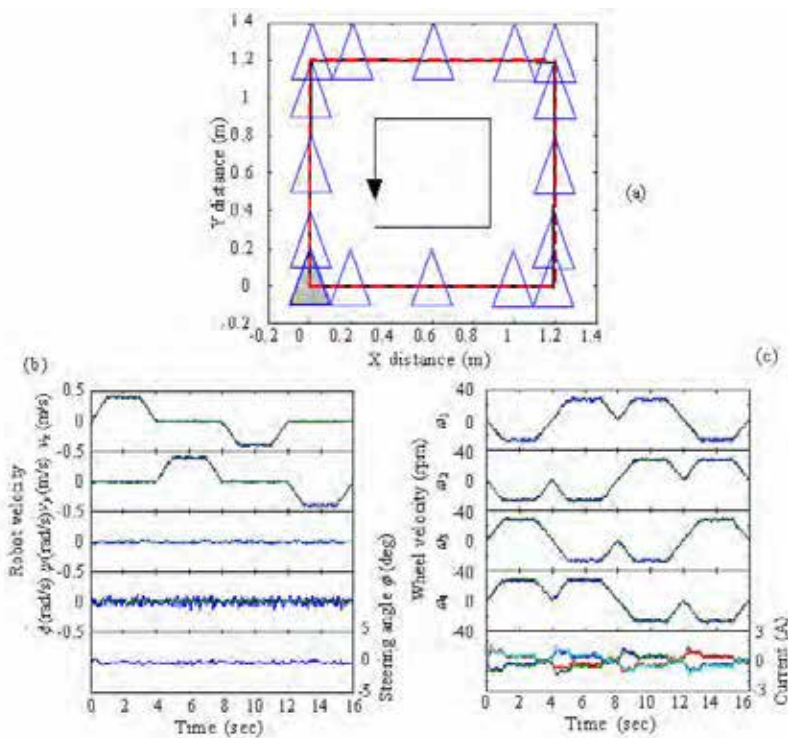


Fig. 15. Experimental results of tracking performance for a rectangular trajectory.

Experiments of Fig. 15 are associated with only translational motions. However, Fig. 16 showing tracking performance for a circular trajectory is associated with both translational and rotational motion. In the experiment, the robot moves in the x -direction and simultaneously rotates about the z -axis. It is seen that the actual trajectory represented in the

solid line tracks the reference reasonably relatively well. Some error is observed around the finish since the prototype vehicle does not implement any position control algorithm for this test and thus the position error has been accumulated during motion.

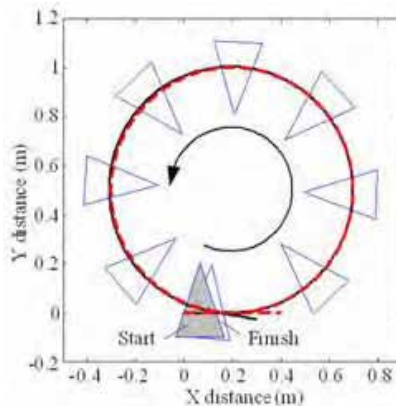


Fig. 16. Experimental results of tracking performance for a circular trajectory.

In the next experiment, a half of the square trajectory existed on a ramp whose slope was 10° as shown in Fig. 17. To follow a given velocity command, the motors should generate much more torque in the ramp than in the ground, so the current is increased. Therefore, the measured current indirectly gives information on the ground conditions or disturbances. Even for a ramp or disturbance, the steering control algorithm based on the measured current can select proper steering angles. The consumed energy was measured as 767.5J for the fixed angle and 653.4J for the case of the steering algorithm, thus showing 14% reduction in energy.

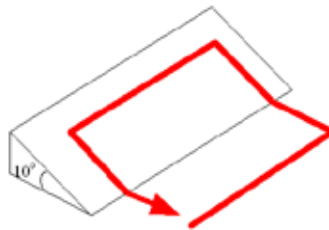


Fig. 17. Square trajectory with ramp.

Next, energy consumption according to the wheel arrangement was investigated. The robot traveled at a speed of 0.05m/s in the y -axis in Fig. 6. This motion could be achieved in various wheel arrangements. Among them, 4 configurations were chosen including 3 omnidirectional drive modes and 1 differential drive mode (see Fig. 6). The experimental results are summarized in Table 1. As expected, the differential drive provided better energy efficiency than the omnidirectional drives. This result justifies the proposed mechanism capable of conversion between the omnidirectional and the differential drive mode depending on the drive conditions.

Experiments	ϕ	Average current (A)	Power (W)	Energy (J)
(a)	30	0.385	9.246	924.6
(b)	0	0.296	7.112	711.2
(c)	-30	0.275	6.605	660.5
(d)	-45	0.266	6.402	640.3

Table 1. Comparison of omnidirectional drive with differential drive.

Conventional wheels used in automobiles usually show better performance than the omnidirectional wheels with passive rollers. This is because the height of a surmountable bump for the omnidirectional wheels is limited by the radius of the smallest passive roller and the friction force of the roller. Thus if the passive rollers are constrained not to rotate as in the differential drive mode, even omnidirectional wheels can function as conventional ones. The omnidirectional wheel can go over a 5cm high bump, which is greater than the radius of the passive roller.

5. Conclusions

In this chapter, an omnidirectional mobile robot with steerable omnidirectional wheels (OMR-SOW) has been proposed and the kinematic and dynamic analysis of a proposed robot has been conducted. The motion control system of a robot was developed and various experiments were conducted. As a result of this research, the following conclusions are drawn.

1. The OMR-SOW has 4 DOFs which consist of 3 DOFs for omnidirectional motion and 1 DOF for steering. This steering (DOF) functions as a continuously variable transmission (CVT). Therefore, the OMR-SOW can be also considered as an omnidirectional mobile robot with CVT.
2. The proposed steering control algorithm for CVT can provide a significant reduction in driving energy than the algorithm using a fixed steering angle. Therefore, the size of an actuator to meet the specified performance can be reduced or the performance such as gradability of the mobile robot can be enhanced for given actuators.
3. Energy efficiency can be further improved by selecting the differential drive mode through the adjustment of OMR-SOW wheel arrangement. The surmountable bump in the differential drive mode is much higher than that in the omnidirectional drive mode.

One of the most important features of the OMR-SOW is the CVT function which can provide energy efficient drive of a robot. If the CVT is not properly controlled, however, energy efficiency capability can be deteriorated. Hence, research on the proper control algorithm is under way for energy efficient drive.

6. References

- Asama, H.; Sato, M., Bogoni, L., Kaetsu, H., Masumoto, A. & Endo, I. (1995). Development of an Omnidirectional Mobile Robot with 3 DOF Decoupling Drive Mechanism, *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1925-1930.
- Blumrich, J. F. (1974). Omnidirectional vehicle, United States Patent 3,789,947.
- Byun, K.-S. & Song, J.-B. (2003). Design and Construction of Continuous Alternate Wheels for an Omni-Directional Mobile Robot, *Journal of Robotic Systems*, Vol. 20, No. 9, pp. 569-579.

- Carlisle, B. (1983). An Omnidirectional Mobile Robot, *Development in Robotics*, Kempston, pp. 79-87.
- Campion, G.; Bastin, G. & D'Andrea-Novet, B. (1996). Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robot, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 1, pp. 47-62.
- Ilou, B. E. (1975). Wheels for a course stable self-propelling vehicle movable in any desired direction on the ground or some other base, United States Patent 3,876,255.
- Muir, P. & Neuman, C. (1987). Kinematic Modeling of Wheeled Mobile Robots, *Journal of Robotic Systems*, Vol. 4, No. 2, pp. 281-340.
- Pin, F. & Killough, S. (1999). A New Family of Omnidirectional and Holonomic Wheeled Platforms for Mobile Robot, *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 6, pp. 978-989.
- Song, J.-B. & Byun, K.-S. (2004) Design and Control of a Four-Wheeled Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels, *Journal of Robotic Systems*, Vol. 21, No. 4, pp. 193-208
- Tahboub, K. & Asada, H. (2000). Dynamic Analysis and Control of a Holonomic Vehicle with Continuously Variable Transmission, *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2466-2472.
- Wada, M. & Mory, S. (1996). Holonomic and omnidirectional vehicle with conventional tires, *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 3671-3676.
- Wada, M. & Asada, H. (1999). Design and Control of a Variable Footprint Mechanism for Holonomic Omnidirectional Vehicles and Its Application to Wheelchairs, *IEEE Trans. on Robotics and Automation*, Vol. 15, No. 6, pp. 978-989.
- West, M. & Asada, H. (1997). Design of ball wheel mechanisms for omnidirectional vehicles with full mobility and invariant kinematics, *Journal of Mechanical Design*, pp. 119-161.

Dynamic Model, Control and Simulation of Cooperative Robots: A Case Study

Jorge Gudiño-Lau* & Marco A. Arteaga**

* *Facultad de Ingeniería Electromecánica
Universidad de Colima
Mexico*

** *Departamento de Control y Robótica
División de Ingeniería Eléctrica, de la Facultad de Ingeniería
Universidad Nacional Autónoma de México
México*

1. Introduction

The modelling and control of multiple robotic manipulators handling a constrained object requires more sophisticated techniques compared with a single robot working alone. Since the theory employed for cooperative robots is independent of their size, one can think of them as mechanical hands. The study of mechanical hands is important not only because these can be used as prosthetic devices for humans, but also because they increase considerably the manipulation capacity of a robot when substituting the usual gripper. Thus, robot hands (as well as cooperative robots), may find many areas of application nowadays. Many benefits can be obtained by using them in industrial manufacturing. A typical example is in a flexible assembly, where the robots join two parts into a product. Cooperative manipulators can also be used in material handling, *e.g.*, transporting objects beyond the load carrying capacity of a single robot. Furthermore, their employment allows to improve the quality of tasks in the manufacturer industry that require of great precision. On the other hand, cooperative robots are indispensable for skillful grasping and dexterous manipulation of objects. However, the literature about experimental results on the modeling, simulation and control of systems of multiple manipulators holding a common object is rather sparse.

A dynamic analysis for a system of multiple manipulators is presented in Orin and Oh (Orin & Oh 1981), where the formalism of Newton-Euler for open chain mechanisms is extended for closed chain systems. Another approach widely used is the Euler-Lagrange method (Naniwa *et al.* 1997). The equations of motion for each manipulator arm are developed in the Cartesian space and the impact of the closed chain is investigated when the held object is in contact with a rigid environment, for example the ground. Another general approach to obtain the dynamic model of a system of multiple robots is based on the estimation of the grasping matrix (Cole *et al.* 1992, Kuc *et al.* 1994, Liu *et al.* 2002, Murray *et al.* 1994, Yoshikawa & Zheng 1991). Here, the grasping matrix is used to couple the manipulators dynamics with that of the object, while this is

modeled by the Newton-Euler formulation. The dynamic analysis for cooperative robots with flexible joints holding a rigid object is presented in Jankowski *et al.* (Jankowski *et al.* 1993).

This work presents the study of the dynamic equations of a cooperative robot system holding a rigid object without friction. The test bed is made up of two industrial robots and it is at the Laboratory for Robotics of the National University of Mexico. The dynamic model for the manipulators is obtained independently from each other with the Lagrangian approach. Once the robots are holding the object, their joint variables are kinematically and dynamically coupled. Assuming that the coupling of the system is described by holonomic constraints, the manipulators and object equations of motion are combined to obtain the dynamic model of the whole system, which can be used for simulation purposes. It is important to stress that a robot manipulator in free motion does not have geometric constraints; therefore, the dynamic model is described by Ordinary Differential Equations (ODE). When working with constrained motion, the dynamic model is described by Differential Algebraic Equations (DAE). It is shown how the simulation of this kind of systems can be carried out, including a general approach to simulate contact forces by solving DAE's.

Early attempts to establish a relationship between the automatic control of robots carrying out a shared task are referred to Khatib's operational space formulation (Khatib 1987). During the 1980's, the most important research results considered the contact evolution during manipulation (Montana 1988). Such a contact evolution requires a perfect combination of position and force control. Some of the first approaches following this objective are presented in Ly and Sastry (Li & Sastry 1989) and Cole (Cole 1990). In those works, the dynamics of the object is considered explicitly. In Parra-Vega and Arimoto (Parra-Vega & Arimoto 1996), Liu *et al.* (Liu *et al.* 1997) and Parra-Vega *et al.* (Parra-Vega *et al.* 2001), control schemes which do not take into account the dynamics of the object but rather the motion constraints are designed. These control approaches have the advantage that they do not require an exact knowledge of the system model parameters, since an adaptive approach is introduced. More recently, Schlegl *et al.* (Schlegl *et al.* 2001) show some advances on hybrid (in terms of a combination of continuous and discrete systems) control approaches.

Despite the fact that Mason and Salisbury (Mason & Salisbury 1985) proposed the base of sensor-less manipulation in the 1980's, there are few control algorithms for cooperative robot systems which take into account the possible lack of velocity measurements. Perhaps because, since a digital computer is usually employed to implement a control law, a good approximation of the velocity vector can be obtained by means of numerical differentiation. However, recent experimental results have shown that a (digitalized) observer in a control law performs better (Arteaga & Kelly 2004). Thus, in Gudiño-Lau *et al.* (Gudiño-Lau *et al.* 2004) a decentralized control algorithm for cooperative manipulators (or robot hands) which achieves asymptotic stability of tracking of desired positions and forces by using an observer is given. In this work, a new control law based on a force filter is presented. This is a general control law, so that it can be applied to a system with more than two manipulators involved as well. The control scheme is of a decentralized architecture, so that the input torque for each robot is calculated in its own joint space and takes into account motion constraints rather than the held object dynamics. Also, an observer is employed to avoid velocity measurements and experimental results are presented to validate the theoretical results.

2. Experimental System

The system under study is made up of two industrial robots and it is at the Laboratory for Robotics of the National University of Mexico (Figure 1). They are the A465 and A255 of *CRS Robotics*. Even though the first one has six degrees of freedom and the second one five, only the first three joints of each manipulator are used in this case, while the rest of them are mechanically braked. Each joint is actuated by a direct current motor with optical encoders. Both manipulators have a crash protection device in the end effector and a force sensor installed on it; an aluminum finger is mounted on the sensor. The object is constituted by a melamine plastic box with dimensions $0.15\text{m} \times 0.15\text{m} \times 0.311\text{m}$ and weight 0.400kg . The experiments are performed in a Pentium IV to 1.4 GHz personal computer with two PCI-FlexMotion-6C boards of *National Instruments*. The sampling time is of 9ms. Controllers are programmed in the *LabWindows/CVI* software of *National Instruments*.

A schematic diagram of the robots holding an object is depicted in Figure 2. The system variables are the generalized coordinates, velocities, and accelerations, as well as the contact forces exerted by the end effector on the common rigid object, and the generalized input forces (*i.e.*, torques) acting on the joints.



Fig. 1. Robots A465 and A255 of *CRS Robotics*.

To describe the kinematic relationships between the robots and the object, a stationary coordinate frame C_0 attached to the ground serves as reference frame, as shown in Figure 2. An object coordinate frame C_2 is attached at the center of mass of the rigid object. The origin of the coordinate frame C_1 is located at the center of the end effector of robot A465. In the same way, the origin of the coordinate frame C_3 is located at the

center of the end effector of robot A255. The coordinate frame C_0 has been considered to be the inertial frame of the whole system. 0p_2 is the position vector of the object center of mass expressed in the coordinate system C_0 . 0p_1 and 0p_3 are vectors that describe the position of the contact points between the end effectors of robots A465, A255 and the object, respectively, expressed in the coordinate system C_0 (Gudiño-Lau & Artega 2005).

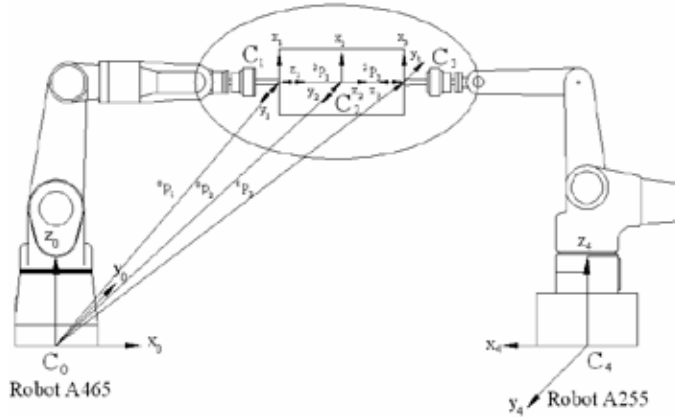


Fig. 2. Schematic diagram of robots holding an object.

3. The Cooperative Robots Dynamic Model

Consider the cooperative system with two robot arms shown in Figure 1, each of them with $n_i = 3$ degrees of freedom and $m_i = 1$ constraints arising from the contact with the held object. Then, the total number of degrees of freedom is given by $n = \sum_{i=1}^2 n_i$ with a total number of $m = \sum_{i=1}^2 m_i$ constraints.

3.1 Dynamic model with constraint motion and properties

The dynamic model for each individual manipulator, $i=1,2$, is obtained by the Lagrange's formulation as (Parra-Vega *et al.* 2001)

$$H_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + D_i\dot{q}_i + g_i(q_i) = \tau_i + J_{\phi_i}^T(q_i)\lambda_i \quad (1)$$

where $q_i \in \mathbb{R}^{n_i}$ is the vector of generalized joint coordinates, $H_i(q_i) \in \mathbb{R}^{n_i \times n_i}$ is the symmetric positive definite inertia matrix, $C_i(q_i, \dot{q}_i) \in \mathbb{R}^{n_i}$ is the vector of Coriolis and centrifugal torques, $g_i(q_i) \in \mathbb{R}^{n_i}$ is the vector of gravitational torques, $D_i \in \mathbb{R}^{n_i \times n_i}$ is the positive semidefinite diagonal matrix accounting for joint viscous friction coefficients, $\tau_i \in \mathbb{R}^{n_i}$ is the vector of generalized torques acting at the joints, and $\lambda_i \in \mathbb{R}^m$ is the vector of Lagrange multipliers (physically represents the force applied at the contact point). $J_{\phi_i}^T(q_i)\lambda_i$ represents the interaction of the rigid object with the two manipulators. $J_{\phi_i}(q_i) = \nabla \phi(q_i) \in \mathbb{R}^{m \times n_i}$ is assumed to be full rank in this paper. $\nabla \phi(q_i)$ denotes the gradient of the object surface vector $\phi \in \mathbb{R}^m$, which maps a vector onto the normal plane at the tangent plane that arises at the contact point described by

$$\varphi_i(q_i) = 0. \tag{2}$$

Equation (2) is a geometrical constraint expressed in an analytical equation in which only position is involved and that does not depend explicitly of time t . Constraints of this forms are known as *holonomic constraints* (they are also classified as *sclero-holonomic*).

Note that equation (2) means that homogeneous constraints are being considered (Parra-Vega *et al.* 2001). The complete system is subjected to 2 holonomic constraints given by

$$\varphi(q) = 0, \tag{3}$$

where $\varphi(q) = \varphi(q_1, q_2) \in \mathbb{R}^2$. This means that the object being manipulated and the environment are modeled by the constraint (3). If the holonomic constraints are correctly calculated, then the object will remain hold.

Let us denote the largest (smallest) eigenvalue of a matrix by $\lambda_{\max}(\cdot)$ ($\lambda_{\min}(\cdot)$). The norm of an $n \times 1$ vector x is defined by $\|x\| = \sqrt{x^T x}$ while the norm of an $m \times n$ matrix A is the corresponding induced norm $\|A\| = \sqrt{\lambda_{\max}(A^T A)}$. By recalling that revolute joints are considered, the following properties can be established (Liu *et al.* 1997, Arteaga Pérez 1998, Parra-Vega *et al.* 2001):

Property 3.1. Each $H_i(q_i)$ satisfies $\lambda_{H_i} \|x\|^2 \leq x^T H_i(q_i) x \leq \lambda_{H_i} \|x\|^2 \quad \forall q_i, \quad x \in \mathbb{R}^{n_i}$, where $\lambda_{H_i} \triangleq \min_{q_i \in \mathbb{R}^{n_i}} \lambda_{\min}(H_i)$, $\lambda_{h_i} \triangleq \max_{q_i \in \mathbb{R}^{n_i}} \lambda_{\max}(H_i)$ and $0 < \lambda_{h_i} \leq \lambda_{H_i} < \infty$. Δ

Property 3.2. With a proper definition of $C_i(q_i, \dot{q}_i)$, $\dot{H}_i(q_i) - 2C_i(q_i, \dot{q}_i)$ is skew-symmetric. Δ

Property 3.3. The vector $C_i(q_i, x)y$ satisfies $C_i(q_i, x)y = C_i(q_i, y)x \quad \forall x, y \in \mathbb{R}^{n_i}$. Δ

Property 3.4. It is satisfied $\|C_i(q_i, x)\| \leq k_{C_i} \|x\|$ with $0 < k_{C_i} < \infty, \forall x \in \mathbb{R}^{n_i}$. Δ

Property 3.5. The vector \hat{q}_i can be written as

$$\begin{aligned} \hat{q}_i &= \dot{q}_i + (J_{i0}^+ J_{i0} \dot{q}_i - J_{i0}^+ J_{i0} \dot{q}_i) \\ &= (J_{i0} - J_{i0}^+ J_{i0}) \dot{q}_i + J_{i0}^+ \dot{q}_i \\ &\triangleq Q_i(q_i) \dot{q}_i + J_{i0}^+(q_i) \dot{p}_i, \end{aligned} \tag{4}$$

where $J_{i0}^+ = J_{i0}^T (J_{i0} J_{i0}^T)^{-1} \in \mathbb{R}^{n_i \times m_i}$ stands for the Penrose's pseudoinverse and $Q_i \in \mathbb{R}^{n_i \times n_i}$ satisfies $\text{rank}(Q_i) = n_i - m_i$. These two matrices are orthogonal, i.e. $Q_i J_{i0}^+ = 0$ (and $Q_i J_{i0}^T = 0$). $\dot{p}_i \triangleq J_{i0} \dot{q}_i \in \mathbb{R}^{m_i}$ is the so called constrained velocity. Furthermore, in view of constraint (3), it holds

$$\sum_{i=1}^l \dot{p}_i = 0 \quad \text{and} \quad \sum_{i=1}^l p_i = \sum_{i=1}^l \int_0^t J_{i0} \dot{q}_i d\theta = 0. \tag{5}$$

Since homogeneous constraints are being considered, it also holds in view of (2) that

$$\dot{p}_i = 0 \quad \text{and} \quad p_i = 0. \tag{6}$$

for $i=1, \dots, l$, p_i is called the constrained position. Δ

As shown in Liu *et al.* (Liu *et al.* 1997), if we consider homogeneous holonomic constraints we can write the constrained position, constrained velocity and constrained acceleration as

$$\varphi_i(q_i) = 0 \tag{7}$$

$$\dot{\varphi}_i(q_i) = J_{\varphi i}(q_i) \dot{q}_i = 0 \tag{8}$$

$$\ddot{\varphi}_i(q_i) = J_{\varphi i}(q_i) \ddot{q}_i + \dot{J}_{\varphi i}(q_i) \dot{q}_i = 0, \tag{9}$$

respectively. Recall that in our case $n_i = 3$, $n = 6$, $m_i = 1$, and $m = 2, i=1,2$.

3.2 Dynamic model of the rigid object

The motion of the two robot arms is dynamically coupled by the generalized contact forces interacting through the common rigid object. To describe this interaction, it is necessary to know the object dynamics. According to the free body diagram of Figure 3, Newton's equation of motion are

$$m_o \ddot{x}_o - m_o g_o = f_1 - f_2, \tag{10}$$

where $m_o \in \mathbb{R}^{3 \times 3}$ is the diagonal mass matrix of the object, $\ddot{x}_o \in \mathbb{R}^3$ is the vector describing the translational acceleration of the center of mass of the rigid object, $f_1 \in \mathbb{R}^3$ and $f_2 \in \mathbb{R}^3$ are forces exerted by the robots, and $g_o \in \mathbb{R}^3$ is a gravity vector. All vectors are expressed with reference to the inertial coordinate frame C_0 . The contact forces vector are given by

$$f_i = n_i \lambda_i, \tag{11}$$

where $n_i \in \mathbb{R}^3$ represents the direction of the force (normal to the constraint) and $\lambda_i \in \mathbb{R}$ given in (1).

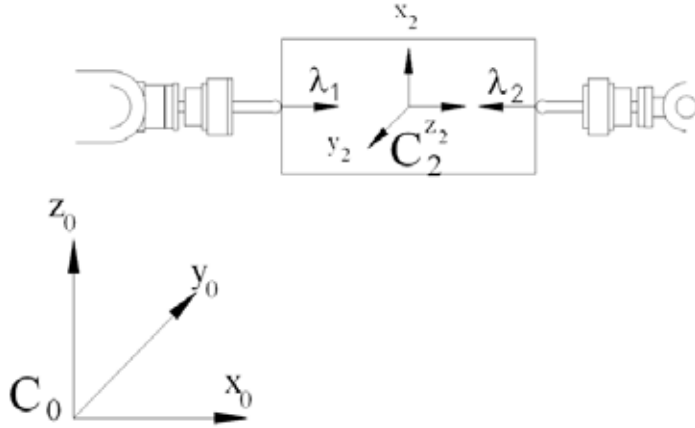


Fig. 3. Force free body diagram.

The following assumptions are made to obtain the dynamic model for the cooperative system and to design the control-observer scheme:

Assumption 3.1 *The end effectors (fingers) of the two robot arms are rigid.* Δ

Assumption 3.2 *The object is undeformable, and its absolute and relative position are known.* Δ

Assumption 3.3 *The kinematics of each robot is known* Δ

Assumption 3.4 *The l robots of which the system is made up satisfy constraints (2) and (6) for all time. Furthermore, none of the robots is redundant and they do not reach any singularity.* Δ

Assumption 3.5 *The matrix J_{ϕ_i} is Lipschitz continuous, i. e.*

$$\|J_{\phi_i}(q_i) - J_{\phi_i}(q_{ai})\| \leq L_i \|q_i - q_{ai}\|, \tag{12}$$

for a positive constant L_i and for all $q_i, q_{ai} \in \mathbb{R}^n$. Besides, there exist positive finite constants c_{0i} and c_{1i} which satisfies

$$c_{0i} \triangleq \max_{\forall q_i \in \mathbb{R}^n} \|J_{oi}^+(q_i)\| \quad (13)$$

$$c_{1i} \triangleq \max_{\forall q_i \in \mathbb{R}^n} \|J_{oi}^-(q_i)\| \quad (14)$$

△

Note that Assumption 3.4 is a common one in the field of cooperative robots. None of the robots can be redundant that (2) is satisfied only by a bounded vector q_i . On the other hand, the closed kinematic loop that arises when the manipulators are holding an object is redundant. Assumption 3.5 is quite reasonable for revolute robots, since the elements of q_i appear as argument of sines and cosines functions. This is why (13)-(14) is valid.

3.3 Dynamic coupling

The position, velocity and acceleration of the object center of mass with reference to the inertial coordinated frame are given in Cartesian coordinates by:

$$x_o = h_i(q_i) \quad (15)$$

$$\dot{x}_o = J_{oi}(q_i)\dot{q}_i \quad (16)$$

$$\ddot{x}_o = J_{oi}(q_i)\ddot{q}_i + \dot{J}_{oi}(q_i)\dot{q}_i, \quad (17)$$

respectively, with $i = 1, 2$. $h_i(q_i) \in \mathbb{R}^3$ is the forward kinematics of the center of mass of the object expressed in the coordinate system C_0 , and $J_{oi}(q_i) \in \mathbb{R}^{3 \times n}$ is the corresponding Jacobian matrix of $h_i(q_i)$. Substituting (17) into (10) yields

$$m_o J_{oi}(q_i)\ddot{q}_i + m_o \dot{J}_{oi}(q_i)\dot{q}_i - m_o g_o = f_1 - f_2. \quad (18)$$

Now, consider writing (1) as (Murray *et. al* 1994)

$$H_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + D_i\dot{q}_i + g_i(q_i) = \tau_i + J_{ai}^T(q_i)f_i. \quad (19)$$

Note that

$$J_{oi}^T(q_i) = J_{ai}^T(q_i)n_i, \quad (20)$$

in view of (11). $J_{ai}(q_i)$ is the manipulator analytical Jacobian. On the other hand, from (18) one gets

$$f_1 = m_o J_{oi}(q_i)\ddot{q}_i + m_o \dot{J}_{oi}(q_i)\dot{q}_i - m_o g_o + f_2. \quad (21)$$

Then, for $i=1$ in (19) one gets

$$\begin{aligned} H_1(q_1)\ddot{q}_1 + C_1(q_1, \dot{q}_1)\dot{q}_1 + D_1\dot{q}_1 + g_1(q_1) &= \\ \tau_1 + J_{a1}^T(q_1)(m_o J_{o1}(q_1)\ddot{q}_1 + m_o \dot{J}_{o1}(q_1)\dot{q}_1 - m_o g_o + f_2) &= \\ \tau_1 + J_{a1}^T(q_1)m_o J_{o1}(q_1)\ddot{q}_1 + J_{a1}^T(q_1)m_o \dot{J}_{o1}(q_1)\dot{q}_1 - J_{a1}^T(q_1)m_o g_o + J_{a1}^T(q_1)f_2, \end{aligned} \quad (22)$$

or

$$\begin{aligned} \tau_1 + J_{a1}^T(q_1)f_2 &= (H_1(q_1) - J_{a1}^T(q_1)m_o J_{o1}(q_1))\ddot{q}_1 \\ &+ (C_1(q_1, \dot{q}_1) + D_1 - J_{a1}^T(q_1)m_o \dot{J}_{o1}(q_1))\dot{q}_1 + g_1(q_1) + J_{a1}^T(q_1)m_o g_o. \end{aligned} \quad (23)$$

By defining

$$H_{T1}(q_1) \triangleq H_1(q_1) - J_{a1}^T(q_1)m_o J_{o1}(q_1) \quad (24)$$

$$C_{T1}(q_1, \dot{q}_1) \triangleq C_1(q_1, \dot{q}_1) + D_1 - J_{a1}^T(q_1)m_o \dot{J}_{o1}(q_1) \quad (25)$$

$$g_{T1}(q_1) \triangleq g_1(q_1) + J_{a1}^T(q_1)m_o g_o, \quad (26)$$

one finally gets

$$H_{T1}(q_1)\dot{q}_1 + C_{T1}(q_1, \dot{q}_1)\dot{q}_1 + g_{T1}(q_1) = \tau_1 + J_{a1}^T(q_1)f_2. \quad (27)$$

In the same fashion, we make the analysis for the robot A255, to get

$$H_{T2}(q_2)\dot{q}_2 + C_{T2}(q_2, \dot{q}_2)\dot{q}_2 + g_{T2}(q_2) = \tau_2 + J_{a2}^T(q_2)f_1, \quad (28)$$

with

$$H_{T2}(q_2) \triangleq H_2(q_2) + J_{a2}^T(q_2)m_o J_{o2}(q_2) \quad (29)$$

$$C_{T2}(q_2, \dot{q}_2) \triangleq C_2(q_2, \dot{q}_2) + D_2 + J_{a2}^T(q_2)m_o \dot{J}_{o2}(q_2) \quad (30)$$

$$g_{T2}(q_2) \triangleq g_2(q_2) - J_{a2}^T(q_2)m_o g_o. \quad (31)$$

The dynamic models in (27)-(28) describe the motion of the entire closed chain, where each individual manipulator represents a subsystem coupled to the other one through kinematic and dynamic constraints.

3.4 Force modeling for cooperative robots

A robot manipulator in free motion does not have geometric constraints; therefore, the dynamic model is described by Ordinary Differential Equations (ODE). When working with constrained motion, there appear holonomic constraints; for this reason, the dynamic model is described by Differential Algebraic Equations (DAE). To simulate contact forces, DAE's must be solved. First of all, from the dynamic model for cooperative robots (1), we obtain

$$\ddot{q}_i = H_i^{-1}(q_i)(\tau_i + J_{o_i}^T(q_i)\lambda_i - C_i(q_i, \dot{q}_i)\dot{q}_i - D_i\dot{q}_i - g_i(q_i)) \quad (32)$$

However, (7)-(9) must hold as well. Substituting the right hand side of (32) into (9) yields

$$\begin{aligned} \ddot{\phi}_i(q_i) &= J_{o_i}(q_i) \left[H_i^{-1}(q_i)(\tau_i + J_{o_i}^T(q_i)\lambda_i - C_i(q_i, \dot{q}_i)\dot{q}_i - D_i\dot{q}_i - g_i(q_i)) \right] \\ &\quad + \dot{J}_{o_i}(q_i)\dot{q}_i \\ &= J_{o_i}(q_i)H_i^{-1}(q_i)J_{o_i}^T(q_i)\lambda_i + \dot{J}_{o_i}(q_i)\dot{q}_i \\ &\quad + J_{o_i}(q_i)H_i^{-1}(q_i)(\tau_i - C_i(q_i, \dot{q}_i)\dot{q}_i - D_i\dot{q}_i - g_i(q_i)) \\ &= 0. \end{aligned} \quad (33)$$

From the previous equation we obtain

$$\begin{aligned} \lambda_i &= \left(J_{o_i}(q_i)H_i^{-1}(q_i)J_{o_i}^T(q_i) \right)^{-1} \left[\ddot{\phi}_i(q_i) - \dot{J}_{o_i}(q_i)\dot{q}_i \right. \\ &\quad \left. - J_{o_i}(q_i)H_i^{-1}(q_i)(\tau_i - C_i(q_i, \dot{q}_i)\dot{q}_i - D_i\dot{q}_i - g_i(q_i)) \right] \end{aligned} \quad (34)$$

The system described by (11), (27)-(28) and (34) could now be simulated as second order differential equations. However, the inclusion of the constraints in the form (34) does not guarantee the convergence of the contact velocity and position constraints to zero. This is because $\ddot{\phi}_i(q_i)=0$ represents a double integrator. Thus, any small

difference of $\varphi_i(q_i)$ or $\dot{\varphi}_i(q_i)$ from zero in (7)-(9) will diverge. This problem has been successfully addressed by the constraint stabilization method in the solution of DAE's (Baumgarte 1972).

According to this approach, the constraints are asymptotically stabilized by using

$$\ddot{\varphi}_i(q_i) + 2\alpha_i\dot{\varphi}_i(q_i) + \beta_i\varphi_i(q_i) = 0, \quad (35)$$

instead of $\ddot{\varphi}_i(q_i) = 0$. α_i and β_i are chosen appropriately to ensure the fast convergence of both the constraint position $\varphi_i(q_i)$ and velocity constraint $\dot{\varphi}_i(q_i)$ to zero (in case of offset). Equations (11), (27)-(28) and (34)-(35) fully describe the motion of the system to be simulated (Gudiño-Lau & Arteaga 2005).

4. Control with velocity estimation

4.1 Control Law

In this section, a linear filter for the force error and the tracking control problem of a cooperative system of rigid robots are studied. Consider model (1) and define the tracking and observation errors as

$$\tilde{q}_i \triangleq q_i - q_{di} \quad (36)$$

$$z_i \triangleq \dot{q}_i - \dot{\tilde{q}}_{i,*} \quad (37)$$

where q_{di} is a desired smooth bounded trajectory satisfying constraint (2), and $(\dot{\bullet})_*$ represents the estimated value of (\bullet) . Other error definitions are

$$\Delta p_i \triangleq p_i - p_{di} \quad (38)$$

$$\Delta \lambda_i \triangleq \lambda_i - \lambda_{di} \quad (39)$$

where p_{di} is the desired constrained position which satisfies (6). λ_{di} is the desired force to be applied by each finger on the constrained surface. Other useful definitions are

$$\dot{q}_{ni} \triangleq Q_i(q_i)(\dot{q}_{di} - \Lambda_i(\tilde{q}_i - q_{di})) + J_{ni}^+(q_i)(\dot{p}_{di} - \beta_i\Delta p_i + \bar{\xi}_i\phi + \xi_i\Delta F_i) \quad (40)$$

$$s_i \triangleq \dot{q}_i - \dot{q}_{ni} \\ = Q_i(q_i)(\dot{\tilde{q}}_i + \Lambda_i\tilde{q}_i - \Lambda_i z_i) + J_{ni}^+(q_i)(\Delta \dot{p}_i + \beta_i\Delta p_i - \bar{\xi}_i\phi - \xi_i\Delta F_i) - \quad (41)$$

$$\triangleq s_{pi} + s_{fi}$$

$$\Delta F_i \triangleq \int_0^t \Delta \lambda_i(\vartheta) d\vartheta, \quad (42)$$

where $\Lambda_i = k_i I \in \mathbb{R}^{n_i \times n_i}$ with $k_i > 0$, and $\bar{\xi}_i, \xi_i \in \mathbb{R}^{n_i \times m_i}$ are diagonal positive definite matrices, and β_i is a positive constant. To get (41) the equality $\dot{\tilde{q}}_i - \dot{q}_{di} = \dot{\tilde{q}}_i - z_i$ has been used. Note also that s_{pi} and s_{fi} are orthogonal vectors. $\phi \in \mathbb{R}^m$ is the output of the linear filter given by

$$\dot{w}_i = -A_i w_i + \Delta \lambda_i \quad w_i(0) = 0 \quad (43)$$

$$\dot{\phi} = B_i w_i. \quad (44)$$

$A_i, B_i \in \mathbb{R}^{n_i \times n_i}$ are diagonal positive definite matrices and $w_i \in \mathbb{R}^{n_i}$ is the state for the filter. Also, we define

$$\zeta_i \triangleq \dot{\phi} = -B_i A_i w_i + B_i \Delta \lambda_i = -A_i \phi + B_i \Delta \lambda_i. \quad (45)$$

Now, let us analyze q_{ri} . This quantity is given by

$$\begin{aligned} \ddot{q}_{ri} \triangleq & \mathbf{Q}_i(q_i)(\ddot{q}_{di} - \Lambda_i(\dot{q}_i - \dot{q}_{di})) + \mathbf{J}_{oi}^+(q_i)(\ddot{p}_{di} - \beta_i(\dot{p}_i - \dot{p}_{di}) + \bar{\xi}_i\zeta_i + \xi_i\Delta\lambda_i) \\ & + \dot{\mathbf{Q}}_i(q_i)(\dot{q}_{di} - \Lambda_i(\dot{q}_i - \dot{q}_{di})) + \dot{\mathbf{J}}_{oi}^+(q_i)(\dot{p}_{di} - \beta_i\Delta p_i + \bar{\xi}_i\phi_i + \xi_i\Delta F_i). \end{aligned} \quad (46)$$

As it will be shown later, \ddot{q}_{ri} is necessary to implement the controller and the observer. However, this quantity is not available since \dot{q}_i is not measurable. In order to overcome this drawback, let us consider $\mathbf{Q}_i(q_i) \in \mathbb{R}^{n_i \times n_i}$. Then you have

$$\dot{\mathbf{Q}}_i(q_i) = \begin{bmatrix} \frac{\partial a_{11}(q_i)}{\partial q_i} \dot{q}_i & \dots & \frac{\partial a_{1n_i}(q_i)}{\partial q_i} \dot{q}_i \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{n_i1}(q_i)}{\partial q_i} \dot{q}_i & \dots & \frac{\partial a_{n_in_i}(q_i)}{\partial q_i} \dot{q}_i \end{bmatrix}, \quad (47)$$

where $a_{\alpha\beta}$ is the $\alpha\beta$ element of $\mathbf{Q}_i(q_i)$. Based on (47), consider the following definition

$$\dot{\hat{\mathbf{Q}}}_i(q_i) \triangleq \begin{bmatrix} \frac{\partial a_{11}(q_i)}{\partial q_i} \dot{q}_{oi} & \dots & \frac{\partial a_{1n_i}(q_i)}{\partial q_i} \dot{q}_{oi} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{n_i1}(q_i)}{\partial q_i} \dot{q}_{oi} & \dots & \frac{\partial a_{n_in_i}(q_i)}{\partial q_i} \dot{q}_{oi} \end{bmatrix}, \quad (48)$$

with

$$\dot{q}_{oi} \triangleq \dot{q}_i - \Lambda_i z_i. \quad (49)$$

Then, one can compute

$$\ddot{\hat{\mathbf{Q}}}_i(r_i) \triangleq \dot{\mathbf{Q}}_i(q_i) \cdot \dot{\hat{\mathbf{Q}}}_i(\dot{q}_{oi}) = \begin{bmatrix} \frac{\partial a_{11}(q_i)}{\partial q_i} r_i & \dots & \frac{\partial a_{1n_i}(q_i)}{\partial q_i} r_i \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{n_i1}(q_i)}{\partial q_i} r_i & \dots & \frac{\partial a_{n_in_i}(q_i)}{\partial q_i} r_i \end{bmatrix}, \quad (50)$$

where

$$r_i \triangleq \dot{q}_i - \dot{q}_{oi} = \dot{z}_i + \Lambda_i z_i. \quad (51)$$

In view of (48), we propose the following substitution for \ddot{q}_{ri}

$$\begin{aligned} \ddot{q}_{ri} \triangleq & \mathbf{Q}_i(q_i)(\ddot{q}_{di} - \Lambda_i(\dot{q}_i - \dot{q}_{di})) + \mathbf{J}_{oi}^+(q_i)(\ddot{p}_{di} - \beta_i(\dot{p}_i - \dot{p}_{di}) + \bar{\xi}_i\zeta_i + \xi_i\Delta\lambda_i) \\ & + \dot{\hat{\mathbf{Q}}}_i(q_i)(\dot{q}_{di} - \Lambda_i(\dot{q}_i - \dot{q}_{di})) + \dot{\mathbf{J}}_{oi}^+(q_i)(\dot{p}_{di} - \beta_i\Delta p_i + \bar{\xi}_i\phi_i + \xi_i\Delta F_i), \end{aligned} \quad (52)$$

where $\dot{\hat{\mathbf{J}}}_{oi}^+(q_i)$ is defined in the very same fashion as $\dot{\hat{\mathbf{Q}}}_i(q_i)$ in (48). Note that \dot{p}_i is still used since this value is known from (6). After some manipulation, it is possible to get

$$\ddot{\hat{q}}_{ri} = \ddot{q}_{ri} + e_i(r_i), \quad (53)$$

where

$$e_i(r_i) \triangleq -\ddot{Q}_i(q_i)(\dot{q}_{di} - \Lambda_i \ddot{q}_i + \Lambda_i z_i) - \ddot{J}_{ci}^T(q_i)(\dot{p}_{ci} - \beta_i \Delta p_i + \ddot{\xi}_i \phi_i + \xi_i^T \Delta F_i) \quad (54)$$

The proposed controller is then given for each single input by

$$\begin{aligned} \tau_i &\triangleq H_i(q_i) \ddot{q}_{oi} + C_i(q_i, \dot{q}_{oi}) \dot{q}_{oi} + D_i \dot{q}_{oi} + g_i(q_i) \\ &\quad - K_{Ri}(\dot{q}_{oi} - \dot{q}_{ri}) - J_{oi}^T(q_i)(\lambda_{oi} + B_i^{-1} \zeta_i - K_{Fi} \Delta F_i) \end{aligned} \quad (55)$$

where $K_{Ri} \in \mathbb{R}^{n \times n}$, $K_{Fi} \in \mathbb{R}^{n \times n}$ are diagonal positive definite matrices. Note that from (41) and (51) it is $\dot{q}_{oi} - \dot{q}_{ri} = s_i - r_i$. Thus, from (53) one gets

$$\begin{aligned} \tau_i &= H_i(q_i)(\dot{q}_{oi} + e_i(r_i)) + C_i(q_i, \dot{q}_{oi}) \dot{q}_{oi} + D_i \dot{q}_{oi} + g_i(q_i) \\ &\quad - K_{Ri}(s_i - r_i) - J_{oi}^T(q_i)(\lambda_{oi} + B_i^{-1} \zeta_i - K_{Fi} \Delta F_i) \end{aligned} \quad (56)$$

By substituting (56) into (1), the closed loop dynamics becomes

$$\begin{aligned} H_i(q_i) \dot{s}_i &= -C_i(q_i, \dot{q}_i) s_i - K_{DRi} s_i + K_{Ri} r_i \\ &\quad + J_{oi}^T(q_i)(B_i^{-1} A_i \phi_i + K_{Fi} \Delta F_i) - C_i(q_i, \dot{q}_{oi}) s_i + H_i(q_i) e_i(r_i) \end{aligned} \quad (57)$$

after some manipulation, where $K_{DRi} \triangleq K_{Ri} + D_i$. In order to get (57), Property 3.3 has been used.

4.2 Observer definition

The proposed dynamics of the observer is given by

$$\begin{aligned} \dot{\hat{q}}_i &= \hat{q}_{oi} + \Lambda_i z_i + k_{di} z_i \\ \dot{\hat{q}}_{oi} &= \hat{q}_{ri} + k_{di} \Lambda_i z_i \end{aligned} \quad (58)$$

where k_{di} is a positive constant. This observer is simpler than the one given in (Gudiño-Lau *et al.* 2004), where the inverse of the inertia matrix and force measurements are required. Since from (58) you have $\ddot{\hat{q}}_{oi} = \ddot{\hat{q}}_i - \Lambda_i \dot{z}_i - k_{di} \dot{z}_i$, (59) becomes

$$\dot{s}_i = \dot{r}_i + k_{di} r_i + e_i(r_i), \quad (60)$$

in view of (53). By multiplying both sides of (60) by $H_i(q_i)$, and by taking into account (57), one gets

$$\begin{aligned} H_i(q_i) \dot{r}_i &= -H_{rdi} r_i - C_i(q_i, \dot{q}_i) s_i - C_i(q_i, \dot{q}_{oi}) s_i - K_{DRi} s_i \\ &\quad + J_{oi}^T(q_i)(B_i^{-1} A_i \phi_i + K_{Fi} \Delta F_i), \end{aligned} \quad (61)$$

where $H_{rdi} \triangleq k_{di} H_i(q_i) - K_{Ri}$. Finally, by using Property 3.3 again and some manipulation, it is

$$\begin{aligned} H_i(q_i) \dot{r}_i &= -C_i(q_i, \dot{q}_i) r_i - H_{rdi} r_i + C_i(q_i, s_i + \dot{q}_{oi}) r_i - K_{DRi} s_i \\ &\quad - C_i(q_i, s_i + 2\dot{q}_{oi}) s_i + J_{oi}^T(q_i)(B_i^{-1} A_i \phi_i + K_{Fi} \Delta F_i). \end{aligned} \quad (62)$$

Now, let us define

$$x_i \triangleq [s_i^T \quad r_i^T \quad \Delta F_i^T \quad \phi_i^T]^T, \quad (63)$$

as state for (42), (45), (57) and (62). The main idea of the control-observer design is to show that whenever $\|x_i\|$ tends to zero, the tracking errors \tilde{q}_i , $\dot{\tilde{q}}_i$, Δp_i , $\Delta \dot{p}_i$ and $\Delta \lambda_i$ and the

observation errors \mathbf{z}_i and $\dot{\mathbf{z}}_i$ will do it as well. From (51), this is rather obvious for \mathbf{z}_i and $\dot{\mathbf{z}}_i$. However, it is not clear for the other variables. The following lemma shows that this is indeed the case under some conditions.

Lemma 4.1 If \mathbf{x}_i is bounded by $x_{\max i}$ and tends to zero, then the following facts hold:

- a) $\Delta \mathbf{p}_i$ and $\Delta \dot{\mathbf{p}}_i$ remain bounded and tend to zero
- b) $\tilde{\mathbf{q}}_i$ and $\dot{\tilde{\mathbf{q}}}_i$ remain bounded. Furthermore, if the bound $x_{\max i}$ for $\|\mathbf{x}_i\|$ is chosen small enough so as to guarantee that $\|\tilde{\mathbf{q}}_i\| \leq \eta_i$ for all t , with η_i a positive and small enough constant, then both $\tilde{\mathbf{q}}_i$ and $\dot{\tilde{\mathbf{q}}}_i$ will tend to zero as well.
- c) If, in addition, the velocity vector $\dot{\mathbf{q}}_i$ is bounded, then $\Delta \mathcal{A}_i$ will remain bounded and tend to zero.

The proof of Lemma 4.1 can be found in Appendix B. It is interesting to note that, if $\|\mathbf{x}_i\|$ is bounded by $x_{\max i}$, then it is always possible to find a bound for $\mathbf{e}_i(\mathbf{r}_i)$ in (54) which satisfies

$$\|\mathbf{e}_i(\mathbf{r}_i)\| \leq M_{\mathbf{e}_i}(x_{\max i}) \|\mathbf{r}_i\| < \infty. \quad (64)$$

Consider now the following function

$$V_i(\mathbf{x}_i) = \frac{1}{2} \mathbf{x}_i^T M_i \mathbf{x}_i, \quad (65)$$

where $M_i \triangleq \text{block diag}\{H_i(\mathbf{q}_i), H_i(\mathbf{q}_i), R_i\}$, and

$$R_i \triangleq \begin{bmatrix} N_i B_i^{-1} & -N_i \\ -N_i & N_i B_i \end{bmatrix}, \quad (66)$$

with $N_i \triangleq (\xi_i B_i^{-1} A_i + \bar{\xi}_i K_{F_i}) A_i^{-1}$. In Appendix C it is shown that $V_i(\mathbf{x}_i)$ is a positive definite function. Suppose that one may find a region

$$\mathbb{D}_i = \{\mathbf{x}_i : \|\mathbf{x}_i\| \leq x_{\max i}\}, \quad (67)$$

so that for all time $\dot{V}_i(\mathbf{x}_i) \leq 0$ with $\dot{V}_i(\mathbf{x}_i) = 0$ if and only if $\mathbf{x}_i = \mathbf{0}$. If $x_{\max i}$ is small enough in the sense of Lemma 4.1, then from the former discussion one can conclude the convergence to zero of all error signals. The following theorem establishes the conditions for the controller-observer parameters to guarantee this.

Teorema 4.1 Consider the cooperative system dynamics given by (1), (2) and (6), in closed loop with the filter (43)-(45), the control law (55) and the observer (58)-(59), where \mathbf{q}_{di} and \mathbf{p}_{di} are the desired bounded joint and constrained positions, whose derivatives $\dot{\mathbf{q}}_{di}$, $\ddot{\mathbf{q}}_{di}$, $\dot{\mathbf{p}}_{di}$, and $\ddot{\mathbf{p}}_{di}$ are also bounded, and they all satisfy constraint (6). Consider also l given regions defined by (67) for each subsystem, where the bounds $x_{\max i}$, $i = 1, \dots, l$, are chosen according to

$$x_{\max i} \leq \frac{\eta_i \alpha_i}{\left(1 + c_{0i} \left(\lambda_{\max}(\bar{\xi}_i) + \lambda_{\max}(\xi_i)\right) + \sqrt{n_i}\right)} \quad (68)$$

with α_i defined in Appendix B. Then, every dynamic and error signal remains bounded and asymptotic stability of tracking, observation and force errors arise, i. e.

$$\lim_{t \rightarrow \infty} \tilde{q}_i = 0 \quad \lim_{t \rightarrow \infty} \dot{\tilde{q}}_i = 0 \quad \lim_{t \rightarrow \infty} z_i = 0 \quad \lim_{t \rightarrow \infty} \dot{z}_i = 0 \quad \lim_{t \rightarrow \infty} \Delta \lambda_i = 0, \quad (69)$$

if the following conditions are satisfied

$$\lambda_{\min}(\mathbf{K}_{Ri}) \geq \mu_{1i} + 1 + \delta_i \quad (70)$$

$$k_{di} \geq \frac{\lambda_{\max}(\mathbf{K}_{Ri}) + w_i}{\lambda_{hi}} \quad (71)$$

$$\lambda_{\min}(\mathbf{E}_i) \geq \delta_i + 1 \quad (72)$$

$$\lambda_{\min}(\xi_i \mathbf{K}_{Fi}) \geq \delta_i + 1, \quad (73)$$

where $w_i = \mu_{2i} + \frac{1}{4}(\lambda_{di} + \mu_{3i} + \mu_{4i})^2 + \delta_i + \frac{1}{4}c_{1i}^2 a_i^2 b_i^2 + \frac{1}{4}c_{1i}^2 \lambda_{\max}^2(\mathbf{K}_{Fi})$, $\mathbf{E}_i = \bar{\xi}_i \mathbf{B}_i^{-1} \mathbf{A}_i + \xi_i \mathbf{B}_i^2 \mathbf{A}_i + \bar{\xi}_i \mathbf{K}_{Fi} \mathbf{B}_i^{-1}$, δ_i

a positive constant and μ_{1i} , μ_{2i} , μ_{3i} , μ_{4i} and λ_{di} defined in Appendix D. Δ

The proof of the Theorem 4.1 can be found in Appendix D

Remark 4.1 The result of Theorem 4.1 is only local. Also, it is rather difficult to find analytically a region of attraction, but it should be noticed that it $\{\setminus\}$ cannot be made arbitrarily large. This is to guarantee the convergence to zero of the tracking errors \tilde{q}_i and $\dot{\tilde{q}}_i$. However, this does not represent a serious drawback since for grasping purposes it is usual to give smooth trajectories with zero initial position errors. On the other hand, it is worthy pointing out that a controller-observer scheme is implemented for every robot separately, while only the knowledge of each constraint of the form (2) is required. Δ .

5. Simulation and Experimental Results

In this section, some simulation results are presented. To test the accuracy of the modeling approach, experimental results are carried out as well. To protect the manipulators of the cooperative system against possible damages, the position/force control law (55) has been used for validation purposes, the motors dynamics has to be taken into account. For the object equation of motion given in (10), it is $m_o = m_{obj} \mathbf{I}$,

$m_{obj} = 0.400\text{kg}$, and $\mathbf{g}_o^T = \{g_x \quad g_y \quad g_z\} = \{0 \quad 0 \quad -9.81\text{m/s}^2\}$. The object dimensions are $0.15\text{m} \times 0.15\text{m} \times 0.311\text{m}$. In (35) one has $\alpha_i = 10$ and $\beta_i = 100$. The robots models are given in Appendix A.

The palm frame of the whole system is at the base of the robot A465, with its x -axis pointing towards the other manipulator. The task consists in lifting the object and pushing with a desired force, so that the constraints in Cartesian coordinates are simply given by

$$\varphi_i = x_i - b_i = 0, \quad (74)$$

for $i = 1, 2$ and b_i a positive constant. The desired trajectories are given by

$$x_{d1} = 0.554 \text{ [m]} \quad x_{d2} = 0.865 \text{ [m]} \quad (75)$$

$$y_{d1,2} = 0.05 \sin(w(t-t_i)) \text{ [m]} \quad (76)$$

$$z_{d1,2} = (0.635 + 0.05 \cos(w(t-t_i)) - 0.05) \text{ [m]} \quad (77)$$

Note that the inverse kinematics of the manipulators has to be employed to compute \mathbf{q}_{di} . These trajectories are valid from an initial time $t_i = 10\text{s}$ to a final time $t_f = 70\text{s}$. Before t_i and after t_f the robots are in free motion. w is a fifth order polynomial designed to satisfy $w(t_i) = w(t_f) = 0$. The derivatives of w are also zero at t_i and t_f . By choosing (75)-(77), the robots will make a circle in the y - z plane. The only difference between the trajectories for robots A465 and A255 is the width of the object. Also, no force control is carried out until the manipulators are in the initial position to hold the object, at $(0.554, 0, 0.510)$ [m] for the first manipulator and $(0.865, 0, 0.510)$ [m] for the second one. The desired pushing forces are then given from $t = t_i = 10\text{s}$ to $t = t_f = 70\text{s}$ by.

$$f_{dx1,2} = 15(t-t_i)/10 \text{ [N]} \quad t_i \text{ s} \leq t < 20\text{s} \quad (75)$$

$$f_{dx1,2} = 15 + 5 \sin(3\pi(t-20)/40) \text{ [N]} \quad 20 \leq t < 60\text{s} \quad (76)$$

$$f_{dx1,2} = 15 - 7.5(t-60)/10 \text{ [N]} \quad 60 \leq t < t_f \text{ s} \quad (77)$$

and $f_{dy1,2} = f_{dz1,2} = 0$ [N]. Note that, for simplicity, the desired forces are expressed in the base coordinate frame of each robot.

The controller has also been digitalized for the simulation. The experiment lasts 80s. The object is held at $t=10\text{s}$. Before, the robots are in free movement and the control law (55) and the observer (58)-(59) are used with the force part set to zero (*i. e.* $\mathbf{Q}_i = \mathbf{I}$ and $\mathbf{J}_{oi} = \mathbf{0}$). It is rather easy to prove that this scheme is stable for unconstrained motion. From $t=10\text{s}$ to $t=70\text{s}$ it is switched on, *i. e.*, the complete control-observer force scheme is employed only during this period of time. From $t=70\text{s}$ to $t=80\text{s}$ the robots go back to their initial positions in free motion. From $t=10\text{s}$ to $t=15\text{s}$ they begin pushing at their initial positions to hold the object, and from $t=15\text{s}$ to $t=20\text{s}$ they lift it to the position where the circle will be made. From $t=20\text{s}$ to $t=60\text{s}$ this is done while the desired force is changed for a sinus signal, as can be seen in Figure 6. Note that our purpose is to show that simulation results of the constrained system are acceptable by using the approach described in Section 3. For this reason, the desired forces (or positions) are not shown. Only the real and simulated signals are presented. As can be seen, there is a good match. Of course, simulation results are free of noise. Note also that, since we have not proposed any special method to simulate the moment when the object is held, *i. e.*, when the robots change from free to constrained motion, there is a peak at $t=10\text{s}$ in the simulation. From $t=60\text{s}$ to $t=65\text{s}$ the object is put down on the table and from $t=65\text{s}$ to $t=70\text{s}$, the robots diminish pushing. Figure 4 shows the simulation and experimental results of the joint coordinates, while Figure 5 shows the results in Cartesian coordinates. As can be appreciated, the results are good in both cases. On the other hand, the Figure 7 and 8 show only the experimental results, for demonstrate the accuracy of the controller-observer scheme. The Figure 7 show the observation error, as can be appreciated, they are pretty. Finally, Figure 8 show the input voltages. In can be observed that there are not saturation problems. This demonstrates the efficacy of designing a decentralized controller.

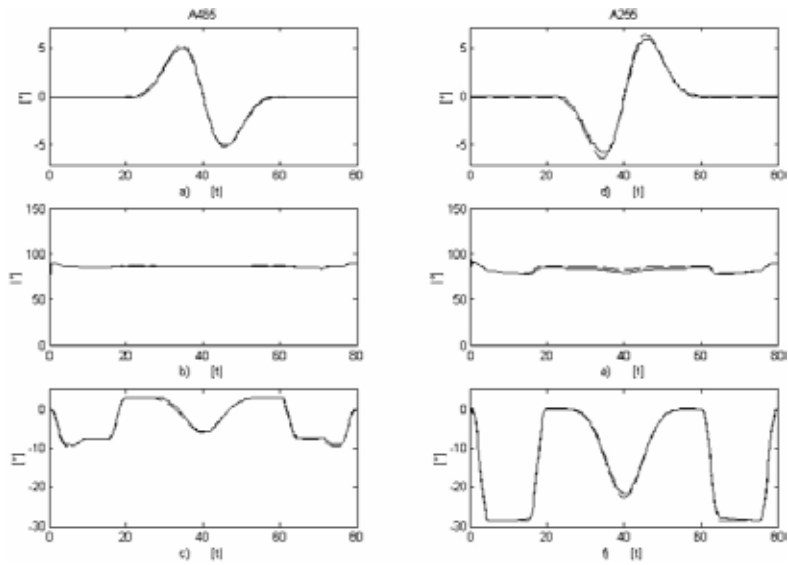


Fig. 4. Tracking in joint coordinates. a) q_{11} . b) q_{12} . c) q_{13} . d) q_{21} . e) q_{22} . f) q_{23} . ----- experiment - - - simulation.

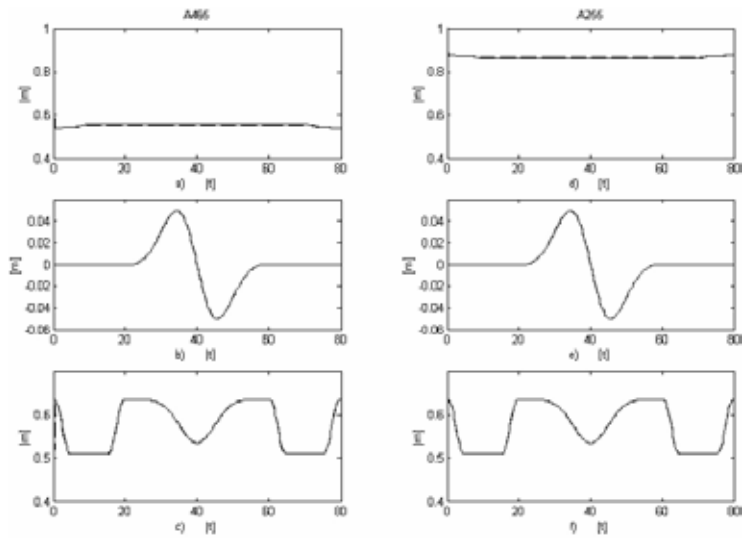


Fig. 5. Tracking in Cartesian coordinates. a) x_1 . b) y_1 . c) z_1 . d) x_2 . e) y_2 . f) z_2 . ----- experiment - - - simulation.

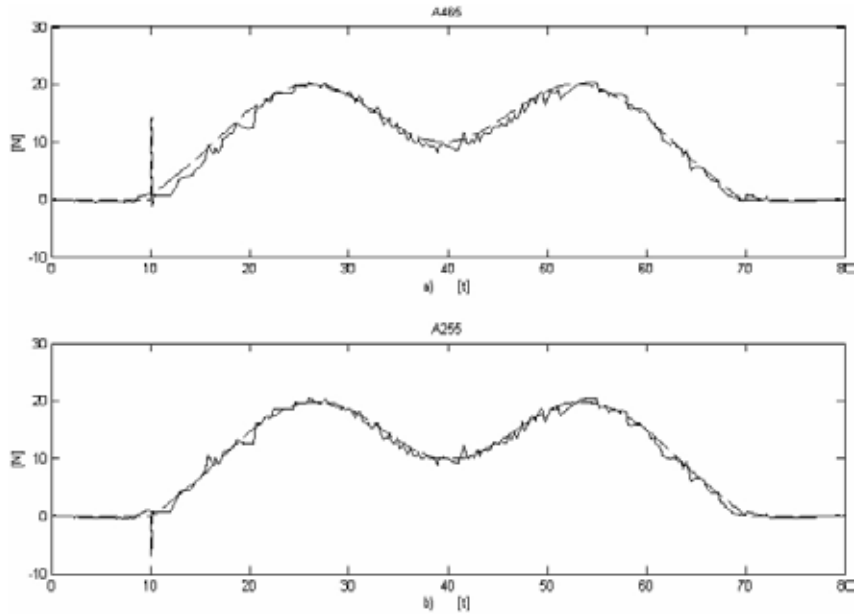


Fig. 6. Force measurements. a) λ_1 . b) λ_2 ----- experiment - - - simulation.

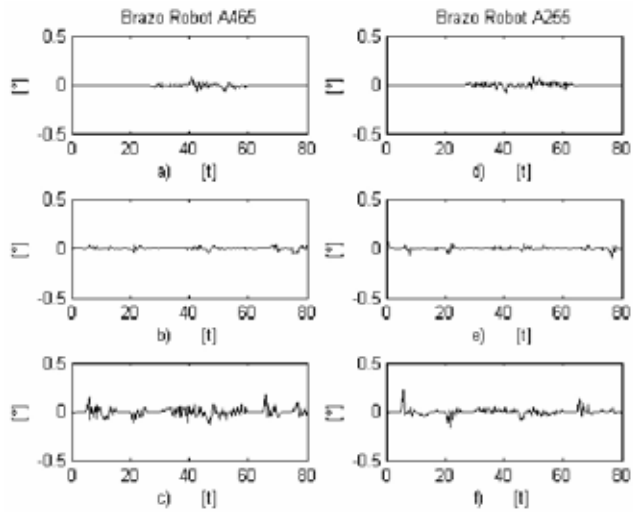


Fig. 7. Observation errors. a) z_{11} . b) z_{12} . c) z_{13} . d) z_{21} . e) z_{22} . f) z_{23} . — experiment - - - simulation.

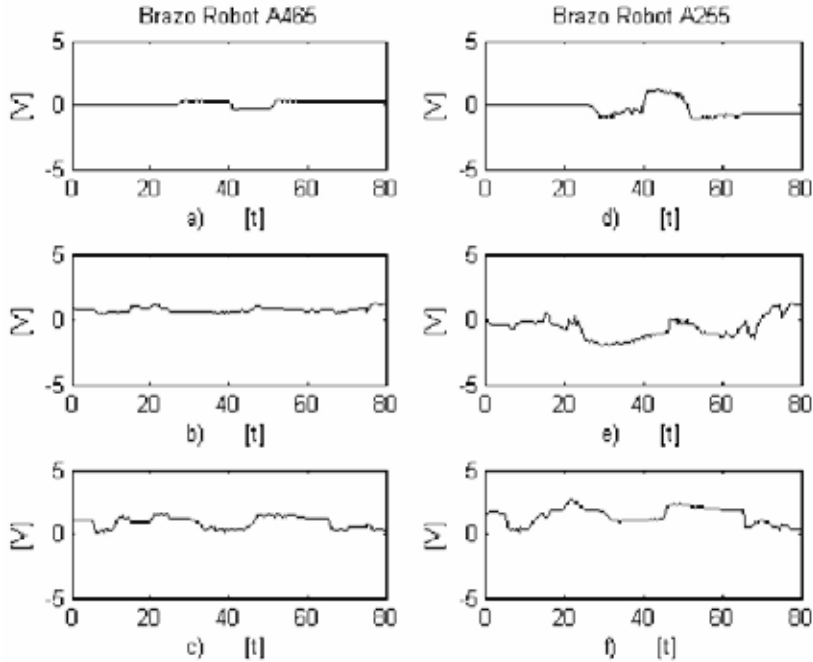


Fig. 8. Input voltages. a) v_{11} . b) v_{12} . c) v_{13} . d) v_{21} . e) v_{22} . f) v_{23} . ----- experiment - - - simulation.

6. Conclusions

In this chapter, we developed the model for two cooperative industrial robots holding a rigid object without friction. The dynamic model for the manipulators is obtained independently from each other with the Lagrangian approach. Once the robots are holding the object, their joint variables are kinematically and dynamically coupled. These coupling equations are combined with the dynamic model of the object to obtain a mathematical description for the cooperative system.

Besides, the tracking control problem for cooperative robots without velocity measurements is considered. The control law is a decentralized approach which takes into account motion constraints rather than the held object dynamics. By assuming that fingers dynamics are well known and that contact forces measurements are available, a linear observer for each finger is proposed which does not require any knowledge of the robots dynamics. Despite the fact that the stability analysis is complex, the controller and specially the observer are not.

Some experiments and simulations have been carried out to test the theoretical results. The overall outcome of the mathematical model compared with the real system can be considered good, which validates the approach used.

7. Appendix

A The A465 and A255 robot models.

This section presents the A465 and A255 robot models as well as the corresponding parameter values. The models used for implementation and simulation purposes include Coulomb friction term for both robots. The approach to model them can be found in any standard book for robotics (*e. g.* (Sciavicco & Siciliano 2000)). Recall that only the first three degrees of freedom of each manipulator are being considered. Additionally, since the actuators are DC motors, their dynamics must be taken into account. Thus, for each manipulator (in free motion), one has

$$\mathbf{H}_i(\mathbf{q}_i)\ddot{\mathbf{q}}_i + \mathbf{C}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)\dot{\mathbf{q}}_i + \mathbf{D}_i\dot{\mathbf{q}}_i + \mathbf{f}_{ci}(\dot{\mathbf{q}}_i) + \mathbf{g}_i(\mathbf{q}_i) = \mathbf{D}_{ni}^{-1}\mathbf{D}_{ki}\mathbf{v}_i, \quad (78)$$

where $\mathbf{f}_{ci}(\dot{\mathbf{q}}_i) \in \mathbb{R}^3$ represents the Coulomb friction term and \mathbf{D}_{ni} and $\mathbf{D}_{ki} \in \mathbb{R}^{3 \times 3}$ are to be defined later. The motors inertias are included in the matrix $\mathbf{H}_i(\mathbf{q}_i)$ so as to have a minimum set of parameters. The elements of matrices $\mathbf{H}_i(\dot{\mathbf{q}}_i)$, $\mathbf{C}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i)$, $\mathbf{f}_{ci}(\dot{\mathbf{q}}_i)$ and $\mathbf{g}_i(\mathbf{q}_i)$ of the model for Robot Arm A465 of CRS Robotics are computed as

$$\begin{aligned} h_1(1,1) &= \text{aux}_1 p_1 + \text{aux}_2 p_2 + \text{aux}_3 p_3 + \text{aux}_4 p_4 + \text{aux}_5 p_5 + \text{aux}_6 p_6 + \text{aux}_7 p_7 + p_8 \\ h_1(1,2) &= 0 \\ h_1(1,3) &= 0 \\ h_1(2,1) &= 0 \\ h_1(2,2) &= \frac{1}{2} p_1 + p_2 + 2s_3 p_3 + p_4 + p_9 \\ h_1(2,3) &= \frac{1}{2} p_1 + p_2 + s_3 p_3 + p_4 + p_5 \\ h_1(3,1) &= 0 \\ h_1(3,2) &= \frac{1}{2} p_1 + p_2 + s_3 p_3 + p_4 + p_5 \\ h_1(3,3) &= \frac{1}{2} p_1 + p_2 + p_4 + p_{10} \\ c_1(1,1) &= \text{aux}_8 p_3 + \text{aux}_9 p_5 - \frac{1}{2} \dot{q}_{12} \sin(2q_{12}) p_6 + \frac{1}{2} \dot{q}_{12} \sin(2q_{12}) p_7 \\ c_1(1,2) &= \dot{q}_{11} \cos(2q_{12} + q_{13}) p_3 + \frac{1}{2} \dot{q}_{11} \sin(2q_{12} + 2q_{13}) p_5 - \frac{1}{2} \dot{q}_{11} \sin(q_{12}) p_6 + \frac{1}{2} \dot{q}_{11} \sin(2q_{12}) p_7 \\ c_1(1,3) &= \text{aux}_{10} \cdot p_3 + \frac{1}{2} \dot{q}_{11} \sin(2q_{12} + 2q_{13}) p_5 \\ c_1(2,1) &= -\dot{q}_{11} \cos(2q_{12} + q_{13}) p_3 - \frac{1}{2} \dot{q}_{11} \sin(2q_{12} + 2q_{13}) p_5 + \frac{1}{2} \dot{q}_{11} \sin(2q_{12}) p_6 - \frac{1}{2} \dot{q}_{11} \sin(2q_{12}) p_7 \\ c_1(2,2) &= \dot{q}_{13} c_3 p_3 \\ c_1(2,3) &= (\dot{q}_{12} c_3 + \dot{q}_{13} c_3) p_3 \\ c_1(3,1) &= -\left(\frac{1}{2} \dot{q}_{11} c_3 + \frac{1}{2} \dot{q}_{11} \cos(2q_{12} + q_{13}) \right) p_3 - \frac{1}{2} \dot{q}_{11} \sin(2q_{12} + 2q_{13}) p_5 \\ c_1(3,2) &= -\dot{q}_{12} c_3 p_3 \\ c_1(3,3) &= 0 \end{aligned}$$

$$\begin{aligned}
f_{c1}(1) &= p_{14} \operatorname{sgn}(\dot{q}_{11}) \\
f_{c1}(2) &= p_{15} \operatorname{sgn}(\dot{q}_{12}) \\
f_{c1}(3) &= p_{16} \operatorname{sgn}(\dot{q}_{13}) \\
g_1(1) &= 0 \\
g_1(2) &= c_2 p_{17} + \sin(q_{12} + q_{13}) p_{21} \\
g_1(3) &= \sin(q_{12} + q_{13}) p_{18} .
\end{aligned}$$

Also, it is $D_1 = \operatorname{blocdiag}\{p_{11} \ p_{12} \ p_{13}\} \cdot v_1^T = \{v_{11} \ v_{12} \ v_{13}\}$ is the input voltage. The motor

dynamics data are $D_{n1} = \operatorname{blocdiag}\left\{\frac{1}{r_{11}^2} \ \frac{1}{r_{12}^2} \ \frac{1}{r_{13}^2}\right\}$ and $D_{k1} = \operatorname{blocdiag}\left\{\frac{k_{a11}}{R_{a11}r_{11}} \ \frac{k_{a12}}{R_{a12}r_{12}} \ \frac{k_{a13}}{R_{a13}r_{13}}\right\}$.

Where r stands for the gear ratio, K_a is the torque constant and R_a is the armature resistance. The associated values are $r_{11} = r_{12} = r_{13} = 100$, $K_{a11} = K_{a12} = K_{a13} = 0.1424$ Nm/A and $R_{a11} = R_{a12} = R_{a13} = 0.84\Omega$.

The elements of the corresponding matrices for the Robot Arm A255 are given by

$$\begin{aligned}
h_2(1,1) &= \cos(2q_{22})\bar{p}_1 + (2c_5 + 2c_6)\bar{p}_2 + \bar{p}_3 \\
h_2(1,2) &= 0 \\
h_2(1,3) &= 0 \\
h_2(2,1) &= 0 \\
h_2(2,2) &= \bar{p}_4 \\
h_2(2,3) &= \cos(q_{22} - q_{23})\bar{p}_5 \\
h_2(3,1) &= 0 \\
h_2(3,2) &= \cos(q_{22} - q_{23})\bar{p}_5 \\
h_2(3,3) &= \bar{p}_6 \\
c_2(1,1) &= -\dot{q}_{22}\sin(2q_{22})\bar{p}_1 - (s_5\dot{q}_{22} + s_6\dot{q}_{23})\bar{p}_2 \\
c_2(1,2) &= -\dot{q}_{21}\sin(2q_{22})\bar{p}_1 - \dot{q}_{21}s_5\bar{p}_2 \\
c_2(1,3) &= -\dot{q}_{21}s_6\bar{p}_2 \\
c_2(2,1) &= \dot{q}_{21}\sin(2q_{22})\bar{p}_1 + \dot{q}_{21}s_5\bar{p}_2 \\
c_2(2,2) &= 0 \\
c_2(2,3) &= \dot{q}_{23}\sin(q_{22} - q_{23})\bar{p}_5 \\
c_2(3,1) &= \dot{q}_{21}s_6\bar{p}_2 \\
c_2(3,2) &= -\dot{q}_{22}\sin(q_{22} - q_{23})\bar{p}_5 \\
c_2(3,3) &= 0 \\
f_{c2}(1) &= \bar{p}_{10} \operatorname{sgn}(\dot{q}_{21}) \\
f_{c2}(2) &= \bar{p}_{11} \operatorname{sgn}(\dot{q}_{22}) \\
f_{c2}(3) &= \bar{p}_{12} \operatorname{sgn}(\dot{q}_{23}) \\
g_2(1) &= 0 \\
g_2(2) &= c_5\bar{p}_{13} \\
g_2(3) &= c_6\bar{p}_{14} .
\end{aligned}$$

For this robot one has $D_2 = \operatorname{blocdiag}\{\bar{p}_7 \ \bar{p}_8 \ \bar{p}_9\}$. The input voltages vector is given by

$v_2^T = \{v_{21} \ v_{22} \ v_{23}\}$. The motor dynamics data are $D_{n2} = \operatorname{blocdiag}\left\{\frac{1}{r_{21}^2} \ \frac{1}{r_{22}^2} \ \frac{1}{r_{23}^2}\right\}$ and

$D_{a2} = \text{bloc diag} \left\{ \frac{k_{a21}}{R_{a21}r_{21}}, \frac{k_{a22}}{R_{a22}r_{22}}, \frac{k_{a23}}{R_{a23}r_{23}} \right\}$. With $r_{21} = r_{22} = r_{23} = 72$, $K_{a21} = K_{a22} = K_{a23} = 0.0657 \text{ Nm/A}$ and $R_{a21} = R_{a22} = R_{a23} = 2.40\Omega$. Note that in the model of both robots we could have chosen the parameters in a different fashion to have a smaller set. However, we made the definitions according to the computation of the inertia of the links. The elements of the analytical Jacobian $J_{a1}(q_1)$ of robot A465 are given by

$$j_{a1}(1,1) = -a_{12}s_1c_2 - (d_{13} + d_{14})s_1\sin(q_{12} + q_{13}) \quad (79)$$

$$j_{a1}(1,2) = -a_{12}c_1s_2 + (d_{13} + d_{14})c_1\cos(q_{12} + q_{13}) \quad (80)$$

$$j_{a1}(1,3) = (d_{13} + d_{14})c_1\cos(q_{12} + q_{13}) \quad (81)$$

$$j_{a1}(2,1) = a_{12}c_1c_2 + (d_{13} + d_{14})c_1\sin(q_{12} + q_{13}) \quad (82)$$

$$j_{a1}(2,2) = -a_{12}s_1s_2 + (d_{13} + d_{14})s_1\cos(q_{12} + q_{13}) \quad (83)$$

$$j_{a1}(2,3) = (d_{13} + d_{14})s_1\cos(q_{12} + q_{13}) \quad (84)$$

$$j_{a1}(3,1) = 0 \quad (85)$$

$$j_{a1}(3,2) = a_{12}c_2 + (d_{13} + d_{14})\sin(q_{12} + q_{13}) \quad (86)$$

$$j_{a1}(3,3) = (d_{13} + d_{14})\sin(q_{12} + q_{13}), \quad (87)$$

and those of $J_{a2}(q_2)$ are:

$$j_{a2}(1,1) = -a_{22}s_4c_5 - a_{23}s_4c_6 - d_{24}s_4 \quad (88)$$

$$j_{a2}(1,2) = -a_{22}c_4s_5 \quad (89)$$

$$j_{a2}(1,3) = -a_{23}c_4s_6 \quad (90)$$

$$j_{a2}(2,1) = a_{22}c_4c_5 + a_{23}c_4c_6 + d_{24}c_4 \quad (91)$$

$$j_{a2}(2,2) = -a_{22}s_4s_5 \quad (92)$$

$$j_{a2}(2,3) = -a_{23}s_4s_6 \quad (93)$$

$$j_{a2}(3,1) = 0 \quad (94)$$

$$j_{a2}(3,2) = a_{22}c_5 \quad (95)$$

$$j_{a2}(3,3) = a_{23}c_6. \quad (96)$$

The constraint Jacobian matrix $J_{\phi 1}(q_1)$ of the robot arm A465 is:

$$J_{\phi 1}^T(q_1) = \begin{bmatrix} -a_{12}s_1c_2 - (d_{13} + d_{14})s_1\sin(q_{12} + q_{13}) \\ -a_{12}c_1s_2 + (d_{13} + d_{14})c_1\cos(q_{12} + q_{13}) \\ (d_{13} + d_{14})c_1\cos(q_{12} + q_{13}) \end{bmatrix}, \quad (97)$$

and the constraint Jacobian matrix $J_{\phi 2}(q_2)$ of the robot arm A255 is:

$$J_{\phi 2}^T(q_2) = \begin{bmatrix} -a_{22}s_4c_5 - a_{23}s_4c_6 - d_{24}s_4 \\ -a_{22}c_4s_5 \\ -a_{23}c_4s_6 \end{bmatrix}. \quad (98)$$

Note that, for simplicity, both $J_{a2}^T(q_1)$ and $J_{\phi 2}(q_2)$ are expressed with respect with an inertial system fixed at the base of robot A255.

$p_1 = 0.0055 \text{ kg m}^2$	$p_2 = 0.0080 \text{ kg m}^2$	$p_3 = 0.0024 \text{ kg m}^2$	$p_4 = 0.0118 \text{ kg m}^2$	$p_5 = 0.0041 \text{ kg m}^2$
$p_6 = 0.0009 \text{ kg m}^2$	$p_7 = 0.0007 \text{ kg m}^2$	$p_8 = 2.0007 \text{ kg m}^2$	$p_9 = 11.800 \text{ kg m}^2$	$p_{10} = 2.8000 \text{ kgm}^2$
$p_{11} = 25.000 \text{ Nm s}$	$p_{12} = 35.000 \text{ Nm s}$	$p_{13} = 36.000 \text{ Nm s}$	$p_{14} = 0.2000 \text{ Nm s}$	$p_{15} = 2.5000 \text{ Nm s}$
$p_{16} = 2.5000 \text{ Nm}$	$p_{17} = 22.000 \text{ N m}$	$p_{18} = 11.000 \text{ N m}$		

Table 1. Physical parameters of robot arm A465.

$\bar{p}_1 = 0.2500 \text{ kg m}^2$	$\bar{p}_2 = 0.0500 \text{ kgm}^2$	$\bar{p}_3 = 0.5750 \text{ kg m}^2$	$\bar{p}_4 = 1.1000 \text{ kgm}^2$	$\bar{p}_5 = 0.0300 \text{ kgm}^2$
$\bar{p}_6 = 0.5700 \text{ kg m}^2$	$\bar{p}_7 = 3.2000 \text{ Nm s}$	$\bar{p}_8 = 1.8000 \text{ Nm s}$	$\bar{p}_9 = 1.2000 \text{ Nm s}$	$\bar{p}_{10} = 0.0150 \text{ Nm s}$
$\bar{p}_{11} = 0.8000 \text{ N m}$	$\bar{p}_{12} = 0.7000 \text{ N m}$	$\bar{p}_{13} = 0.0001 \text{ N m}$	$\bar{p}_{14} = 1.8000 \text{ N m}$	

Table 2. Physical parameters of robot arm A255.

$s_1 = \sin(q_{11})$	$c_1 = \cos(q_{11})$
$s_2 = \sin(q_{12})$	$c_2 = \cos(q_{12})$
$s_3 = \sin(q_{13})$	$c_3 = \cos(q_{13})$
$s_4 = \sin(q_{21})$	$c_4 = \cos(q_{21})$
$s_5 = \sin(q_{22})$	$c_5 = \cos(q_{22})$
$s_6 = \sin(q_{23})$	$c_6 = \cos(q_{23})$

Table 3. Auxiliary definitions.

$\text{aux}_1 = \frac{3}{4} + \frac{1}{4} \cos(2q_{12} + 2q_{13})$
$\text{aux}_2 = \frac{1}{4} - \frac{1}{2} \cos(2q_{12} + 2q_{13})$
$\text{aux}_3 = s_3 + \sin(2q_{12} + q_{13})$
$\text{aux}_4 = \frac{1}{2} + \frac{1}{2} \cos(2q_{12} + 2q_{13})$
$\text{aux}_5 = \frac{1}{2} - \frac{1}{2} \cos(2q_{12} + 2q_{13})$
$\text{aux}_6 = \frac{1}{2} + \frac{1}{2} \cos(2q_{12})$
$\text{aux}_7 = \frac{1}{2} - \frac{1}{2} \cos(2q_{12})$
$\text{aux}_8 = \left[\dot{q}_{12} \cos(2q_{12} + q_{13}) + \frac{1}{2} \dot{q}_{13} c_3 + \frac{1}{2} \dot{q}_{13} \cos(2q_{12} + q_{13}) \right] \frac{1}{\text{seg}}$
$\text{aux}_9 = \left[\frac{1}{2} \dot{q}_{13} \sin(2q_{12} + 2q_{13}) + \frac{1}{2} \dot{q}_{12} \sin(2q_{12} + 2q_{13}) \right] \frac{1}{\text{seg}}$
$\text{aux}_{10} = \left[\frac{1}{2} \dot{q}_{11} c_3 + \frac{1}{2} \dot{q}_{11} \cos(2q_{12} + q_{13}) \right] \frac{1}{\text{seg}}$

Table 4. Auxiliary variables in the model of the robot arm A465.

Tables 1. and 2 show the parameter values, and Tables 3 and 4 the auxiliary variables for both robots. The parameters for the different Jacobian matrices are presented in Table 5.

Robot arm A465	Robot arm A255
$d_{11} = 0.330$ [m]	$d_{21} = 0.381$ [m]
$a_{12} = 0.305$ [m]	$a_{22} = 0.254$ [m]
$d_{13} = 0.330$ [m]	$a_{23} = 0.254$ [m]
$d_{14} = 0.208$ [m]	$d_{24} = 0.183$ [m]

Table 5. Data of the different Jacobian matrices.

B Proof of Lemma 4.1

In this appendix Lemma 4.1 is proven, whose main assumption is that $\begin{bmatrix} \mathbf{s}_i^T & \mathbf{r}_i^T & \Delta \mathbf{F}_i^T & \boldsymbol{\phi}_i^T \end{bmatrix}^T$ is bounded by $\mathbf{x}_{\max i}$ and tends to zero.

- a) First of all we show that $\Delta \mathbf{p}_i$ and $\Delta \dot{\mathbf{p}}_i$ are bounded for all time, with $i = 1, \dots, l$.¹ To

do this we use the fact that $J_{\varphi_i}(\mathbf{q}_i)\mathbf{s}_i = \Delta \dot{\mathbf{p}}_i + \beta_i \Delta \mathbf{p}_i - \bar{\boldsymbol{\xi}}_i \boldsymbol{\phi}_i - \boldsymbol{\xi}_i \Delta \mathbf{F}$ or

$$\Delta \dot{\mathbf{p}}_i + \beta_i \Delta \mathbf{p}_i = J_{\varphi_i}(\mathbf{q}_i)\mathbf{s}_i + \bar{\boldsymbol{\xi}}_i \boldsymbol{\phi}_i + \boldsymbol{\xi}_i \Delta \mathbf{F}. \quad (99)$$

The righthand side of equation (99) is bounded and tends to zero because of the assumption on \mathbf{x}_i . Since the lefthand side of (99) represents a stable linear filter, both $\Delta \mathbf{p}_i$ and $\Delta \dot{\mathbf{p}}_i$ must be bounded and tend to zero.

- b) The next step is to analyze the behavior of the tracking errors $\tilde{\mathbf{q}}_i$ and $\dot{\tilde{\mathbf{q}}}_i$. Since \mathbf{x}_i is bounded, so is \mathbf{s}_i . According to (41), both \mathbf{s}_{pi} and \mathbf{s}_{fi} are bounded since they are orthogonal vectors. Note that \mathbf{s}_{pi} can be written as

$$\mathbf{s}_{pi} = Q_i(\mathbf{q}_i)(\dot{\tilde{\mathbf{q}}}_i + \boldsymbol{\Lambda}_i \tilde{\mathbf{q}}_i - \boldsymbol{\Lambda}_i \mathbf{z}_i) = (I - J_{\varphi_i}^+ J_{\varphi_i})(\dot{\tilde{\mathbf{q}}}_i + \boldsymbol{\Lambda}_i \tilde{\mathbf{q}}_i - \boldsymbol{\Lambda}_i \mathbf{z}_i), \quad (100)$$

in view of the definition of $Q_i(\mathbf{q}_i)$. Equation (100) can be rewritten as

$$\dot{\tilde{\mathbf{q}}}_i + \boldsymbol{\Lambda}_i \tilde{\mathbf{q}}_i = \underbrace{\boldsymbol{\Lambda}_i \mathbf{z}_i + Q_i(\mathbf{q}_i)(\dot{\tilde{\mathbf{q}}}_i + \boldsymbol{\Lambda}_i \tilde{\mathbf{q}}_i - \boldsymbol{\Lambda}_i \mathbf{z}_i)}_{\text{bounded}} - \underbrace{J_{\varphi_i}^+ J_{\varphi_i} \boldsymbol{\Lambda}_i \mathbf{z}_i}_{\text{bounded?}} + \underbrace{J_{\varphi_i}^+ J_{\varphi_i}(\dot{\tilde{\mathbf{q}}}_i + \boldsymbol{\Lambda}_i \tilde{\mathbf{q}}_i)}_{\text{bounded?}}. \quad (101)$$

One can be sure that the first terms in (101) are bounded because the case is being analyzed when \mathbf{x}_i in (63) is bounded. The conclusion can be drawn from equations (13)-(14) (boundedness of $J_{\varphi_i}^+$ and J_{φ_i}), equations (51) and (63) (boundedness of \mathbf{z}_i), and equation (100) (boundedness of $Q_i(\mathbf{q}_i)(\dot{\tilde{\mathbf{q}}}_i + \boldsymbol{\Lambda}_i \tilde{\mathbf{q}}_i - \boldsymbol{\Lambda}_i \mathbf{z}_i)$). It is then clear that if the last term of the righthand side of (101) is bounded, then $\tilde{\mathbf{q}}_i$ and $\dot{\tilde{\mathbf{q}}}_i$ must be bounded since the lefthand side of (101) represents a stable linear filter. Note then that

$$J_{\varphi_i} \dot{\tilde{\mathbf{q}}}_i = \dot{\mathbf{p}}_i - \dot{\mathbf{p}}_{di} - (J_{\varphi_i} - J_{\varphi_{di}}) \dot{\mathbf{q}}_{di} \quad (102)$$

is bounded, with $J_{\varphi_{di}} = J_{\varphi_i}(\dot{\mathbf{q}}_{di})$. Thus, the lefthand side of (101) can only be unbounded if

$$J_{\varphi_i}^+ J_{\varphi_i} \boldsymbol{\Lambda}_i \tilde{\mathbf{q}}_i = J_{\varphi_i}^+ J_{\varphi_i} \boldsymbol{\Lambda}_i (\mathbf{q}_i - \mathbf{q}_{di})$$

is not bounded. But in fact this term will be bounded as long as \mathbf{q}_i is. At this point we recall that, from Assumption 3.1, \mathbf{q}_i must satisfy the constraint $\boldsymbol{\varphi}_i(\mathbf{q}_i) = \mathbf{0}$. Furthermore, it has been assumed that none of the robots is redundant nor it is in a singularity. Thus, the

¹ Note that we do this only for formality's sake, since these two errors are bounded and equal to zero for any time in view of the fact that all \mathbf{p}_i , \mathbf{p}_{di} , $\dot{\mathbf{p}}_i$ and $\dot{\mathbf{p}}_{di}$.

constraint can only be satisfied by a finite bounded vector q_i . We can then conclude, that both \tilde{q}_i and $\dot{\tilde{q}}_i$ remain bounded as long as x_i is. To show that the tracking errors tend to zero whenever x_i does, we use the following approach. If $x_i \equiv \mathbf{0}$ then s_i becomes

$$Q_i(q_i)(\dot{\tilde{q}}_i + \Lambda_i \tilde{q}_i) + J_{\varphi_i}^+(q_i)(J_{\varphi_i}(q_i)\dot{q}_i - J_{\varphi_i}(q_{di})\dot{q}_{di} + \beta_i(\varphi_i(q_i) - \varphi_i(q_{di}))) = \mathbf{0}. \quad (103)$$

Suppose that $q_i \equiv q_{di}$, then (103) becomes

$$Q_i(q_i)\dot{\tilde{q}}_i + J_{\varphi_i}^+(q_i)J_{\varphi_i}(q_i)\dot{\tilde{q}}_i = \dot{\tilde{q}}_i = \mathbf{0}. \quad (104)$$

This shows that if $\|\tilde{q}_i\| \rightarrow 0$ then $\|\dot{\tilde{q}}_i\| \rightarrow 0$. In other words, by proving that \tilde{q}_i will tend to zero one can ensure that $\dot{\tilde{q}}_i$ will do it as well. Since homogeneous constraints are being used, the following relationship is satisfied

$$\Delta p_i = \varphi_i(q_i) - \varphi_i(q_{di}).$$

By developing a Taylor series around the desired trajectory q_{di} we have for $\varphi_i(q_i)$

$$\varphi_i(q_i) = \varphi_i(q_{di}) + \left. \frac{\partial \varphi_i}{\partial q_i} \right|_{q_i=q_{di}} (q_i - q_{di}) + \text{h. o. t.} \quad (105)$$

There exists a positive value η_i small enough such that if

$$\|q_i\| \leq \eta_i, \quad (106)$$

then the higher order terms in (105) can be neglected and the approximation

$$\Delta p_i = J_{\varphi_i}(q_{di})\tilde{q}_i, \quad (107)$$

becomes valid. The following discussion will be carried out assuming that (106) is holding. To prove that \tilde{q}_i tends to zero we require to find a dynamic equation which describes the behavior of this variable. It can be formed in the following fashion. From s_{pi} in (41) one gets

$$\tilde{q}_i^T s_{pi} = \tilde{q}_i^T \dot{\tilde{q}}_i + \tilde{q}_i^T \Lambda_i \tilde{q}_i - \tilde{q}_i^T \Lambda_i z_i - \tilde{q}_i^T \bar{P}_i \dot{\tilde{q}}_i - \tilde{q}_i^T \bar{P}_i \Lambda_i \tilde{q}_i + \tilde{q}_i^T \bar{P}_i \Lambda_i z_i, \quad (108)$$

with $\bar{P}_i \triangleq J_{\varphi_i}^+(q_i)J_{\varphi_i}(q_i)$. In view of the fact that

$$\tilde{q}_i^T \dot{\tilde{q}}_i = \|\tilde{q}_i\| \frac{d}{dt} \|\tilde{q}_i\|,$$

one can rewrite (108) as

$$\|\tilde{q}_i\| \frac{d}{dt} \|\tilde{q}_i\| = -\tilde{q}_i^T \Lambda_i \tilde{q}_i + \tilde{q}_i^T \bar{P}_i \dot{\tilde{q}}_i + \tilde{q}_i^T \bar{P}_i \Lambda_i \tilde{q}_i + \tilde{q}_i^T \Lambda_i z_i - \tilde{q}_i^T \bar{P}_i \Lambda_i z_i + \tilde{q}_i^T s_{pi}. \quad (109)$$

To develop the righthand side of this equation we analyze the term $\tilde{q}_i^T \bar{P}_i \dot{\tilde{q}}_i$ first. From (102)

$$\Delta \dot{p}_i = J_{\varphi_i} \dot{\tilde{q}}_i + (J_{\varphi_i} - J_{\varphi_{di}})\dot{q}_{di}. \quad (110)$$

By multiplying (107) by β_i and adding the result to (110) one gets

$$J_{\varphi_i} \dot{\tilde{q}}_i + \beta_i J_{\varphi_{di}} \tilde{q}_i = \Delta \dot{p}_i - (J_{\varphi_i} - J_{\varphi_{di}})\dot{q}_{di} + \beta_i \Delta p_i. \quad (111)$$

Thus we have

$$\begin{aligned}\tilde{q}_i^T \bar{P} \dot{\tilde{q}}_i &= \tilde{q}_i^T J_{\varphi_i}^T (J_{\varphi_i} J_{\varphi_i}^T)^{-1} J_{\varphi_i} \dot{\tilde{q}}_i \\ &= -\beta_i \tilde{q}_i^T \bar{P} \tilde{q}_i + \tilde{q}_i^T J_{\varphi_i}^+ \Delta \dot{p}_i + \beta_i \tilde{q}_i^T J_{\varphi_i}^+ \Delta p_i - \tilde{q}_i^T J_{\varphi_i}^+ (J_{\varphi_i} - J_{\varphi_{di}}) (\dot{q}_{di} - \beta_i \tilde{q}_i).\end{aligned}\quad (112)$$

Substituting in (109) one gets

$$\begin{aligned}\|\tilde{q}_i\| \frac{d}{dt} \|\tilde{q}_i\| &= -k_i \|\tilde{q}_i\|^2 + (k_i - \beta_i) \tilde{q}_i^T \bar{P} \tilde{q}_i - \tilde{q}_i^T J_{\varphi_i}^+ (J_{\varphi_i} - J_{\varphi_{di}}) (\dot{q}_{di} - \beta_i \tilde{q}_i) \\ &\quad + \tilde{q}_i^T J_{\varphi_i}^+ (\Delta \dot{p}_i + \beta_i \Delta p_i) + k_i \tilde{q}_i^T Q_i z_i + \tilde{q}_i^T s_{\varphi_i},\end{aligned}\quad (113)$$

where the definition $\mathbf{A}_i \triangleq k_i \mathbf{I}$ has been used. In view of Assumption 3.2, the following bound can be established

$$\left\| \tilde{q}_i^T J_{\varphi_i}^+ (J_{\varphi_i} - J_{\varphi_{di}}) (\dot{q}_{di} - \beta_i \tilde{q}_i) \right\| \leq \gamma_i \|\tilde{q}_i\|^2, \quad (114)$$

with

$$\gamma_i \triangleq c_{0i} L_i (v_{mi} + \beta_i \eta_i). \quad (115)$$

v_{mi} is a bound for the desired velocity vector \dot{q}_{di} , i. e. $\|\dot{q}_{di}\| \leq v_{mi}$ for all time, and c_{0i} and L_i are defined in (13)-(14). On the other hand, from (41) you have

$$s_{\varphi_i} = s_i - J_{\varphi_i}^+ (q_i) (\Delta \dot{p}_i + \beta_i \Delta p_i - \bar{\xi}_i \phi_i - \xi_i \Delta F_i). \quad (116)$$

By substituting (114)-(116) in (113), taking norms and dividing by $\|\tilde{q}_i\|$ one gets

$$\frac{d}{dt} \|\tilde{q}_i\| \leq -\alpha_i \|\tilde{q}_i\| + \|s_i\| + \lambda_{\max}(\bar{\xi}_i) c_{0i} \|\phi_i\| + \lambda_{\max}(\xi_i) c_{0i} \|\Delta F_i\| + k_i \|z_i\|, \quad (117)$$

with

$$\alpha_i \triangleq k_i - |k_i - \beta_i| - \gamma_i. \quad (118)$$

Note that k_i , β_i , v_{mi} and η_i , can always be chosen so as to get $\alpha_i > 0$ and that the last four elements of the righthand side of (117) are bounded since x_i is. However, (117) is valid only if (106) holds. Thus, in the end, it is not enough for x_i to be bounded. We must find a bound $x_{\max i}$ such that (106) holds. In order to guarantee this, $x_{\max i}$ should appear in (117) explicitly. Of course, if $\|x_i\| \leq x_{\max i}$, then one has

$$\|x_i\|^2 = \|s_i\|^2 + \|r_i\|^2 + \|\phi_i\|^2 + \|\Delta F_i\|^2 \leq x_{\max i}^2, \quad (119)$$

with $r_i = \dot{z}_i + \mathbf{A}_i z_i$. Since in (117) one has z_i and not r_i , it should be noted that every element of r_i is given by $r_{ij} = \dot{z}_{ij} + k_{ij} z_{ij}$, for $j = 1, \dots, n_i$. This is a stable linear filter

with gain $\frac{1}{k_i}$, i.e. one has $|z_{ij}| \leq \frac{1}{k_i} \max |r_{ij}| \quad \forall t$. Thus, it is straightforward to show that

$$\|z_i\| \leq \frac{\sqrt{n_i}}{k_i} x_{\max i}, \quad (120)$$

which allows to rewrite (117) as

$$\frac{d}{dt} \|\tilde{q}_i\| \leq -\alpha_i \|\tilde{q}_i\| + \sigma_i, \quad (121)$$

with

$$\sigma_i \triangleq \left(1 + \lambda_{\max}(\bar{\xi}_i) c_{0i} + \lambda_{\max}(\xi_i) c_{0i} + \sqrt{n_i}\right) x_{\max i}. \quad (122)$$

From the Comparison Lemma (Khalil 2002), it is

$$\|\tilde{q}_i(t)\| \leq \frac{\sigma_i}{\alpha_i} + e^{-\alpha_i t} \left(\|\tilde{q}_i(0)\| - \frac{\sigma_i}{\alpha_i} \right) \tag{123}$$

for all time. Equation (106) will be satisfied for sure if

$$\|\tilde{q}_i(0)\| \leq \frac{\sigma_i}{\alpha_i} = \frac{(1 + \lambda_{\max}(\bar{\xi}_i) c_{0i} + \lambda_{\max}(\xi_i) c_{0i} + \sqrt{n_i}) x_{\max i}}{\alpha_i} \leq \eta_i. \tag{124}$$

Equation (124) can be accomplished if

$$x_{\max i} \leq \frac{\eta_i \alpha_i}{1 + \lambda_{\max}(\bar{\xi}_i) c_{0i} + \lambda_{\max}(\xi_i) c_{0i} + \sqrt{n_i}}. \tag{125}$$

Finally, if $\|x_i\|$ tends to zero, it is clear from (117) that $\|\tilde{q}_i\|$ will do it as well. Recall that this implies the convergence to zero of $\|\dot{q}_i\|$.

- c) When x_i (and thus ΔF_i) tend to zero, $\Delta \lambda_i$ does not necessarily do it nor remains bounded. In order to prove that, one may use the fact that $J_{\phi_i}(q_i) s_i = \Delta \dot{p}_i + \beta_i \Delta p_i - \bar{\xi}_i \phi_i - \xi_i \Delta F_i = -\bar{\xi}_i \phi_i - \xi_i \Delta F_i$. The last equality is valid since constraint (6) must be satisfied. Thus you have

$$J_{\phi_i}(q_i) \dot{s}_i + \dot{J}_{\phi_i}(q_i) s_i = -\bar{\xi}_i \zeta_i - \xi_i \Delta \lambda_i = \bar{\xi}_i A_i \phi_i - \bar{\xi}_i B_i \Delta \lambda_i - \xi_i \Delta \lambda_i, \tag{126}$$

or

$$\Delta \lambda_i = -(\bar{\xi}_i B_i + \xi_i)^{-1} (J_{\phi_i}(q_i) \dot{s}_i + \dot{J}_{\phi_i}(q_i) s_i - \bar{\xi}_i A_i \phi_i), \tag{127}$$

Where (45) has been used. Since from (57) \dot{s}_i is bounded when x_i is, we conclude that $\Delta \lambda_i$ must be bounded (recall the assumption on \dot{q}_i). Finally, if x_i tends to zero then \dot{s}_i , s_i and ϕ_i do and we arrive to the conclusion that $\Delta \lambda_i$ tends to zero as well.

C Positive definiteness of $V_i(x_i)$ in (65)

To prove that $V_i(x_i) = \frac{1}{2} x_i^T (\text{block diag} \{H_i(q_i), H_i(q_i), R_i\}) x_i$ in (65) is actually a positive definite function, recall first that such a function must satisfy that $V_i(0) = 0$ and $V_i(x_i) > 0$ for $x_i \neq 0$. Since $H_i(q_i)$ is a positive definite matrix, the single problem is that R_i in (66) is only semipositive definite. However, developing the term in (65) which involves R_i yields

$$\frac{1}{2} \begin{bmatrix} \phi_i^T & \Delta F_i^T \end{bmatrix} R_i \begin{bmatrix} \phi_i \\ \Delta F_i \end{bmatrix} = \frac{1}{2} \left(B_i^{\frac{1}{2}} \phi_i - B_i^{\frac{1}{2}} \Delta F_i \right)^T N_i \left(B_i^{\frac{1}{2}} \phi_i - B_i^{\frac{1}{2}} \Delta F_i \right), \tag{128}$$

since B_i is a positive definite diagonal matrix. In view of the fact that $N_i > 0$, (128) can only be zero if

$$\phi_i = B_i \Delta F_i. \tag{129}$$

On the other hand, note that $V_i(x_i)$ includes a term $\frac{1}{2} s_i^T H_i(q_i) s_i$ which can be zero only if $s_i = 0$. Otherwise, it is positive. But from (41) you have $s_i = s_{pi} + s_{fi}$, where s_{pi} and s_{fi} are

orthogonal, so that they both must be zero in order for S_i to be zero. In particular, one has $s_i = J_{\phi_i}^+(q_i)(\Delta\dot{p}_i + \beta_i\Delta p_i - \bar{\xi}_i\phi_i - \xi_i\Delta F_i)$, with $J_{\phi_i}^+(q_i)$ full rank. Then, in view of constraint (6), you have that $S_{\bar{f}_i}$ becomes zero only if $\bar{\xi}_i\phi_i + \xi_i\Delta F_i$ does, that is if

$$\phi_i = -\bar{\xi}_i^{-1}\xi_i\Delta F_i. \quad (130)$$

Comparing (129) with (130) one concludes that $\frac{1}{2}s_i^T H_i(q_i)s_i$ and (128) cannot be simultaneously zero unless both ϕ_i and ΔF_i are, because B_i , $\bar{\xi}_i$ and ξ_i are all positive definite diagonal matrices. In other words, $V_i(x_i) = 0$ if and only if $x_i = 0$. Thus, it is a (continuous) positive definite function.

D Proof of Theorem 4.1

Recall the following well known theorem (Khalil 2002, pp. 100)

Theorem D.1 *Let $x = 0$ be an equilibrium point for $\dot{x} = f(x)$ and $\mathbb{D} \subset \mathbb{R}^n$ be a domain containing $x = 0$. Let $V : \mathbb{D} \rightarrow \mathbb{R}$ be a continuously differentiable function, such that $V(0) = 0$ and $V(x) > 0$ in $\mathbb{D} - \{0\}$, and $\dot{V}(x) \leq 0$ in \mathbb{D} . Then, $x = 0$ is stable. Moreover, if $\dot{V}(x) < 0$ in $\mathbb{D} - \{0\}$ then $x = 0$ is asymptotically stable.* Δ

To take advantage of Theorem D.1, we just have to find domains \mathbb{D}_i for which each $V_i(x_i)$ satisfies $\dot{V}_i(x_i) < 0$ in $\mathbb{D}_i - \{0\}$ (because each $V_i(x_i)$ is positive definite in \mathbb{R}^n). In doing so, one can prove that $x_i \rightarrow 0$ for all i . Then, Lemma 4.1 can be employed to analyze the behavior of the different error signals. Based on the discussion given in Appendix B, we define each domain \mathbb{D}_i as

$$\mathbb{D}_i = \{x_i \in \mathbb{R}^n \mid \|x_i\| \leq x_{\max i}\}, \quad (131)$$

where $x_{\max i}$ is chosen to satisfy (125) and cannot be done arbitrarily large. In \mathbb{D}_i one can define

$$\mu_{1i} \triangleq \max_{\|q_i\| \leq x_{\max i}} \|C_i(q_i, \dot{q}_i)\| \quad (132)$$

$$\mu_{2i} \triangleq \max_{\|q_i\| \leq x_{\max i}} \|C_i(q_i, s_i + \dot{q}_i)\| \quad (133)$$

$$\mu_{3i} \triangleq \max_{\|q_i\| \leq x_{\max i}} \|C_i(q_i, s_i + 2\dot{q}_i)\| \quad (134)$$

$$\mu_{4i} \triangleq M_{ii}(x_{\max i})\lambda_{10} \quad (135)$$

$$\lambda_{20i} \triangleq \lambda_{\max}(D_i) \quad (136)$$

Note that it is straightforward to compute (132)-(136) as functions of the different constants defined throughout the paper, but we skip it for simplicity's sake. The next step is to compute the derivative of the Lyapunov function candidate in (65), which can be rewritten as

$$V_i(x_i) = \frac{1}{2}s_i^T H_i(q_i)s_i + \frac{1}{2}r_i^T H_i(q_i)r_i + \frac{1}{2}[\phi_i^T \quad \Delta F_i^T] \begin{bmatrix} N_i B_i^{-1} & -N_i \\ -N_i & N_i B_i \end{bmatrix} \begin{bmatrix} \phi_i \\ \Delta F_i \end{bmatrix}.$$

Then, the derivative of (65) along (42), (45), (57) and (62) can be computed as

$$\begin{aligned}
\dot{V}_i(x_i) &= \frac{1}{2} \dot{s}_i^T \dot{H}_i(q_i) s_i + \frac{1}{2} r_i^T \dot{H}_i(q_i) r_i - s_i^T C_i(q_i, \dot{q}_i) s_i - s_i^T K_{DR} s_i \\
&+ s_i^T K_{Ri} r_i + s_i^T J_{\phi i}^T(q_i) (B_i^{-1} A_i \phi_i + K_{Fi} \Delta F_i) - s_i^T C_i(q_i, \dot{q}_{ri}) s_i \\
&+ s_i^T H_i(q_i) e_i(r_i) - r_i^T C_i(q_i, \dot{q}_i) r_i - r_i^T H_{rdi} r_i + r_i^T C_i(q_i, s_i + \dot{q}_{ri}) r_i \\
&- r_i^T C_i(q_i, s_i + 2\dot{q}_{ri}) s_i - r_i^T K_{DR} s_i + r_i^T J_{\phi i}^T(q_i) (B_i^{-1} A_i \phi_i + K_{Fi} \Delta F_i) \\
&+ \begin{bmatrix} \phi_i^T & \Delta F_i^T \end{bmatrix} \begin{bmatrix} N_i B_i^{-1} & -N_i \\ -N_i & N_i B_i \end{bmatrix} \begin{bmatrix} \zeta_i \\ \Delta \lambda_i \end{bmatrix}.
\end{aligned} \tag{137}$$

To simplify (137) one should take into account that $s_i^T K_{Ri} r_i - s_i^T K_{DR} r_i = -s_i^T D_i r_i$ and that $s_i^T J_{\phi i}^T(q_i) (B_i^{-1} A_i \phi_i + K_{Fi} \Delta F_i) = -\phi_i^T \bar{\xi}_i B_i^{-1} A_i \phi_i - \phi_i^T (\bar{\xi}_i K_{Fi} + \xi_i B_i^{-1} A_i) \Delta F_i - \Delta F_i^T \xi_i K_{Fi} \Delta F_i$. The last equality is valid in view of Property 3.5, from (41) and by the fact that constraint (6) must be satisfied for p_i , \dot{p}_i , p_{di} and \dot{p}_{di} . Furthermore, one has

$$\begin{bmatrix} \phi_i^T & \Delta F_i^T \end{bmatrix} \begin{bmatrix} N_i B_i^{-1} & -N_i \\ -N_i & N_i B_i \end{bmatrix} \begin{bmatrix} \zeta_i \\ \Delta \lambda_i \end{bmatrix} = -\phi_i^T N_i B_i^{-1} A_i \phi_i + \Delta F_i^T N_i A_i \phi_i$$

in view of (45). Recalling that $N_i = (\xi_i B_i^{-1} A_i + \bar{\xi}_i K_{Fi}) A_i^{-1}$ and defining $E_i \square \bar{\xi}_i B_i^{-1} A_i + \xi_i B_i^{-1} A_i + \bar{\xi}_i K_{Fi} B_i^{-1}$, (137) can be simplified to

$$\begin{aligned}
\dot{V}_i(x_i) &\leq -s_i^T K_{Ri} s_i - r_i^T H_{rdi} r_i - \phi_i^T E_i \phi_i - \Delta F_i^T \xi_i K_{Fi} \Delta F_i - s_i^T D_i r_i + s_i^T H_i(q_i) e_i(r_i) \\
&- r_i^T C_i(q_i, s_i + 2\dot{q}_{ri}) s_i + r_i^T J_{\phi i}^T(q_i) B_i^{-1} A_i \phi_i + r_i^T J_{\phi i}^T(q_i) K_{Fi} \Delta F_i \\
&- s_i^T C_i(q_i, \dot{q}_{ri}) s_i + r_i^T C_i(q_i, s_i + \dot{q}_{ri}) r_i,
\end{aligned} \tag{138}$$

by taking Property 3.2 into account. Since we are only interested in the behavior of $\dot{V}_i(x_i)$ for x_i in D_i , we have from (132)-(136)

$$\begin{aligned}
\dot{V}_i(x_i) &\leq -\lambda_{\min}(K_{Ri}) \|s_i\|^2 - \lambda_{\min}(H_{rdi}) \|r_i\|^2 - \lambda_{\min}(E_i) \|\phi_i\|^2 - \lambda_{\min}(\xi_i K_{Fi}) \|\Delta F_i\|^2 \\
&+ \mu_{1i} \|s_i\|^2 + \mu_{2i} \|r_i\|^2 + (\lambda_{Di} + \mu_{3i} + \mu_{4i}) \|s_i\| \|r_i\| \\
&+ c_{1i} b_i a_i \|r_i\| \|\phi_i\| + c_{1i} \lambda_{\max}(K_{Fi}) \|r_i\| \|\Delta F_i\|,
\end{aligned} \tag{139}$$

where c_{1i} is given in (14) and

$$b_i \triangleq \max\{b_{ij}^{-1}\} \tag{140}$$

$$a_i \triangleq \max\{a_{ij}\} \tag{141}$$

a_{ij} and b_{ij} , with $j = 1, \dots, n_i$, are elements of A_i and B_i , respectively. The next step is to choose the different gains to guarantee that $\dot{V}_i(x_i) < 0$ in $D_i - \mathbf{0}$. First of all, consider $\lambda_{\min}(K_{Ri})$ in (70) and k_{di} in (71), such that one has from the first line of (139)

$$\begin{aligned}
&- (\mu_{1i} + 1 + \delta_i) \|s_i\|^2 - \left(\lambda_{hi} \left(\frac{\lambda_{\max}(K_{Ri}) + w_i}{\lambda_{hi}} \right) - \lambda_{\max}(K_{Ri}) \right) \|r_i\|^2 \leq \\
&- \delta_i \|s_i\|^2 - \delta_i \|r_i\|^2 - \left(\frac{1}{4} c_{1i}^2 a_i^2 b_i^2 + \frac{1}{4} c_{1i}^2 \lambda_{\max}^2(K_{Fi}) \right) \|r_i\|^2,
\end{aligned}$$

because $w_i = \mu_{2i} + \frac{1}{4} (\lambda_{Di} + \mu_{3i} + \mu_{4i})^2 + \delta_i + \frac{1}{4} c_{1i}^2 a_i^2 b_i^2 + \frac{1}{4} c_{1i}^2 \lambda_{\max}^2(K_{Fi})$. Thus, (139) becomes

$$\begin{aligned} \dot{V}_i(x_i) \leq & -\delta_i \|s_i\|^2 - \delta_i \|r_i\|^2 - \left(\frac{1}{4} c_{1i}^2 a_i^2 b_i^2 + \frac{1}{4} c_{1i}^2 \lambda_{\max}^2(K_{Fi}) \right) \|r_i\|^2 \\ & - \lambda_{\min}(E_i) \|\phi\|^2 - \lambda_{\min}(\mathcal{G}_i K_{Fi}) \|\Delta F_i\|^2 \\ & + c_{1i} a_i b_i \|r_i\| \|\phi\| + c_{1i} \lambda_{\max}(K_{Fi}) \|r_i\| \|\Delta F_i\|. \end{aligned} \quad (142)$$

Finally, by considering $\lambda_{\min}(E_i)$ in (72) and $\lambda_{\min}(\mathcal{G}_i K_{Fi})$ in (73) it is easy to conclude that

$$\dot{V}_i(x_i) \leq -\delta_i \|s_i\|^2 - \delta_i \|r_i\|^2 - \delta_i \|\phi\|^2 - \delta_i \|\Delta F_i\|^2, \quad (143)$$

or

$$\dot{V}_i(x_i) \leq -\delta_i \|x_i\|^2. \quad (144)$$

By applying Theorem D.1 one concludes that $x_i \rightarrow 0$. Now, from definition (51) one has directly

$$\lim_{t \rightarrow \infty} z_i = 0 \quad \lim_{t \rightarrow \infty} \dot{z}_i = 0.$$

Furthermore, in view of (131) one has $\|x_i\| \leq x_{\max i}$ (and thus $\|\tilde{q}_i\| \leq \eta_i$ from the discussion in Appendix B). Thus, from Lemma 4.1 a) and b), we get

$$\lim_{t \rightarrow \infty} \Delta \dot{p}_i = 0 \quad \lim_{t \rightarrow \infty} \Delta p_i = 0 \quad \lim_{t \rightarrow \infty} \dot{\tilde{q}}_i = 0 \quad \lim_{t \rightarrow \infty} \tilde{q}_i = 0.$$

To applied c) of Lemma 4.1, we only need to show that $\dot{\tilde{q}}_i$ is bounded. This is certainly the case because \tilde{q}_i and \dot{q}_{di} are bounded. Thus we get

$$\lim_{t \rightarrow \infty} \Delta \mathcal{L}_i = 0.$$

Finally, the stability of the whole system can be proven using

$$V = \sum_{i=1}^l V_i(x_i).$$

It should be noted that a region of attraction has not been given explicitly. However, it is a subset of D_i^- and cannot be made arbitrarily large because of the fact that $\|x_i\| \leq \eta_i$ must hold for all time (and thus $\|x_i\| \leq x_{\max i}$ must be valid as well), with η_i small enough.

7. References

- Arteaga Pérez, M. (1998). On the properties of a dynamic model of flexible robot manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control*. Vol. 120, 8-14.
- Arteaga Pérez, M. & R. Kelly. (2004). Robot control without velocity measurements: New theory and experimental results. *IEEE Transactions on Robotics and Automation*. Vol. 20, No. 2, 297-308.
- Baumgarte, J. (1972). Stabilization of constraints and integrals of motion in dynamical systems. *Comput. Meth. Appl. Mech. Eng.* Vol. 1, 1-16.
- Cole, A. (1990). Constrained motion of grasped objects by hybrid control. *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*. 1954-1960.
- Cole, A., P. Hsu & S. Sastry. (1992). Dynamic control of sliding by robot hands for *regrasping*. *IEEE International Conference on Robotics and Automation*. Vol. 8, 42-52.

- Gudiño-Lau, J., M. Arteaga Pérez, L. Muñoz & V. Parra-Vega. 2004. On the control of cooperative robots without velocity measurements. *IEEE Transactions on Control Systems Technology*. Vol. 12, No. 4, 600–608.
- Gudiño-Lau, J. & M. Arteaga Pérez. 2005. Dynamic model and simulation of cooperative robots: a case study. *ROBOTICA*. Vol. 23, 615–624.
- Jankowski, K., H. ElMaraghy & W. ElMaraghy. 1993. Dynamic coordination of multiple robot arms with flexible joints. *International Journal of Robotics Research*. Vol. 12, No. 6, 505–528.
- Khalil, H. (2002). *Nonlinear Systems Third Edition*. U.S.A.: Prentice Hall. 746p.
- Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*. Vol. 3, 43–53.
- Kuc, T., J. Lee & B. Park. (1994). Learning control of cooperating robot manipulators handling an unknown object. *IEEE International Conference on System*. Vol. 3, 3354–3339.
- Li, z. & S. Sastry. (1989). Grasping and coordinated manipulation by a multifingered robot hand. *International Journal of Robotics Research*. Vol. 8, No. 4, 33–50.
- Liu, G., J. Li & Z. Li. (2002). Coordinated manipulation of objects by multifingered robotic hand in contact space and active joint space. *IEEE International Conference on Robotics and Automation*. Vol. 4, 3743–3748.
- Liu, Y., S. Arimoto, V. Parra-Vega & K. Kitagaki. (1997). Decentralized adaptive control of multiple manipulators in cooperations. *International Journal of Control*. Vol. 67, No. 5, 649–673.
- Mason, M. & J. Salisbury. (1985). *Robot Hands and the Mechanics of Manipulation*. London: The MIT Press.
- Montana, D. (1988). The kinematics of contact and grasp. *International Journal of Robotics Research*. Vol. 7, No. 3, 743–748.
- Murray, R., Z. Li & S. Sastry. (1994). *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, Florida, U. S. A.: CRC Press.
- Naniwa, T., S. Arimoto & K. Wada. (1997). Learning and adaptive controls for coordination of multiple manipulators holding a geometrically constrained object. *IEEE International Conference on Intelligent Robots*. Vol. 1, 484–494.
- Orin, D. & S. Oh. (1981). Control of force distribution in robotic mechanisms containing closed kinematics chains. *ASME Journal of Dynamic Systems, Measurement, and Control*. Vol. 102, 134–141.
- Parra-Vega, V. & S. Arimoto. (1996). A passivity-based adaptive sliding mode position-force control for robot manipulators. *International Journal of Adaptive Control and Signal Processing*. Vol. 10, 365–377.
- Parra-Vega, V., A. Rodríguez-Ángeles, S. Arimoto & G. Hirzinger. (2001). High precision constrained grasping with cooperative adaptive control. *Journal of Intelligent and Robotic System*. Vol. 32, 235–254.
- Schlegel, T., M. Buss, T. Omata & G. Schmidt. (2001). Fast dextrous regrasping with optimal contact forces and contact sensor based impedance control. *IEEE Int. Conf. on Robotics and Automation*. 103–109.

-
- Sciavicco, L. & B. Siciliano. (2000). *Modeling and Control of Robot Manipulators*. 2th Edition: McGraw-Hill.
- Yoshikawa, T. & X. Zheng. (1991). Coordinated dynamic control for multiple robotic mechanisms handling an object. *IEEE/RSJ International Workshop on Intelligent Robots and Systems*. 315-320.

Biologically-Plausible Reactive Control of Mobile Robots

Zapata René, Lépinay Pascal
Université Montpellier II
France

1. Introduction

In many robotic tasks, a mobile robot has to perform four subtasks constituting a hierarchy from high level to low level actions:

- Path planning
- Localization
- Target pursuit
- Collision avoidance

The last 3 tasks which obviously need perception capabilities and vision (like for natural beings) can provide the robot with very rich information. Unfortunately, artificial vision algorithms present the major disadvantage of being difficult to process in real time (for instance, the collision avoidance process typical time is less than 0.01s). Therefore including real time vision in a robot control loop system requires hardware implementation. But efficient reactive behaviours can also be performed with simpler sensors, like range sensors (Ultrasonic sensors for instance).

Biologically-plausible algorithms spring from the observed natural systems especially through analogies between artificial perception/actions systems and natural ones (Berthoz and Weiss, 2000). Many papers deal with this subject and whole conferences are devoted to it (see for instance SAB proceedings (SAB, 2003)). Neural network theories are an important part of this research and much work has been done on neural aspects of artificial vision. For instance, two recent works are closely correlated with biological experiments (Wurtz and Lourens, 2000) (VanRullen, 2002).

This chapter is divided into 4 parts. The first section describes a reactive control algorithm based on the modelling of the robot/environment interaction. Like Potential fields methods (Kathib 1985) or (Holenstein and Badreddin, 1991), this approach can present the problem of local minima. However, it turns out that it is quite robust to unknown and dynamic worlds. The second section (section 3) describes two vision-based algorithms: 3D reconstruction and image segmentation, which will be the basis of the implementations proposed in the last section of the chapter. Section 4 presents several simulation experiments illustrating the two previous issues. Section 5 describes the implementation of fast vision-based algorithms on configurable architectures.

2. A bio-plausible reactive control algorithm

This section describes a bio-plausible reactive control algorithm for obstacle avoidance and target pursuit in an unstructured and dynamic world. This algorithm was first described in (Zapata et al., 1994) for many applications and tested for robots moving in 2 dimensions, flying robots or autonomous submarines and also mobile manipulators (Cacitti, 2000). This algorithm that was initially designed for obstacle avoidance, has two main advantages. First, the environment does not need to be *a priori* known, and second, the controller can take into account other constraints such as target pursuit, altitude maintaining and course control.

2.1 General principle

The reactive control algorithm we use is based on the definition of a protecting and deformable zone surrounding the robot. This DVZ (*Deformable Virtual Zone*) is parameterized by the motion variables of the moving robot and can deform in the presence of distance information in the robot workspace. When an obstacle enters the sensor space, it induces a deformation of the DVZ that will be compensated by the robot motion controller. Therefore, the algorithm is a kind of 2-player game: the first one, i.e. the environment, induces undesired deformations; the second one, i.e. the robot controller, tries to rebuild the DVZ.

Figure 1 illustrates this general principle that will be described in paragraph 2.2.

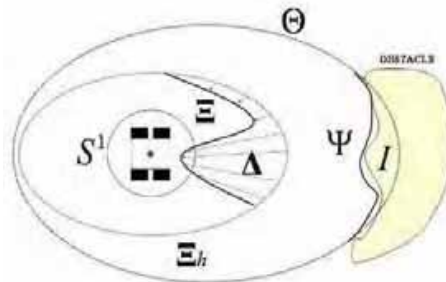


Fig. 1. The DVZ principle in 2D.

2.2 Mathematical basis

The general framework for formalizing this principle is the category D of topologically equivalent sets of the $(n-1)$ -dimensional unitary sphere $S^{n-1}(0,1)$ in \mathbb{R}^n . An object A of this category is related to the unitary sphere through an homeomorphism (imbedding of the sphere):

$$\delta_i : S^{n-1}(0,1) \rightarrow A \subset \mathbb{R}^n \quad (1)$$

The transformations between two objects A and B of D are the deformations obtained by the combination of the two defining homeomorphisms:

$$\delta_B \circ \delta_A^{-1} : A \rightarrow B \quad (2)$$

Let $R \subset \mathbb{R}^n$ be a convex rigid body, subset of the n dimensional-space \mathbb{R}^n . The boundary ∂R of R can also be considered as the result of an imbedding of $S^{n-1}(0,1)$ in \mathbb{R}^n . We have $\partial R \in D$ and:

$$\delta_R : S^{n-1}(0,1) \rightarrow \partial R \subset \mathbb{R}^n \quad (3)$$

Reciprocally, if f is an imbedding of ∂R in \mathbb{R}^n with $E = f(\partial R)$, it is to say that:

$$f : \partial R \rightarrow E \subset \mathbb{R}^n \quad (4)$$

then : $E \in D$ by the choice: $\delta_f = f \circ \delta_R$.

Any object $A \in D$ separates \mathbb{R}^n in two connected components, the interior $Int(A)$ of A and the exterior $Ext(A)$ of A . Therefore we have $\mathbb{R}^n = Int(A) \oplus Ext(A) \oplus A$. A partial order is induced on D by the relation:

$$A < B \Leftrightarrow Int(A) \subset Int(B) \quad (5)$$

The rigid body R will represent a controlled robot moving among obstacles in \mathbb{R}^n . Any n -dimensional state vector characterizing the motion of R (e.g.translational and rotational velocities) is denoted as a vector:

$$\pi = [\rho_1 \ \rho_2 \ \dots \ \rho_n]^T \quad (6)$$

In the following, we will assume that the robot R can be controlled by the derivative of this state vector. We note:

$$\dot{\phi} = \dot{\pi} \quad (7)$$

We define a DVZ of R as any imbedding Ξ of ∂R in \mathbb{R}^n such that the relation $\partial R < \Xi(\partial R)$ holds. We have $\Xi(\partial R) \in D$.

We define a *controlled DVZ*, Ξ_h as a DVZ which depends on the state vector characterizing the motion of R :

$$\Xi_h = \rho(\pi) \quad (8)$$

Let $P = (\Xi_h, \Xi)$ be a pair of two DVZ of R (the first one being a controlled DVZ) and such that $\Xi(\partial R) < \Xi_h(\partial R)$. We define the deformation Δ of the DVZ Ξ_h with respect to Ξ as the functional difference of Ξ and Ξ_h :

$$\Delta = \Xi - \Xi_h \quad (9)$$

According to this definition, the deformation Δ is a one-one map that associates the vector $P - P_h$ to the point $M \in \partial R$, where $P = \Xi(M)$ and $P_h = \Xi_h(M)$. It can therefore be considered as a vector field defined on ∂R .

We also assume that the robot can perceive distances in all directions of space and that the set of maximum distances that can be perceived by R and the set of actually perceived

distances are two objects of the category D , respectively named *Sensor boundary* and *Information boundary* (respectively denoted Θ and Ψ), such that $\Psi \prec \Theta$. The deformation I of Θ with respect to Ψ is given by:

$$I = \Psi - \Theta \quad (10)$$

The deformation I can also be considered as a vector field on ∂R .

We define an *uncontrolled DVZ*, Ξ , as a DVZ which depends on the Sensor boundary deformation I :

$$\Xi = \beta(I) \quad (11)$$

Let $P = (\Xi_h, \Xi)$ be a pair composed of a controlled DVZ and an uncontrolled DVZ, the deformation Δ of the DVZ Ξ_h with respect to Ξ can be written:

$$\Delta = \Xi - \Xi_h = \beta(I) - \rho(\pi) \quad (12)$$

For a given point $M \in \partial R$, the deformation vector $\Delta(M)$ depends on the intrusion of proximity information $I(M)$, in the rigid body workspace, and on the controlled DVZ Ξ_h .

By differentiating equations 12 with respect to time, we get:

$$\dot{\Delta} = -\beta'(I)\psi - \rho'(\pi)\phi \quad (13)$$

where f' represents the derivative of function f . This equation can be rewritten as:

$$\dot{\Delta} = A\phi + B\psi \quad (14)$$

Variations in Δ are controlled by 2 input vectors ϕ (due to the robot controller, tends to minimize deformation of the DVZ) and ψ (unknown and induced by the environment itself).

In order to control the reactive behaviours of the robot, the desired variation of this deformation is chosen as a function of the real deformation and its derivative:

$$\dot{\Delta}_{des} = -K_{prop}\Delta - K_{der}\dot{\Delta} \quad (15)$$

where K_{prop} and K_{der} are heuristically chosen. The computation of the best control vector $\check{\phi}$ at time t obtained by inverting equation 14 after replacing the deformation derivative by its desired value $\dot{\Delta}_{des}$:

$$\check{\phi} = A^+ (\dot{\Delta}_{des} - B\hat{\psi}) \quad (16)$$

where A^+ is the pseudo-inverse of the linear function A and $\hat{\psi}$ is an estimation of the second control vector Ψ at time t obtained at time $t-1$:

$$B\hat{\psi} = \dot{\Delta}_{measured}(t-1) - A\phi(t-1) \quad (17)$$

This control law is illustrated in figure 2.

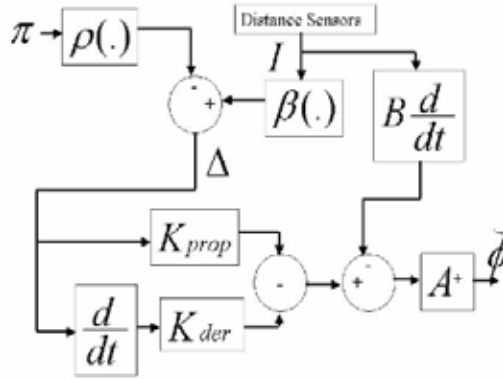


Fig. 2. The DVZ control loop.

Important remarks:

- The control law (Equation 16) tends to minimize the function $\|\Delta_{des} - \Delta\|$ in the Least Squares sense.
- The ∞ -dimensional functional equation 14 cannot, of course, be used directly. It is necessary to sample the sensor space in order to obtain an n -dimensional definition of the DVZ. This can be done by considering that the information vector has n dimensions (as many as the number of distance sensors). Equation 14 keeps its general form but all its entries are now matrices or vectors. For instance, let's consider the simple case of a car-like robot equipped with 32 distance sensors. In this case, the functions I , Δ , $\dot{\Delta}$, $\dot{\Delta}_{des}$, Ξ_h and Ξ are 32-dimensional vectors. The vector π is a 2-dimensional vector and function A is a (32×2) -dimensional matrix. The matrices K_{prop} and K_{der} are (32×32) matrices.

2.3 Implementation

Algorithm

The reactive control is implemented as follows:

- (i) Measurement of the intrusion of information I as an n -dimensional vector (as many dimensions as the number of sensors)
- (ii) Derivation of the deformation Δ and of its derivative $\dot{\Delta}$
- (iii) Estimation of the uncontrolled control vector $\tilde{\psi}$
- (iv) Computation of the best control vector $\tilde{\phi}$

Regarding the target pursuit algorithm in the presence of obstacles, we have the same algorithm except that in this case the deformation vector includes lines of vector components for measuring the deformation between a desired state (robot on target for instance) and the real state (robot at a measured distance from the target):

$$\begin{bmatrix} \Delta_{avoid} \\ \Delta_{target} \end{bmatrix}$$

Practical implementation of the function β

In the rest of the paper, we will assume that the function β relating the intrusion of information to the deformed DVZ Ξ_i is defined as follows:

$$\beta(I) = \begin{cases} \Theta + I & \text{if } \Theta + I < \Xi_h \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

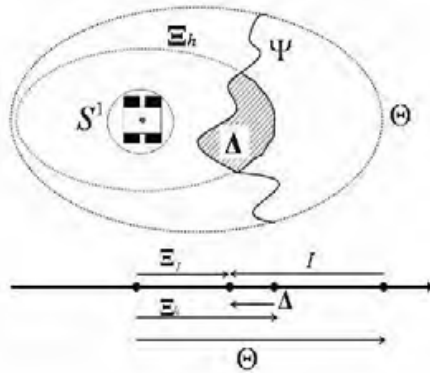


Fig. 3. Computation of function β

where the sign $<$ means the comparison of the sensor information $\Psi = \Theta + I$ and the undeformed DVZ Ξ_h in each direction of measure (see figure 3). This restriction does not change the principle of the DVZ and is just a simplification leading to the confusion of the intrusion I and the deformation Δ (their derivatives are now identical)

3. Vision feedback

In this section, we will discuss two vision-based algorithms: 3D reconstruction and image segmentation, which will be the basis of the implementations proposed in the last section of the paper. For these two algorithms, it is obvious that vision can be helpful way for generating the necessary information.

3.1 Vision feedback for collision avoidance

For the collision avoidance scheme, the main input is the distance field in the robot front space. Stereovision allows this 3D reconstruction by measuring the disparity field in two images.

A very good review of advances in stereovision can be found in (Brown et al., 2003). The authors address three basic topics: correspondence methods (local and global), occlusion problems and real-time implementations. This paper lists more than thirteen references in this last field from 1993 to present. The technologies used range from single off-the-shelf processors to custom FPGAs (Faugeras and al, 1993), (Kimura et al., 1999) and DSPs (Beymer and Konolige, 1999).

The vision-based stereo-matching algorithm for distance estimation and collision avoidance involves spatial stereo-matching of n characteristic points of two images (left and right) and provides a distance field in the robot front-space. This distance field induces the DVZ. The complete algorithm has 7 steps:

- (i) right The right image is divided into regular zones in order to cover the whole front space of the mobile robot. A particular zone of this image called *fovea*, is subdivided into 16 smaller zones in order to increase the perception capabilities of the vision system in a given direction like in biological systems
- (ii) A set of interesting points (e.g. high weighted gradient) is chosen in the right image. A subset of characteristic points is derived from this set, containing one point per zone. Each of these points has the strongest gradient in the corresponding zone. Mathematically, this set is characterized by the positions of these characteristic points in the image space (pixels)
- (iii) A stereo-matching algorithm, based on the evaluation of a distance between a (11x11) patch around the characteristic points and a (11x11) patch moving in the left image, tries to find the correspondences between these 2 images. A counter correlation is then run to confirm the stereo-pairs. The disparity between the characteristic points provides a distance map (vector l) to the robot main computer (figure 4)
- (iv) The controlled DVZ $\Xi_h(t) = \rho(\bar{\pi}, t)$ is computed
- (v) The complete DVZ $\Xi(t) = \Xi_h(t) + \Delta(t)$ is computed
- (vi) The control vector $\check{\phi}$ is computed
- (vii) The DVZ orientation induces re-computing of the position of the fovea in the image for time $t+1$

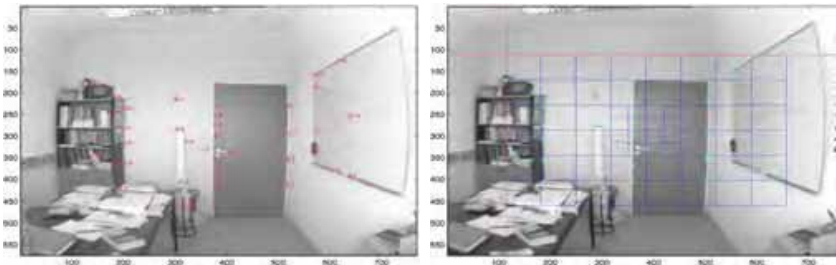


Fig. 4. Left and right images and the distance field on the left image.

3.2 Vision feedback for target pursuit

For the target pursuit scheme, the visual recognition of the target can also involve a visual system. To recognize, or at least to follow, a given pattern in the robot space is a crucial issue in mobile robotics. For instance, in target pursuit tasks, the target has a very special role and its localization, and the measurement of its velocity or its acceleration is essential. Segmentation of a colour image can provide this information. In the other ego-localization example (for path planning purposes), clues in the robot environment can also be detected by colour segmentation. Even for obstacle detection, image segmentation can help in the determination of characteristic points before 3D reconstruction.

We have developed a segmentation algorithm based on two-layer Probabilistic Neural Networks (PNN) and on Mathematical Morphology tools. As summarized here, this algorithm is a 2-step process:

- *Segmentation*: Each neuron of the first layer of the PNN is dedicated to a given user based colour. Its weight W is a point in the RGB space. Each *colour-prototype*, is therefore represented by several neurons in the network. When a pixel is read, its distance to W is computed and its probability of belonging to the neuron class is computed through a normalized Gaussian function. Then its probability of belonging to a given colour-prototype is computed by a combination of all neural probabilities (as many as the number of neurons of the first layer). The next step carried out by the second PNN layer is the competition of all colour-prototype probabilities. At the end of the segmentation process, the original image has been replaced by a labelled image consisting of as many labels as there are different prototypes.
- *Morpho-Filtering*: Each theme can be transformed in a black and white image ($v=1$ for a pixel belonging to the prototype, $v=0$ if not). In order to clean this image, it is then eroded by an erosion process and dilated by a dilatation process. In a second step, the resulting image is labelled in order to detect all objects belonging to the theme.

4. Simulation results

4.1 Collision avoidance and target pursuit

A graphic simulator was developed with MATLAB for experimenting the reactive collision avoidance process based on the DVZ algorithm from either a computational and behavioral point of view. The simulator constitutes an interface that allows us to interactively change the position of the obstacles, the intrinsic parameters of the DVZ, and initial states. It permits as well to visualize the 2-dimensional motion of the robot and the geometry of the DVZ.

The mobile robot presented here is a car-like vehicle equipped with 32 simulated ultrasonic sensors. Next figures show a few results provided by this simulator. In all these cases, the target is virtual: the robot must maintain a constant velocity and a zero-angle for its front wheels

- Left part of figure 5 compares the trajectories between a primary action (Trying to keep [wheel direction=0] and [velocity=5m.s⁻¹]) without collision avoidance versus the same task with collision avoidance of a wall.
- In right part of figure 5 the mobile robot avoids a frontal wall that should have induced a singularity of the DVZ. The noise in sensors avoids this symmetry. To carry out this task, we defined the primary task as before (Keeping [wheel direction=0] and [velocity=4m.s⁻¹]) and turned on the collision avoidance task.
- In left part of figure 6 the mobile robot follows a corridor. To carry out this task, we defined the primary task as before (Keeping [wheel direction=0] and [velocity=3m.s⁻¹]) and turned on the collision avoidance task.
- In right part of figure 6 the mobile robot runs in a complex world up to 4m.s⁻¹. To carry out this task, we defined the primary task as before (Keeping [wheel direction=0] and [velocity=5m.s⁻¹]) and turned on the collision avoidance task.

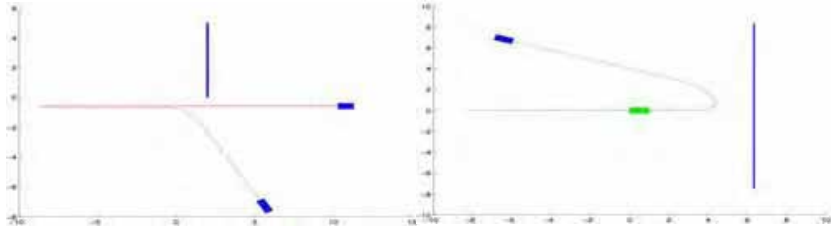


Fig. 5. Avoiding a wall.

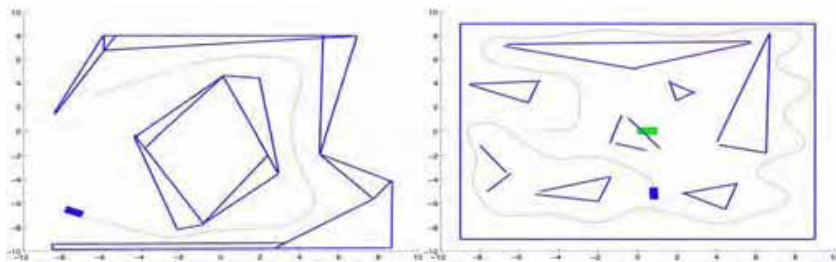


Fig. 6. Moving in a corridor and in a clustered environment.

4.2 Vision feedback for collision avoidance

We tested the spatial stereo-matching algorithm on several pairs of images taken in unknown environments (rooms, corridors, outdoor). The algorithm divides each image into 60 zones (44 for global vision +16 for the mobile fovea) and chooses one point of interest per zone (when there is one). We carried out several experiments with different pairs of images and noticed that the number of characteristic points was not constant, due to the existence of uninteresting zones in the image (no significant gradient). For interesting points (with a strong gradient), if the stereo-matching algorithm reveals a good correlation between the left and right images, the distance of the point is computed (with a pin-hole camera model), and therefore its position in the real robot space is known. Otherwise, this distance is set to infinity. A visual check of the results indicated a success percentage from 90 to 100 percent (Figure 7):

- Number of points: 56
- Number of matching: 51
- Percentage of success: 91%



- Number of points: 46
- Number of matching: 48
- Percentage of success: 95.8%



- Number of points : 37
- Number of matching: 36
- Percentage of success : 97%



Fig. 7. Distance fields in gray level images.

4.2 Vision feedback for target pursuit

We also tested the image segmentation algorithm on several colour images. A simulator was developed for designing different PNNs that are run on colour images. The user decides how many colour-prototypes are included in the network and how many prototypes (neurons) for each colour-prototype. This structure allows testing of several classifiers, depending on the chosen prototype. For instance, it is possible to create a PNN to separate dark objects from light objects (2 prototypes), or to build one to detect a red object among others, and so on.

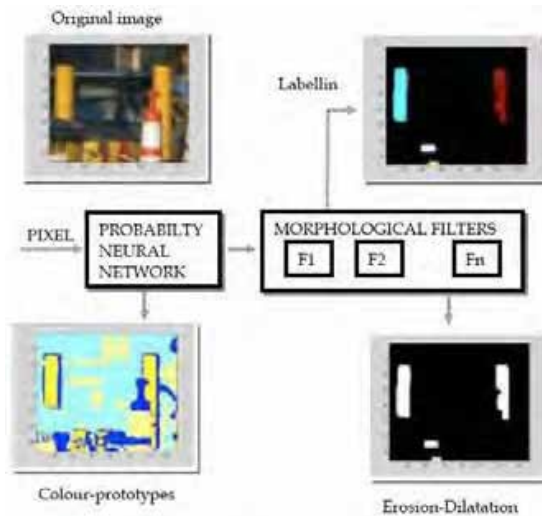


Fig. 8. Segmentation of regions.

Figure 8 exemplifies the segmentation process for the detection of yellow segments in an image. Here, the PNN has 8 colour-prototypes (Red, Blue, Green, Yellow, Magenta, Cyan, Dark and Light). At the end of the neural segmentation process, each original pixel has been transformed in a colour-prototype representing number. The whole image now looks like a pseudo-colour image. Then, the morphological process can be applied to any of these prototypes.

5. Implementation

5.1 Collision avoidance and target pursuit

This algorithm was first described in (Zapata et al., 1994) for many applications and tested for robots moving in 2 dimensions, flying robots or autonomous submarines and also mobile manipulators (Cacitti, 2000). This algorithm that was initially designed for obstacle avoidance, has two main advantages. First, the environment does not need to be *a priori* known, and second, the controller can take into account other constraints such as target pursuit (see paragraph 2.2), altitude maintaining, course control and so on. Figure 9 is a omnidirectional robot (equipped with 3 “sweedish” wheels) on which the DVZ algorithm was implemented. This robot has 3 ultrasonic sensors allowing the obstacle detection in 3 directions of space. The robot moves in the (x,y) plane and constantly rotates in order to update the proximity information in all directions of space.

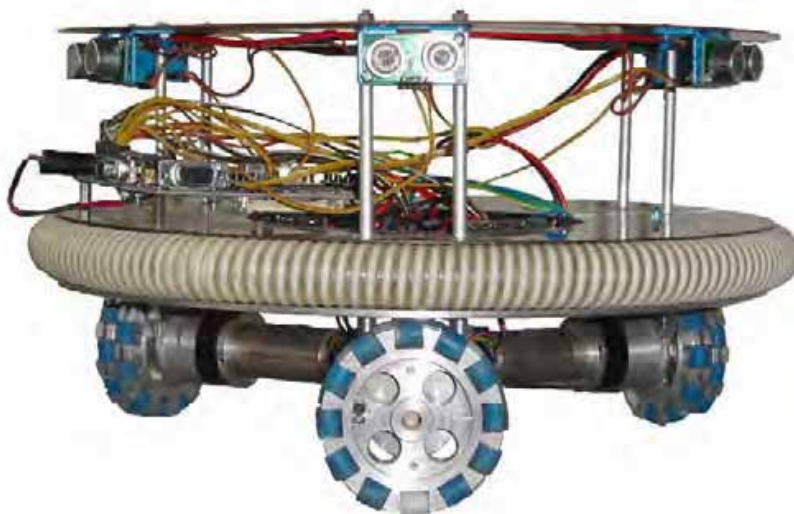


Fig. 9. The RATMOBILE robot.

5.2 Visual information

Images provide high-level sensory information that must be managed to make mobile robots move faster in unpredictable and dynamic worlds. Concurrent design of hardware/software systems (co-design) uses prototyping to evaluate the performance of the final system and to facilitate hardware-software partitioning. Prototype simulation can be achieved in quasi real-time (the clock frequency is generally one order of magnitude smaller than that of the final system on silicon). Signal processing applications (like the image processing algorithm described above) require very long validation digital patterns that lead to a prohibitive time for model simulation. Prototyping a signal processing system on a programmable and configurable platform allows validation of the processing function in real time. Within this framework, the system is first designed at a high abstraction level as a dataflow, with potential links with MATLAB tools.

Many researches have addressed co-design methods. In (Lodha et al.,1998) the authors studied a motion-based collision detection algorithm divided into 2 main subroutines that are allocated to firmware and hardware resources (DSP and FPGA). The algorithm has been tested and compared on a Sun ULTRA1 and Pentium pro133, and targeted to a DSP56002/XILINKS™ FPGAs for real-time implementation.

In (Haldar et al., 2000) the authors designed a real time vision embedded system for motion segmentation purposes, collision detection and object tracking. The hardware is based on a Pentium™ III PC running a Linux light kernel at video rates up to 25 frames per second with medium size images (128 × 96 and 384 × 288).

5.3 Hardware and software description

We co-designed vision systems, implemented and tested them on a EPXA10 development board from the Altera™ company. The Excalibur™ development board is built around an EPXA10 device which integrates an ARM922T-based hard processor with 16 Mbytes of 16-bit-wide flash memory, SDRAM controller, UARTs, DMA, Ethernet and PIOs. This device can also host one or several NIOS™ embedded RISC soft processors with 32-bit data paths with the same kinds of interfaces. The device can also be interfaced (UART) with an onboard PC.

This board comes with a development software kit (Quartus II™ and SOPC™ builder) which allows simulation and of software download. It is also possible to use dedicated IPs (Intellectual Property codes) and to develop dedicated home-made IPs. The prototype board is connected through IOs to the different sensors and actuators. The circuit is divided into 2 zones: the processor zone which is hardware-implemented (in the case of the ARM-based board) or software-implemented (in the case of the NIOS-based board). The PLD zone which contains the different home-made IPs dedicated to the control of sensors, actuators and, of course, the vision system. The design involves defining what part of the whole process will be implemented with hardware resources and what part will be implemented with software resources.

There are 3 “programming” levels:

- A low level design (written in VHDL language), part of the “hardware resources”, devoted to low level functions such as encoder reading, actuator control, image processing and image analysis.
- A medium level design (written in C language), part of the “software resources”, which incorporates functions that are not easy to integrate at a hardware level (e.g. fix point operations).

- A high level design, implemented in C language on the onboard PC to manage communication, supervision and some mathematical issues (e.g. floating point operations).

The development software allows writing and testing of the VHDL code and of the C code is also implemented on the imbedded processors (NIOS and/or ARM).

The vision system involves several home-made IPs (in our case, 3D reconstruction and colour image segmentation) linked to the vision hardware here, two 492×656 colour cameras with LVDS connections to the board.

5.4 Real-time implementation

This paragraph describes several aspects of the real implementation of the vision-based algorithms both on an Excalibur NIOS embedded prototype board and on a Excalibur ARM-based prototype board. The prototyping was only tested here with test images obtained with digital cameras and then downloaded to the embedded processor. We will focus on 4 aspects of this integration:

- The PLD structure for 3D reconstruction
- The VHDL implementation of a vertical gradient
- The VHDL implementation of a set of colour-prototypes
- The morphological dilatation operator

PLD structure for stereovision

The 3D reconstruction algorithm was developed for grey 256-level images. It is composed of 7 main blocks:

- a RAM controller that stores the right and left images
- a zone operator that determines the zone to which the incoming pixel belongs (44 zones for global vision and 16 for the fovea. See \ref{41})
- a gradient operator that computes the threshold of the 256-level image
- an operator that computes the best characteristic point for each of the 60 zones when there is one
- a matching operator that search the left image in order to minimize the SAD (sum of absolute differences) between an 11×11 patch around each characteristic right point and an 11×11 patch moving in the left image
- a sequencing machine that generates all the logical control signals
- a processor (NIOS or ARM) that reads images from a PC, transmits them to the PLD, gets the results and transmits them back to the PC

VHDL implementation of a vertical gradient

VHDL implementation of a gradient operator is a very good example of the difference between a high-level description language (C or MATLAB) and a logical language (VHDL). In this case, the main difficulty is to provide the information needed to compute the convolution of a given image and a gradient mask (in general a 3×3 mask). The $m \times n$ image arrives sequentially and the incoming pixel allows computation of the gradient of the pixel up-and-left from it. The gradient logical function consists of storing 2 lines and 3 pixels, including the incoming pixel, and computing the gradient. These $2n + 3$ pixels are stored in a FIFO-type memory. At each top of the pixel clock, the whole memory is translated and a new gradient is computed.

VHDL implementation of a set of prototypes

Each prototype is a VHDL object involving a maximum of N neurons (subroutines of the VHDL code) and can be configured in C by the NIOS during the programming phase. In this phase, the user chooses the number of prototypes to be programmed, the number of neurons per prototype, and the weight of each neuron. At each top of the programming clock (Progclk), the processor provides the PLD with a validation signal and the weight of the i^{th} neuron to be programmed.

In the application phase, at each incoming pixel arriving at each top of the pixel clock (Pixclk), the theme gives the probability of the pixel belonging to it.

All of these probabilities feed a compete layer that chooses the best prototype. The Gaussian activation function was tabulated in a Look-Up-Table (LUT) whose inputs are a sampling of distances in the RGB space and whose outputs are the corresponding Gaussian probabilities normalized between 0 and 2047 (typically, a power of 2).

Morphological erosion and dilatation

The second step of image segmentation involves filtering using morphological operators. We implemented a sequence of two operators: an erosion followed by a dilatation. The main interest of this sequence is to clean the images, i.e. small objects or protuberances are erased while the size of bigger objects is maintained. In this experiment, we chose a classical 3×3 full matrix as the structural element operating on the image.

Figure 10 shows an original image (left), the pseudo-color image representing the different zones corresponding to the 8 themes previously defined in paragraph \ref{42} (center) and the binary image representing the 6th prototype, in this case the yellow objects (right).

A morphological filter is implemented according to the same principles that underlie the gradient computation. A 3×3 structuring element requires 2 lines and 3 pixels to be stored, including the incoming pixel.

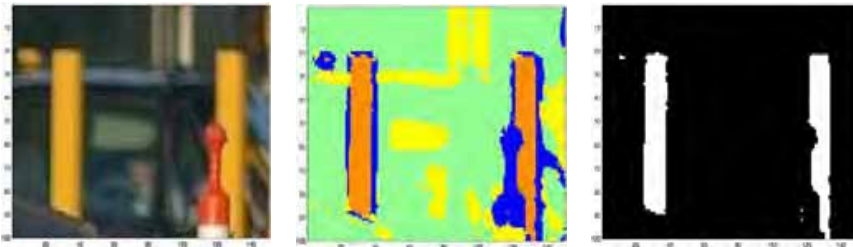


Fig. 10. Colour segmentation and morpho-filtering.

6. Conclusion

This chapter addressed the problem of controlling the reactive behaviours of a mobile robot evolving in unstructured and dynamic environments. We have carried out successful experiments for determining the distance field of a mobile robot using two

cameras. This vision system is coupled with a collision avoidance algorithm based on the DVZ method. In order to implement the previous algorithms, we developed a board including a FPGA programmable circuit from the Altera society. The image storing and the matching of the distance map is made in real time mode. The onboard PC computes the DVZ with the distance information issued from the processing board. This board can be interconnected with all the PC 104 interface board and with all the PC peripherals (screen, keyboard...). The needs for efficient vision-based systems seem obvious. Even if image processing often requires space and time, only high level sensory information provided by vision systems, can provide mobile robots with real autonomous capabilities. Looking for efficiency in vision-based systems can be done either by investigating fast natural vision-systems (natural systems) or by promoting co-designed hardware/software solutions. A mixed approach seems, of course, a good solution.

The bio-plausible reactive control algorithm for obstacle avoidance and target pursuit was tested for robots moving in 2 dimensions, flying robots or autonomous submarines. This algorithm that was initially designed for obstacle avoidance, has two main advantages. First, the environment does not need to be *a priori* known, and second, the controller can take into account other constraints such as target pursuit, altitude maintaining and course control.

7. References

- SAB (1999-2003). Simulation of adaptive behaviors, from animals to animats. 21
- Berthoz, A. and Weiss, G. (2000). *The Brain's Sense of Movement*. Perspectives in Cognitive Neuroscience, Paris.
- Beymer, D. and Konolige, K. (1999). *Real-time tracking of multiple people using continuous detection*. In IEEE Frame Rate Workshop.
- Brown, M., Burschka, D., and Hager, G. (2003). *Advances in computational stereo*. IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol.25, N°8.
- Cacitti, A. (2000). *Comportamento Reattivo di Manipulatori mobili non-Olonomi basato sul metodo del la Zona Virtuale Deformabile*. Tesi di laurea, Facolta di Ingegneria, Universita degli Studi di Bologna.
- Faugeras, O. and al (1993). *Real time correlation-based stereo:algorithm, implementations and applications*. Technical Report 2013, INRIA.
- Haldar, V., Vardhan, G., Saxena, A., Banerjee, S., and Balakrishnan, M. (2000). *Design of embedded systems for real-time vision*. In Indian Conference on Computer, Vision, Graphics and Image Processing (ICVGIP2000), Bangalore, India.
- Holenstein, E.; Badreddin, E. (1991) *Collision Avoidance in a Behaviour-Based Mobile Robot Design*, Proc. IEEE Int. Conf. on Rob. and Aut., Sacramento, California, USA.
- Kathib, O. (1985); *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, Proc. IEEE Int. Conf. on Rob. and Aut., pp.500-505
- Kimura, S., Shinbo, T., Yamaguchi, H., Kawamura, E., and Naka, K. (1999). *A convolver-based real-time stereo machine (sazan)*. Computer Vision and Pattern Recognition, Vol.1, pages 457-463.
- Lodha, S., Gupta, S., Balakrishnan, M., and Banerjee, S. (1998). *Detection and avoidance: A case study for design space exploration in h/w-sw codesign*. In 11th conference on VLSI Design, Chennai, India.

- VanRullen, R. and T. S. (2002). *Surfing a spike wave down the ventral stream*. Vision Research, Vol.42, pages 2593–2615.
- Wurtz, R. and Lourens, T. (2000). Corner detection in color images through a multiscale combination of end-stopped cortical cells. Image and Vision Computing, Vol.18 N°6-7, pages 531-541.
- Zapata, R., Lépinay, P., and Thompson, P. (1994). *Reactive behaviors of fast mobile robots*. Journal of Robotic Systems, Vol.1 N°8.

Transputer Neuro-Fuzzy Controlled Behaviour-Based Mobile Robotics System

E. Al-Gallaf

*Department of Electrical and Electronics Engineering, College of Engineering,
University of Bahrain, P. O. Box 13184
Kingdom of Bahrain*

1. Introduction

1.1 A Transputer Mobile Robotics System

Mobile robots have received a considerable attention from early research community, from (A. Benmounah, 1991), (Maamri, 1991), (Meystel, 1991) up to this instant (Hegazy, et. al, 2004), and (Pennacchio, et. al., 2005). A *fuzzy or neural* control Transputer based control mobile robots has received, rather, little attention. A number of, are (Welgarz,1994), (Probert, et. al, 1989), (Iida and Yuta, 1991), and (Brady, et. al., 1993). Recently, neurofuzzy logic controllers are found well suited for controlling mobile robots, (Rusu et. al. 2003). This is because, they are talented of building inferences even under certain uncertainty and unclear conditions, (Kim, and Trivedi, 1998). Having a hierarchical architecture that divides the neurofuzzy into *several smaller* subsystems will rather condense the negative effect that a large rule-base may have on real-time performance. Problems of *insufficient knowledge* for designing a rule base can be solved by using a neurofuzzy controller. Learning allows autonomous robots to acquire knowledge by interacting with the environment and subsequently adapting their behaviour. Behaviour learning methods are used to solve complex control problems that autonomous robots encounter in an unfamiliar real-world environment.

Neural networks, fuzzy logic, and reinforcement- and evolutionary-learning techniques can be utilized to achieve basic behavioural functions necessary to mobile robotics system. There are large number of recent research in mobile robot fuzzy behavior navigation. For instant, (Willgoss and Iqbal, 1999), have reported the use of neurofuzzy learning for teaching mobile robot behaviors, selecting exemplar cases from a potential continuum of behaviors. Proximate active sensing was successfully achieved with infrared in contrast to the usual ultrasonic and viewed the front area of robot movement. (Pennacchio, et. al., 2005), have presented, the FU.LO.RO., a new controller for mobile robot, that moves itself autonomously in an unknown environments. The system has been developed following two different approaches: first one enables a fuzzy controller to determine the robot's behavior using fuzzy basic rules; the second uses a neurofuzzy controller.

(Tsoukalas, et. al., 1997), have presented a neurofuzzy methodology is for motion planning in semi-autonomous mobile robots. The robotic automata considered are devices whose main feature is incremental learning from a human instructor. Fuzzy descriptions are used for the robot to acquire a repertoire of behaviors from an instructor which it may subsequently refine and recall using neural adaptive techniques. The robot is endowed with sensors providing local environmental input and a neurofuzzy internal state processing

predictable aspects of its environment. Although it has no prior knowledge of the presence or the position of any obstructing objects, its motion planner allows it to make decisions in an unknown terrain. (Hegazy, et. al, 2004) have shown controlling mobile robot navigation system that operates in an unknown and uncertain environment is a difficult operation. Much of this difficulty is due to environmental inconsistencies and sensor inadequacies. Training data was accumulated from robots sensors to generate a set of fuzzy rules that govern the robot navigation system on-line. A Transputer-based (T-805) locomotion module provides all of motor feedback and control of a robot (Iida and Yuta, 1991). Locomotion module was designed to follow a given trajectory, using feedback information from the robot's wheel encoders. The locomotion module operates as a digital PID controller to govern motion of the robot.

1.2 Research Outline

In this respect, this chapter discusses a neurofuzzy controller strategy for sensor-based mobile robotics system navigation for an indoor environment applications. A Transputer computation power is used to carry out complicated needed computation (reading sensors data, deciding actions, outputting wheels data, ... system monitoring).

Robot control mythology was run on a parallel computing environment known as Transputers. The Transputer embedded real-time controller was used on board the robot to meet various intelligence requirements for the free navigation and obstacle avoidance. The control system consists of a hierarchy of robot behaviours. The mobile behavior control system was based on the use of a number of Transputers processors. Behavior methodology was based on the utilization of the structure of a five layers neuro-fuzzy system that learns, trains, and adapts itself to the environment within which it operates for the purpose of robot body maneuvering.

The autonomous mobile robot uses ultra-sonic sensors for detecting targets and avoiding collisions. The control system is organized in a top-bottom hierarchy of various tasks, commands, and behaviours. When multiple low-level behaviours are required, command fusion is used to combine the output of several neuro-fuzzy sub-systems. A switching coordination technique selects a suitable behaviour from the set of possible higher level behaviours. A parallel (Transputers based) fuzzy control is implemented for the robot guidance and obstacle avoidance. The mobile robot used in this work has been designed and constructed by the author at the University of Bahrain. The key issue of this research frame work is the utilization of a neurofuzzy system that runs over a parallel Trasputers. This has shown the ability to reduce the computational time needed for the movement.

2. The Mobil Robot

2.1 (Experimental Testbed) Physical Parameters

The mobile robot can be seen in Fig. 1. It is a small mobile autonomous robotic-Testbed (AL-Gallaf, 2006), is utilized to achieve defined robot behaviors. It has a Transputer system allowing high level control consisting of C++ , Matlab, and Occam routines to provide a multitude of functions. Its drive wheels are driven with a 10 : 1 gear ratio to reach motor torque of 10 N/m. The maximum speed it can reach is 0.7 m/s. The mobile robot weighs 2Kg in a rectangular shape of width 30 cm and length of 40 cm. It has two moving wheels of diameter 10cm located at the center of the robot used for motion and

steering. Shaft encoders are attached to each motor for wheel sensing. Free-wheeling castors are located one in the front and one at the back of the robot to balance the robot. Corners of the robot circumference are flat and with 45° angle from the adjacent flat wall. These flat corners are used to mount the corner ultrasonic sensors in order to enable the robot to see in that direction. Photograph of the mobile robot is shown in Fig. 1. The mobile robot must be capable to follow an $(x-y)$ path in any direction (θ) over the plane. It should have one degree of translation and one degree of rotation. All steering axes are perpendicular to the surface. The mobile robot can follow a path in any direction, first it must rotate around its axis to face that direction. It has two diametrically opposed drive wheels. Because of the simplicity of its mechanical design it has simple kinematics.

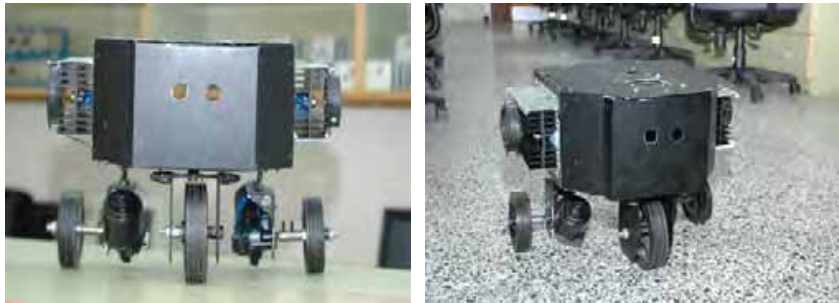


Fig. 1. (A Testbed) Mobile Robot Construction.

2.2 Wheel Model Type

Three types of wheels are used in the wheeled mobile robot design: Conventional, unidirectional, and ball wheels. The robot conventional wheels are used on a fixed axis therefore, there is no steering joint. The conventional wheels are modeled by a planner pair at the point of contact (McKerrow, 1991). With this representation, the multiple degrees of freedom of wheel motion can be modeled without ambiguities in the transformation matrices. A conventional wheel has only two degrees of freedom, because the third degree of freedom in the planar pair model is eliminated when the x component of the wheel velocity is set to zero to eliminate sideways slip. The y component of the wheel velocity is equal to the angular velocity times the radius of the wheel, ($y_x = \omega_x \times r$). The y component of wheel velocity allows travel along a surface in the direction of the wheel orientation. The conventional wheel is by far the most widely used wheel.

2.3 Robot Sensors

Robot sensors are mounted around the circumference of the robot as shown in Fig. 2. They are at 30 centimeters high from the ground floor. Sensors are slightly tilted upward to prevent ultrasonic echo reflection from the ground floor. The mobile robot is powered by four rechargeable 12V batteries, $2 \times 12A_h$ and $2 \times 6A_h$. These batteries are configured to supply +12V, -12V, +24V and -24V. A voltage converter is supplied to provide +5V.

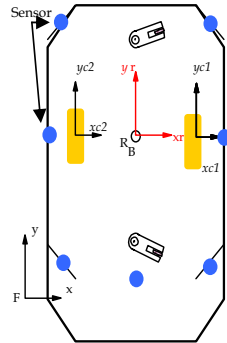


Fig. 2. The Mobile Robot Sensing.

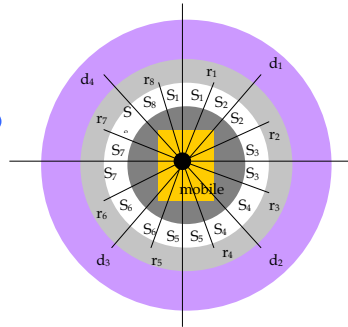


Fig. 3. Sensors grouped in four region.

2.4 The Sensor Fusion

The sensor fusion is the integration of all of the eight ultrasonic sensor to form the inputs to the neuro-fuzzy controller. Robot sensors are grouped in to four regions as shown in Fig. 3. Region-1 consist of sensors (s_1 , s_2 , and s_3). Region-2 consist of sensors (s_3 , s_4 , and s_5). Region-3 consist of sensors (s_5 , s_6 , and s_7). Region-4 consist of sensors (s_7 , s_8 , and s_1). In this arrangement some sensors are part of more than one region and this grouping is consistent with the neuro-fuzzy group classification where the borders between the sets (regions) is not crisp, (Fig. 3.). In the process of selecting a region out of the four, the following steps are implemented :

- Read all the sensors (s_1, s_2 , to s_8).
- Arrange sensor outputs value in ascending order.
- Identifying the three sensors with the lowest value.
- If two of the three are in one region, that region will be selected.
- If no, go to first step.

3. Mobile Robots Kinematics Modeling

To provide a framework within which to develop the robot kinematics models, (Muir and Neuman, 1986) defined a wheeled mobile robot as : " A robot capable of locomotion on a surface solely through the action of wheel assemblies mounted on the robot and in contact with the surface. A motion of a robot is determined from geometry of the constraint imposed by the wheels motion. Kinematic analysis is based on the assignment of coordinate axes within the robot and its environment, and the application of (4×4) matrices to transform between coordinate systems. Kinematic model is derived with respect to coordinate axes assigned to each robot joint as will be shown in the following sections.

3.1 Robot Coordinate Frames Assignment

Coordinate frame is assigned at each link of the mobile robot. The mobile robot links are the floor and the robot body, Fig. 4. These links are connected by two joints: the wheel contact point with the floor and the mid-point of the robot. The mid-point of the robot is not a physical joint, but the relationship between the body of the robot and the floor is

modeled as a planar pair located at the center of the robot. The wheel is also modeled as a planner pair located at the point of contact between the wheel and the floor. The z axis of all these frame is vertical, and is neglected in two-dimensional analysis. The instantaneously coincident coordinate systems (frames C_F and R_F) are stationary coordinate frames located at the same point as the moving coordinate frame at the instant of observation. Position transform between the two frames is zero. These frames are instantaneously fixed with respect to floor and not to the robot. At the instant these frames are considered, they are coincident with the frames attached to the robot. Frame R_F coincides with frame R_B and frame C_F coincides with frame C_L . The driving wheels are fixed to the body, the steering frames do not move with respect to either the robot body frame or wheel contact frame. Floor coordinate frame F is stationary and serves as a reference frame for robot motion. Robot frame R_B is located at the center of the robot, and serves to define the location of the robot with respect to the floor frame for the kinematics. Fig. 5. shows the coordinate frames assignment for the mobile robot seen from top view. Fig. 6. is the side view of the same system.

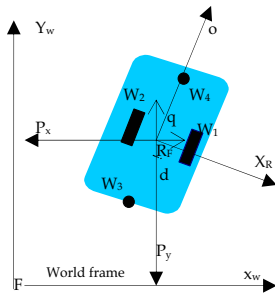


Fig. 5. Robot and World frames Coordinates.

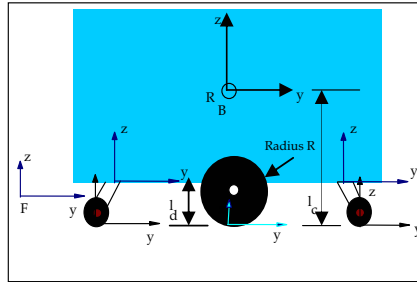


Fig. 6. The coordinate frames assignment for the mobile robot (side view -y axes point out of page).

4. Transformation Matrices

Modeling uses homogeneous transforms to describe the transformation. Homogeneous transformation matrices ($\in \mathbb{R}^{4 \times 4}$) express the relative positions and orientations of coordinate systems (Beom & Cho, 1995). The homogeneous transformation matrix ${}^A\mathbf{I}_B$ transforms the coordinates of the point ${}^B r$ in coordinate frame B to its corresponding coordinates ${}^A r$ in the coordinate frame A:

$${}^A r = {}^A\mathbf{I}_B {}^B r \tag{1}$$

Vectors ${}^A r$ and ${}^B r$ denote points in space consist of three Cartesian coordinates and a scale factor as the fourth element, the scale factor is always unity:

$$A_r = \begin{pmatrix} A_{r_x} \\ A_{r_y} \\ A_{r_z} \\ 1 \end{pmatrix} \tag{2}$$

The transformation matrices contain the $\in \mathbb{R}^{3 \times 3}$ rotational matrix ($n \ o \ a$), and $\in \mathbb{R}^{3 \times 1}$ translational vector p :

$${}^A T_B = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

The three vector components n , o and a of the rotational matrix in (3) express the orientation of the x , y and z , respectively, of B coordinate systems are relative to the A coordinate system. The three components p_x , p_y , and p_z axes of the translational vector p express the displacement of the origin of the B coordinate system relative to the origin of the A coordinate system along the (x , y , and z) axes of the A coordinate system, respectively. All the coordinate systems are assigned to the robot with z axes perpendicular to the travel surface. All rotations between coordinate system are about the z axis. Because of the robot three-dimensional shape, there are translations in all three directions. Thus, a general transformation matrix for the mobile robot between the robot body frame (R) and floor frame (F) is given by (Muir and Neuman, 1987) :

$${}^R T_F = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & p_x \\ \sin\theta & \cos\theta & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

where θ is the rotational angle between robot frame and floor frame (Fig. 5.) and equal to $\omega \times r$. For zero rotational and translation displacements, the coordinate transformation matrix in Equ (1) reduces to an identity matrix. The velocity transformation matrix is calculated by differentiation of Equ (4) matrix component wise, to give :

$${}^R V_F = \begin{bmatrix} -\omega \sin\theta & -\omega \cos\theta & 0 & v_x \\ \omega \cos\theta & -\omega \sin\theta & 0 & v_y \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

where: ω = The angular velocity of the wheel, v_x = robot component of the linear velocity, v_y = robot y component of the linear velocity. From above matrix transformation, the robot linear velocity can be derived as follows (Reister & Pin, 1994) :

$$\theta = \frac{r}{d} (\omega_R + \omega_L) \quad (6)$$

$$v_x = (\omega_R + \omega_L) \times \cos(\theta/2)$$

$$v_y = (\omega_R + \omega_L) \times \sin(\theta/2)$$

in which (r is the radius of each wheel) and (d is the distance between the two wheels). In Equ (6), ω_R and ω_L are wheel right and left angular velocities, respectively in radians per second.

5. The Trasputer Based Parallel Controller

The mobile robot is controlled by a neuro-fuzzy system which runs on a trasputer (a parallel processing computing). Within this section we shall introduce the Occam.

5.1 The Trasputer

The architecture of the Trasputer **T414**, Fig. 7., is rather simple and borrows architectural ideas from Texas Instrument’s TMS 9000 microcomputer and the old Hewlett-Packart calculators. It has 32 bit 10 MIPS processor, 4 Gigabyte linear address space, 32 bit wide 25 MByte/sec memory interface, configurable on-chip memory controller, 2Kbytes high speed on chip RAM, 4 Inter-trasputer links, each with full duplex DMA transfer capability up to 20 Mbits/sec., advanced 1.5 micro CMOS technology, and low power dissipation (less than 500 mw).

The T414 is a 32 bit Trasputer capable of executing up to 10 MIPS at a speed of 20 MHZ, Fig. 7. From the Occam model, Inmos developed a hardware chip to carry out their concurrency model. This hardware is in the form of a very large scale integration (VLSI) integrated chip (IC) called the Trasputer. The Trasputer (Inmos part number T800) was a 32-bit microprocessor (20 MHz clock) that provides 10 MIPS (million instructions per second) and 2.0 MFLOPS (million floating point operations per second) processing power with 4K bytes of fast static RAM (Random Access Memory) and concurrent communication capability all on a single chip. Communication among processes is done by means of channels, Fig. 8. A channel between processes executing on the same Trasputer is a soft channel, while a channel between processes executing on different processors is a hard channel.

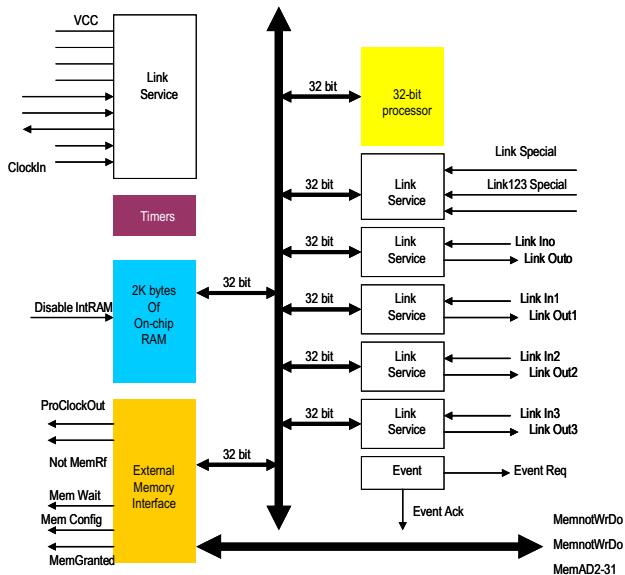


Fig. 7. A Trasputer System.

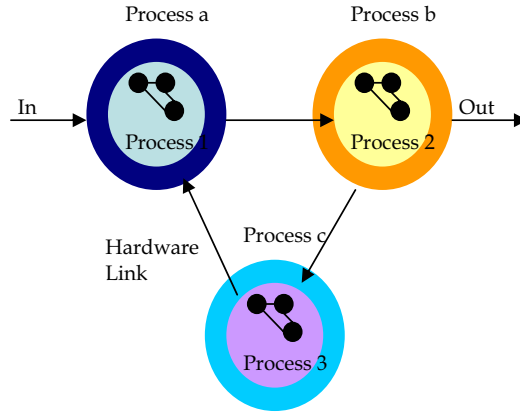


Fig. 8. Occam Processes on Separate Machines.

5.2 The Occam

Occam language enables programs to be written as a collection of self-contained programs (tasks) which may be executed simultaneously (parallel or concurrent) or simply one after another with a built-in inter-process communication mechanism, depending whether these processes are executed on a single machine or on separate ones. Occam is a high-level language, it can be viewed as the assembly language for the Transputer. Unlike most microprocessors, e. g., the M68000, the definition of the operations of the Transputer is in terms of the Occam model and not machine language. Besides being a high performance microprocessor (*half the speed of a VAX 8600*), the Transputer has on its chip four (4) serial bi-directional links (*each 20 Megabits per second*) to provide concurrent message passing to other Transputers. The “channels” in the Occam language are mapped to these hardware links which connect by way of twisted pairs of wires to other Transputers. Transputer hardware supports concurrency by scheduling (*time-slicing*), in round-robin fashion, an arbitrary number of Occam concurrent processes. The language and the hardware are so designed that an Occam program consisting of a collection of concurrent processes may execute on one Transputer (via time slicing between the different concurrent processes) or be spread over many Transputers with little or no change in the Occam code. Consequently, the designer can build up his Occam program on one Transputer, and if higher performance is required, can spread the Occam processes over a network of interconnected Transputers. The original Transputer (T414), having no floating point unit and only 2 Kbytes of RAM. Inmos has developed a new, faster version of the Transputer called the (T9000). The T9000 is a 150 MIPS microprocessor with a 20 MFLOPS floating point unit.

6. Neuro-Fuzzy Control System

Rule-based structure of fuzzy models allows for integrating heuristic knowledge with information obtained from process measurements. Fuzzy sets are used to define the process operating conditions such that fuzzy dynamic model of a nonlinear process can be described in the following way :

$$\begin{aligned}
 R_i &: \text{ If operating condition } i \\
 \text{Then } \hat{y}_i(k) &= \sum_j^o a_{ij} s(k-j) + \sum_j^i b_{ij} u(k-j) \quad (i = 1, 2, \dots, r)
 \end{aligned} \tag{7}$$

Final model output is obtained by the center of gravity defuzzification as follows:

$$\hat{y}(k) = \frac{\sum_{i=1}^r \mu_i \hat{y}_i(k)}{\sum_{i=1}^r \mu_i} \tag{8}$$

In Equ (7) and Equ (8), y is the system output, u is the system input, \hat{y}_i is the prediction of process output in the i^{th} operating region, (r) is the number of fuzzy operating regions, (i) and (o) are the time lags in the input and the output, respectively, μ_i is the membership function for the i^{th} model, and finally, a_{ij} and b_{ij} are the ARMAX model parameters. The membership function for an operating region is constructed in a number of ways. One approach to calculate its membership function as follows :

$$\mu_i = \min(\mu_n(x), \mu_m(y)) \tag{9}$$

$$\mu_i = \mu_n(x), \mu_m(y) \tag{10}$$

In Equ (9), μ_i is a membership function in i^{th} operating region, $\mu_n(x)$ is membership function of x being (*high*), and $\mu_m(y)$ is the membership function of y being (*medium*). In Equ (10), we calculate gradient required for gradient based network training.

6.1 Neural Network Representation of Fuzzy Systems

Fuzzy Systems can be represented by a special type of network topology which is termed here a neuro-fuzzy. Fuzzy reasoning is capable of handling uncertain and imprecise information while a neural network is capable of learning from examples. Neuro-fuzzy intend to combine the advantages of both fuzzy reasoning and neural networks.

6.2 Neuro-fuzzy Architecture

For simplicity, it is assumed, the fuzzy inference system under consideration has large number of inputs, e.g. (d_1, d_2, d_n , sensors data, s_1, s_2, \dots, s_n) and two outputs y_1, y_2 . If a rule base contains two fuzzy if-then rules of Takagi and Sugeno's type. A rule as:

$$\begin{aligned}
 \text{rule 1:} & \quad \text{If } d_1 \text{ is } A_1 \text{ and } d_2 \text{ is } B_1, \text{ then } y_1 = p_1 s(1) + q_1 s(2) + r_1 \\
 \text{rule 2:} & \quad \text{If } d_3 \text{ is } A_2 \text{ and } d_n \text{ is } B_2, \text{ then } y_2 = p_2 s(1) + q_2 s(2) + r_2
 \end{aligned}$$

where p, q , and r are constants and called parameter set. That is, the if parts of the rules are same as in the ordinary fuzzy if-then rules, then parts are linear combinations of the input variables. The employed neuro-fuzzy architecture is shown in Fig. 8., where node functions in the layers are described below :

Within first layer: Each i^{th} node in this layer is a square node with a node function :

$$O_i^1 = \mu_{A_i} s(1) \quad (11)$$

where s_1 is the input to i^{th} node, and A_i is the linguistic label (small , large, .. etc.) associated with this node function. In other words, O_i^1 is the membership function of A_i and it specifies the degree to which the given s satisfies the quantifier A_i . Usually we choose $\mu_{A_i}(s(1))$ to be bell-shaped with maximum equal to 1 and minimum equal to 0, such as :

$$\mu_{A_i} s(1) = \frac{1}{1 + \left[\left(\frac{s(1) - c_i}{a_i} \right)^2 \right]^b} \quad (12)$$

or

$$\mu_{A_i} (s(1)) = \exp \left\{ - \left(\frac{s(1) - c_i}{a_i} \right)^2 \right\} \quad (13)$$

where $\{a_i, b_i, c_i\}$ is the parameter set. As values of these parameters change, membership shaped functions vary accordingly, thus exhibiting various forms of membership functions on the linguistic label A_i . In second layer, every node in this layer is a circle node which multiplies incoming signals and sends their product out. For instance,

$$y_i = \mu_{A_i} s(1) \times \mu_{B_i} s(2) \quad i = 1,2 \quad (14)$$

Each node output represents the firing strength of a rule. In third layer, every node in this layer is a circle node. The i^{th} node calculates the ratio of the i^{th} rule's firing strength to the sum of all rules' firing strengths :

$$\bar{y}_i = \frac{y_i}{y_1 + y_2 + \dots + y_i} \quad i = 1,2 \quad (15)$$

For fourth layer, every node i in this layer is a square node with a node function

$$O_i^4 = \bar{y}_i f_i = \bar{y}_i (p_i s(1) + q_i s(2) + r_i) \quad (16)$$

where \bar{y}_i is the output of third layer, and $\{p_i, q_i, r_i\}$ is the parameter set. Parameters in this layer will be referred to as consequent parameters. Finally, the fifth layer, the node in this layer is a circle node. It computes the overall output as the summation of all incoming signals, i.e. :

$$O_1^5 = \text{overall output} = \sum_i \bar{y}_i f_i = \frac{\sum_i y_i f_i}{\sum_i y_i} \quad (17)$$

Thus we have constructed an adaptive network which is functionally equivalent to a fuzzy inference system, achieved by neural system alone. This is shown in Fig. 9.

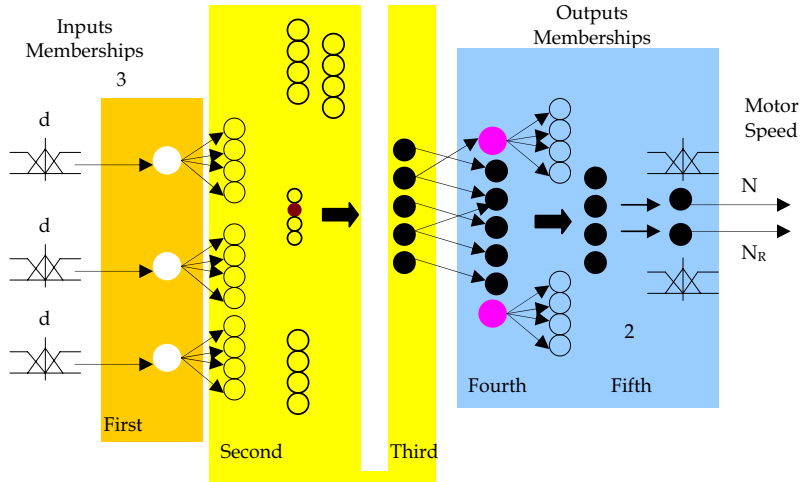


Fig. 9. Employed Neuro-fuzzy system.

6.3 Neuro-fuzzy Training

From the shown Neuro-fuzzy architecture shown in Fig. 9., it is observed that, given values of premise parameters, the entire output is expressed as a linear combinations of the consequent parameters. More precisely, output \hat{y} in Fig. 9. is rewritten as :

$$\begin{aligned} \hat{y}_m &= \frac{y_1}{y_1+y_2} \hat{y}_1 + \frac{y_2}{y_1+y_2} \hat{y}_2 + \dots + \frac{y_{m-1}}{y_m+y_{m-1}} \hat{y}_{m-1} \\ \hat{y}_m &= \bar{y}_1 \hat{y}_1 + \bar{y}_2 \hat{y}_2 + \dots + \bar{y}_{m-1} \hat{y}_{m-1} \\ \hat{y}_m &= (\bar{y}_1 s(1)) p_1 + (\bar{y}_1 s(2)) q_1 + (\bar{y}_1) r_1 \\ &\quad + (\bar{y}_2 s(1)) p_2 + (\bar{y}_2 s(2)) q_2 + (\bar{y}_2) r_2 + \dots \end{aligned} \tag{18}$$

which is linear in the consequent parameters (p_1, q_1, r_1, p_2, q_2 and r_2). The consequent parameters thus identified are optimal (*in the consequent parameter space*) under the condition that the premise parameters are fixed. A model's weights are conventionally identified by performing maximum likelihood estimation. Given a training data set $Z^N = \{y(k), s(k)\}_{k=1}^N$ the task is to find a weight vector which minimizes the following cost function of Equ (19) :

$$J_N(w) = \frac{1}{N} \sum_{k=1}^N [y(k) - \hat{y}(s(k), w)]^2 \tag{19}$$

As the model, $\hat{y}(s(k), w)$, is nonlinear with respect to the weights, linear optimization techniques cannot be applied. Instead the popular Truncated Newton nonlinear optimization algorithm is employed. All the five operations are computed via the Trasputer system, where is capable of computing fuzzy inputs in a concurrent way.

7. The Mobile Robot Controller

7.1 Over ALL Control

Eight ultrasonic sensor range measurement data (s_1, s_2, \dots, s_n) are input to the controller (*Neuro-fuzzy*) as (d_1, d_2, \dots, d_n). Outputs are the left and the right wheel speeds, (N_L and N_R). This controller is integrated with the sensor fusion and the low level PID controller to form the complete controller for the mobile robot as shown in Fig. 10.

The overall controller starts with a user interface where the user can enter the desired control parameters. In this interface the statues of the mobile robot will be displayed before starting the controller.

The robot checking routine will perform self testing on the various parts of the hardware of the robot and report any difficulty before starting. This section forms the very high level of the controller servo.

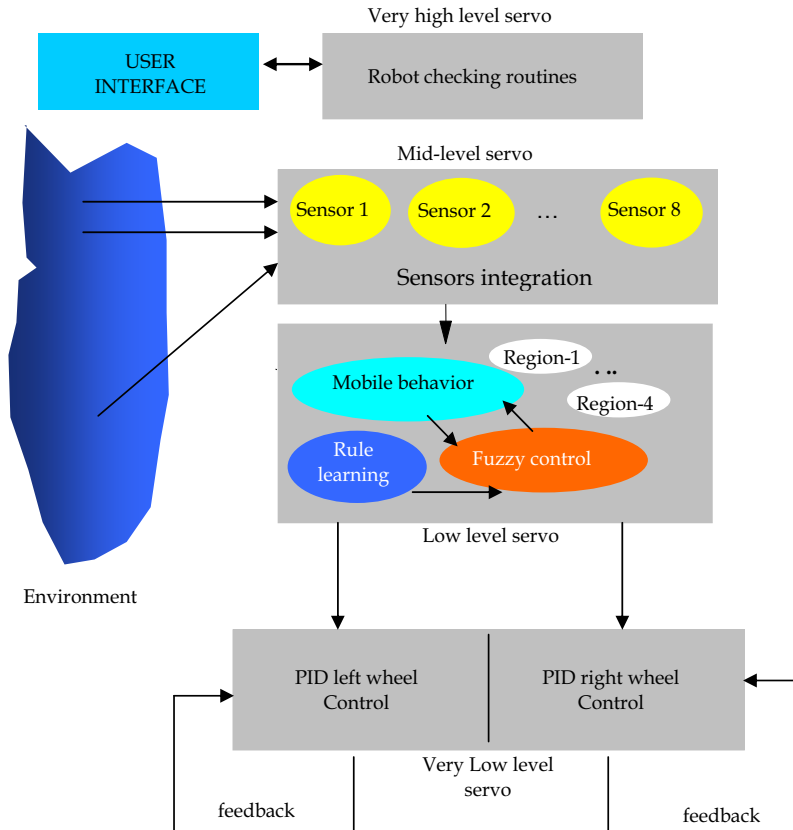


Fig. 10. Overall robot controller.

7.2 Fuzzy Control Information Processing

The processing of sensors information in the neuro-fuzzy process will first include the fuzzification of the sensors input variables (s_1, s_2, \dots, s_n), then the defuzzification of the output variables as follows: First, the fuzzification of the sensor (s_1 and s_2) to input variables (d_{min1} and d_{min2}). Once the region is selected the information supplied by the sensor in that region will indicate the nearest obstacle to the robot and the next nearest obstacle (d_{min1} and d_{min2}) facing that region. The mobile control can be realized by the control of the robot wheels rotation to keep the robot within a pre-set distance from the obstacle facing that region of the robot. The neuro-fuzzy operator converts the crisp sensor input data, (say d), into the linguistic values (\hat{d}) determined as labels of fuzzy sets given by Equ (20) :

$$\text{fuzzifier}(d_1, d_2) \quad (d_1^{\wedge}, d_2^{\wedge}) \tag{20}$$

where fuzzifier denotes a Fuzzification operation. From now on, the sign (\wedge) representing the fuzzy set will be omitted for simplicity. The sensors input linguistic variables d_1 and d_2 are expressed by linguistic terms as illustrated by the membership function in Equ (21) and Equ (22), respectively. Meaning of the linguistic term is given in Table. 1.

$$\{es\ vvs\ vs\ s\ m\ b\ vb\ vvb\ eeb\} \in m(d_1) \tag{21}$$

$$\{es\ vvs\ vs\ s\ m\ b\ vb\ vvb\ eeb\} \in m(d_2) \tag{22}$$

symbols	Linguistic variable
<i>es</i>	<i>extremely small</i>
<i>vvs</i>	<i>very very small</i>
<i>vs</i>	<i>very small</i>
<i>s</i>	<i>small</i>
<i>m</i>	<i>medium</i>
<i>b</i>	<i>big</i>
<i>vb</i>	<i>very big</i>
<i>vvb</i>	<i>very very big</i>
<i>eeb</i>	<i>extremely big</i>

Table 1. Fuzzy linguistic terms of Inputs (d_1, d_2, \dots, d_8).

The controller outputs are right and the left wheels speed (N_L and N_R) in rad/sec. They are expressed by linguistic values with membership functions having bell shape functions. Halfway of this membership function is determined by zero initial value. The linguistic provisions for output variables are given in Equ (23) and in Table. 2. :

$$\{bn, n, mn, z, mp, p, bp\} \subseteq m(N_L) \tag{23}$$

symbols	Linguistic variable
<i>bn</i>	<i>big negative</i>
<i>n</i>	<i>negative</i>
<i>mn</i>	<i>medium negative</i>
<i>z</i>	<i>zero</i>
<i>mp</i>	<i>medium big</i>
<i>p</i>	<i>positive big</i>
<i>bp</i>	<i>big positive</i>

Table 2. Output variables N_L and N_R .

In Equ (23), the eleven states membership function are expressed in terms of degree of membership function (MF). Fuzzy subsets embody elements with degree of membership. On the other hand, fuzzy FM $\mu_x(\square)$ of the fuzzy set $\{\square\}$, assigns a real number (between 0 to 1) to every element in the universe of discourse.

7.3 Construction of the Rule base Fuzzy System

The rule base for realizing each behavior can be constructed based on operator knowledge. For the fuzzy controller, the partial mapping is to be translated in to linguistic rules. A fuzzy rule has an IF-THEN format as follows :

IF (d_1 is *mb* AND d_2 is *vb*) then (N_L is *vvb* AND N_R is *vb*)

where (d_1, d_2), (N_L , and N_R) are the fuzzy variables and (*mb*, *vb*, *vvb* and *vb*) are the fuzzy subsets in the universe of discourses X , Y and Z . A fuzzy rule base consists of several rules of the form given above. The experience of the operator play a big role in the shape and form of the rules and in the number of the rules, in the rule base fuzzy controller. In this fuzzy controller, membership functions ($\mu_1, \mu_2, \mu_3, \mu_4$) with association with the (d_1 and d_2) are computed as follows (Baxter, 1995) :

$$x = \text{int}((d_1+1)/10) \quad (24)$$

$$y = \text{int}((d_2+1)/10) \quad (25)$$

where *int* is the integer part of an expression :

$$xx = (10x + 5 - d_1)$$

$$yy = (10y + 5 - d_2)$$

and the complement of these fuzzy sets is given by :

$$C_x = 1 - xx \text{ and } C_y = 1 - yy \quad \mu_i = \mu_i(d_1) \oplus \mu_i(d_2) \quad (26)$$

$$\mu_1 = \min (xx, yy), \mu_2 = \min (xx, C_x), \mu_3 = \min (C_x, C_y), \mu_4 = \min (C_x, C_y) \quad (27)$$

Equ (26) and Equ (27) is the fuzzy OR operator for the fuzzy set that select a minimum of the two variables. Membership functions of fuzzy variables (as in Equ (23)) are used in the defuzzification process (Zinger and Elbuluk, 1994).

8. Performance Validation

8.1. Behavior Learning From Samples of Robot Movements

Following the five layers neuro-fuzzy system was assembled, inputs to this system are sensory information coming from the robot. They are $(d_1, d_2, \text{ and } d_3)$. This rather represents associated distances and location the robot is with respect to obstacles. This neuro-fuzzy system was run by a net of Transputers through an Occam programming environment. Now the mobile robot is ready to be trained. First, the mobile robot was moved arbitrarily within the space. Sensors data were collected. This rather represents large number of data (since eight sensors were monitored simultaneously). Fig. 11-a shows some robot movement. Typical wheels speeds were also set, hence defining some preset values of the robot movement. Fig. 11-b shows such low level real-time step response of one wheel (recorded via one Transputer node). The wheel was controlled by a digital PID controller.

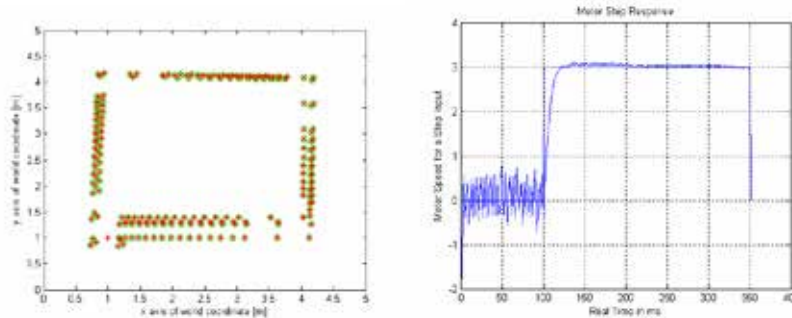


Fig. 11. Overall robot controller. (a: Randomly moved robot), (b: Typical low level wheel control)

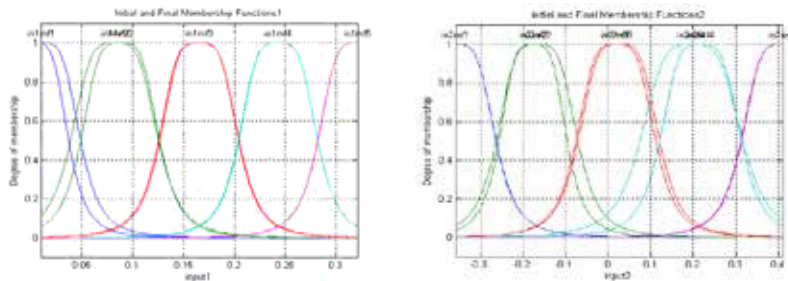


Fig. 12. Initial and final tuned memberships. (a: Sensor data recording, as input to NF, e.g. sensor data d_1), (b: Wheel speeds N_L)

Accordingly, Fig. 12 shows the initial and finally tuned memberships of fuzzy variables $(d_1, d_2, \text{ and } d_3)$. The initial memberships are set by the user himself, whereas, the final ones are as results of the neuro-fuzzy tuning action. By now, the robot is trained. For the purpose of

making sure the mobile robot has learned the surrounding environment, Fig. 13-a. shows a computed errors between the trained neuro-fuzzy system and the robot actual wheel speeds. The computed error is small enough to tell that the mobile robot has learned the fuzzy if then rules via the five layers.

On the other hand, Fig. 13-b. shows the relation between two different inputs of learned neuro-fuzzy system. This 3-D plot represents in fact what will be the associated mobile wheel speeds (N_L and N_R outputs) for any two combinations of inputs (d_1 , d_2 , and d_3). The plot can further shows the learned expert if-then rules. This will be useful batch of information once the trained mobile intelligence is compared with only fuzzy (if-then) rules.

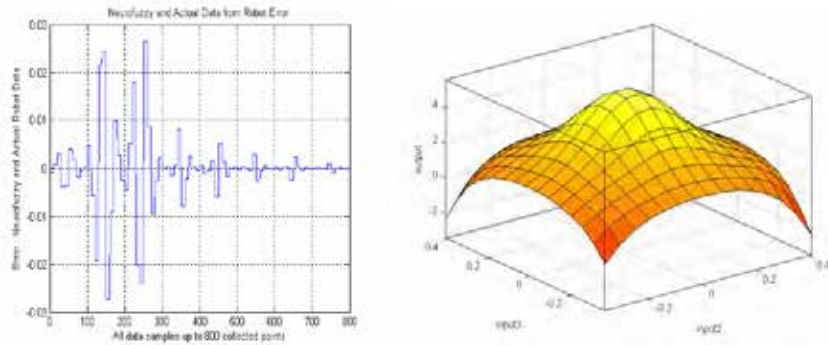


Fig. 13. Learned five layers Neuro-fuzzy system.
(a: Trained Neuro-fuzzy and actual robot error), (b: Learned fuzzy if-then 3-D plot)

8.2 Robot Posture Seeking

An obstacle avoidance and posture seeking behaviors have been simulated and run experimentally. Using neuro-fuzzy controller, the robot was run over an experimental of ($10m \times 10m$), with some obstacles. The code (Occam) begins by inputting an initial x - y locality, an orientation angle with respect to an x - y world coordinate, and the needed final posture. Results have shown that, robot moves around the plane seeking the final destination controlled by fuzzy-base rules with no colliding with surrounding boundaries. This is shown in Fig. 14. Posture seeking with an ability of speed alteration is shown in Fig. 14-a., whereas, in Fig. 14-b. illustrates robot seeking even with robot orientation change (robot was orientated initially by 90° degree up). This shows that neuro-fuzzy is capable of controlling the robot. The goal seeking algorithm is based on first rotating the robot until it does face the goal if it is not already facing it, than the algorithm try to reduce the distance between the robot and the goal to zero as shown in Fig. 11., where the distance to goal d_g (Baxter and Bumby, 1995) :

$$d_g = \sqrt{(x_g - x_v)^2 + (y_g - y_v)^2} \text{ and, heading error } \theta_{he} = (\theta_v + \theta_g)$$

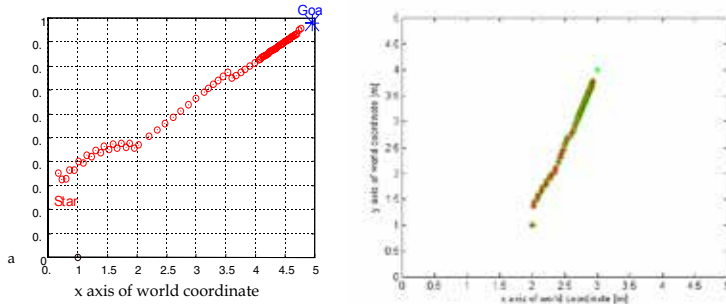


Fig. 14. Robot posture seeking.
 (a: Slow in speed close to target),
 (b: Posture seeking even; orientation change)

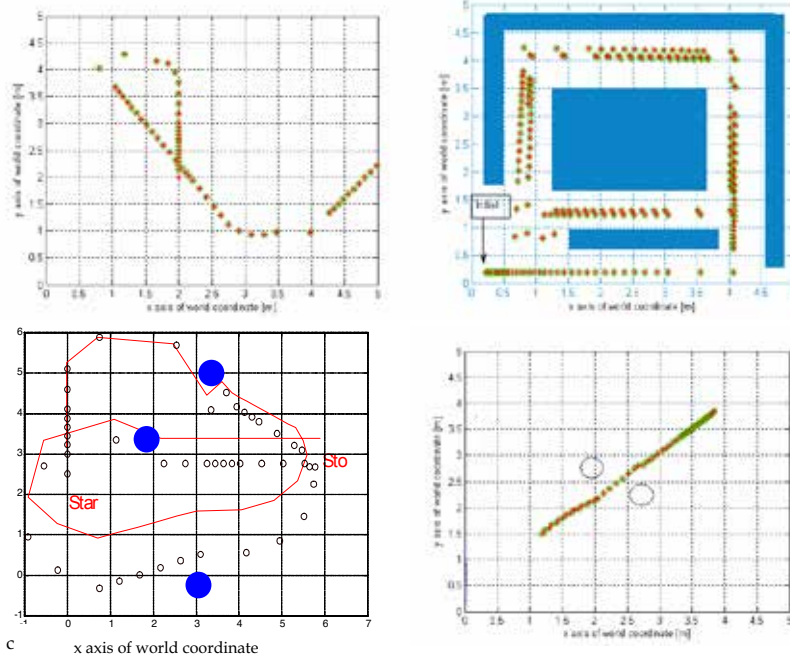


Fig. 15. Robot collision avoidance.
 (a: Easy movement with no collision),
 (b: Posture seeking, even orientation change)
 (c: Easy movement),
 (d: Posture seeking)

8.3 Robot Collision Avoidance

The above neuro-fuzzy controller has been divided in two parts: the obstacle avoidance routine and a goal seeking routine. Obstacle avoidance routine has been tested by simulation and by experiment. The goal seeking simulation was shown in Fig. 14., where the robot starts from known position and orientation in the world coordinate and stopped at the location of the desired goal position. The control algorithm enables the robot to start from a known position and stop at the specified goal position. If the robot encounters any obstacle in its way, the obstacle avoidance controller will take control of the mobile robot until the path of the robot is clear from any obstacle within the seeing range of the robot. This is exposed in Fig. 15. After the path is clear of any obstacle the goal seeking controller will be active again. The designed neuro-fuzzy controller was capable of controlling the mobile robot system with smooth translation from goal seeking to obstacle avoidance routines. Obstacle avoidance routine is much complicated due to needed decisions to turn left, turn right, move slowly, even to move backward. Goal seeking is simpler in term of implementation. This due to the need to minimize the distance between an initial and final robot posture.

9. Conclusions

In this chapter a neuro-fuzzy -based embedded controller has been successfully used to drive a mobile robot run on a Transputer system. The Transputer ability, as an embedded controller, to perform a demanding parallel fuzzy controller, has been demonstrated in the real-time. Experimental testing of the mobile robot for obstacle avoidance behavior has been reported. Sensor fusion (of ultrasonic data) in the mobile robot navigation have been accomplished successfully. The neuro-fuzzy controller and how it was accomplished have been discussed in details. Two results of behaviors have been demonstrated. Obstacle avoidance, and posture seeking. Results have shown that; a Transputer computation system was a very suited environment to achieve an implementation of a neuro-fuzzy system for both obstacle avoidance and posture seeking. That was due to the parallel nature of computation. Sensory data were processed simultaneously. This rather gives the ability to have a quick decision regarding the mobile robot behavior.

10. References

- Welgarz E., (1994), A General Robot Controller Using Transputer Based Hardware-Software System, *European Robotics and Intelligent Systems Conference (EURISCON-94)*, August 22-26, Malaga, Spain, pp. 1055-1064.
- Hu H.; Probert P. J.; & Rao B. Y. S. (1989), A Transputer Architecture for Sensor-Based Autonomous Mobile Robots. *Proceedings of the IEEE Int. Workshop on Intelligent Robots and Systems*.
- Iida S. & Yuta, S. (1991), Vehicle Command System and Trajectory Control for Autonomous Mobile Robots. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, Osaka, Japan, IROS'91, pp. 212-217.
- AL-Gallaf E. (2006), A Mobile Robotics : An Intelligent Testbet Control System, *An Internal Technical Report*, College of Engineering, University of Bahrain.

- Brady M.; Hu M. H.; & Probert P. (1993), Transputer Architecture for Sensor-guided Control of Mobile Robots, *Proceedings of World Transputer Congress'93*, Aachen, September 1993, Germany, pp. 118-133.
- Rusu P.; Petriu E.; Whalen T.; Cornell A.; & Spoelder H., (2003), Behavior-Based Neuro-Fuzzy Controller for Mobile Robot Navigation, *IEEE Transactions On Instrumentations and Measurement*, Vol. 52, No. 4.
- Benmounah, A. (1991), Transputer Control of an AGV, Design, Construction, and Testing of a Mobile Platform, *Ph.D. Thesis*, University of Reading.
- Maamri, M. (1991). Control of Mobile Platform in a Visually Monitored Environment, *Ph.D. Thesis*, University of Reading.
- Meystel, A. (1991). Autonomous Mobile Robots, Vehicles with Cognitive Control, *World Scientific Publishing Company Pte. Ltd.*
- Kim C. Ng & Trivedi Mohan M. (1998), A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multi-Robot Convoying, *IEEE Transactions On Systems, Man, And Cybernetics – PART B: Cybernetics*, Vol. 28, No. 6, pp. 829-840.
- Willgoss; & Iqbal R. J. (1999) Neurofuzzy Learning of Mobile Robot Behaviours: *Advanced Topics in Artificial Intelligence*. 12th Australian Joint Conference on Artificial Intelligence, AI'99. *Proceedings Lecture Notes in Artificial Intelligence*, Vol. 1747, pp. 278-90.
- Pennacchio, S.; Abissi, F.; Petralia, S. & Stendardo G. (2005), New Intelligent Controller for Mobile Robot Navigation in Unknown Environments: *WSEAS Transactions on Systems*, Vol. 4, No. 4, pp. 285-288.
- Tsoukalas H.; Houstis, N., & Jones, V. (1997), Neurofuzzy Motion Planners for Intelligent Robots, *Journal of Intelligent and Robotic Systems: Theory and Applications*, Vol. 19, No. 3, pp. 339-56.
- Hegazy, F., Fahmy, A., and El Refaie, M. (2004), An Intelligent Robot Navigation System Based on Neuro-fuzzy Control: Conference: *PRICAI 2004: Trends in Artificial Intelligence*. 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, Proceedings, 9-13.
- Shgiley J. E. (1969), Kinematics Analysis of Mechanisms, *McGraw-Hill, New York*, 1969. Muir P. F. & Neuman C. P. (1987), Kinematics Modeling of Wheeled Mobile Robots, *Journal of Robotics Systems*, Vol. 4, No. 2.
- Reister D.B. and Pin F.G. (1994), Time-Optimal Trajectories for Mobile Robots With Two Independently Driven Wheels, *The International Journal of Robotics Research*, Vol. 13, No. 1, pp. 38-54.
- Sarkar N.; Yun X. & Kumar V. (1994), Control of Mechanical Systems With Rolling Constraints: Application to Dynamic Control of Mobile Robots, *The International Journal of Robotics Research*, Vol. 13, No. 1, pp. 55-69.
- McKerrow J., (1991), Introduction to Robotics, *Addison-Wesley Publishing Company*.
- Pedrycz W. (1995), Fuzzy Control and Fuzzy Systems, *John Wiley & Sons International*.
- Beom R. & Cho H. (1995), A Sensor-Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning, *IEEE Transaction on Systems, Man, and Cybernetics*, Vol. 25, No. 3.
- Baxter J. W. & Bumby J. R. (1995), Fuzzy Control of a Mobile Robotic Vehicle, *Proceeding of Institute of Mechanical Engineering*, IMechE.

- Klir G.J. & Yuan B. (1995), *Fuzzy Sets And Fuzzy Logic, Theory And Applications*, *Prentice-Hall International*.
- Klafter R. D.; Chmielewski T. A. & Negin M. (1989), *Robotic Engineering an Integrated Approach*, *Prentice-Hall International*.
- Mir S. A.; Zinger D. S.; & Elbuluk M. E. (1994), *Fuzzy Controller for Inverter Fed Induction Machines*, *IEEE Transaction on Industry Applications*, Vol. 30, No. 1.

Mechanism and Control of Anthropomorphic Biped Robots

Hun-ok Lim^{1,2} & Atsuo Takanishi^{2,3}

¹*Department of Mechanical Engineering, Kanagawa University*

²*Humanoid Robotics Institute, Waseda University*

³*Department of Mechanical Engineering, Waseda University*

1. Introduction

Human beings, animals and some birds use their legs to move with great mobility, but we do not yet have a full understanding of their motion mechanism. The reason is that we lack man-made robots that use legs to obtain high mobility. We, especially, need two-legged robots that are able to adapt to human living environments such as offices and homes.

In order to analyze a human walking mechanism, some research groups studied about hydraulically-actuated or pneumatically-activated two-legged mechanisms in the 1960s. In 1973, a world's first full-scale anthropomorphic robot, WABOT-1 (WAseda roBOT-1), was constructed at Waseda University as shown in Figure 1. It was able to communicate with a human in Japanese and to measure the distance and direction of objects using external receptors such as an artificial mouth, ears and eyes (Kato *et al.*, 1973). Hydraulically powered, it used disproportionately large feet for stability. Also, it was able to grip and carry objects using the hands that were equipped with tactile sensors.

In 1983, WL-10R (Wasada Leg-10 Refined) was constructed using rotary type servo-actuators and carbon-fiber reinforced plastic. It was able to walk forwards and backwards, and turn on a flat plane. Based on the WL-10R, WHL-11 was developed by Wasada and Hitachi (Takanishi *et al.*, 1985). It walked more than 85 km at Tsukuba Science Expo'85. In 1986-1995, WL-12 series robots that have a trunk and a 2-DOF waist was constructed to simulate human walking motion including his upper body motion as shown in Figure 2 (Takanishi *et al.*, 1988; Li *et al.*, 1992; Yamaguchi *et al.*, 1995). They performed dynamic walking going up and down a stair with a height of 0.1 m and a slope of 10 deg (Takanishi *et al.*, 1990). Also, they did not fall down and walked under an unknown external force of 100 N applied to their backs (Takanishi *et al.*, 1991). However, the relative motion between their upper body and legs was difficult to be simulated.

An electrically powered human-sized biped humanoid robot, WABIAN (WAseda Biped humANoid), was constructed in 1996 for simulating human motion. It consists of a total of 35 DOF; two 3-DOF legs, two 10-DOF arms, two 2-DOF eyes, a 2-DOF neck and a torso with a 3-DOF waist (Setiawan *et al.*, 1999). After 1996, WABIAN series robots were developed, and achieved quasi-human follow walking, emotional motion, dancing motion and so on (Setiawan *et al.*, 1999; Lim *et al.*, 2004a).

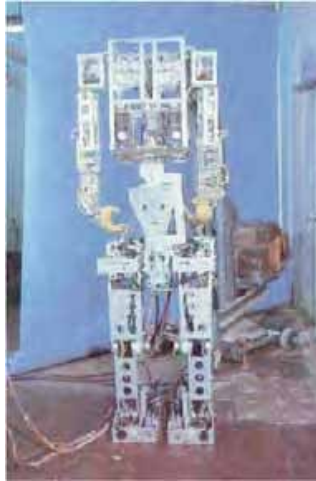


Fig. 1. Photo of WABOT-1.

Section 2 deals with the mechanism and software of WABIAN-RII (WAseda Biped humANoid-Refined II). The WABIAN-RII is a full-scaled biped humanoid robot that has 43 mechanical degrees of freedom (DOF). Section 3 addresses a locomotion pattern generation for biped humanoid robots. Its locomotion pattern is created online or offline according to the terrain conditions. Section 4 describes moment compensation control that is based on the motion of the waist and trunk. The motion of the waist and trunk is applied to maintain a good balance of the whole robot's body while the path of the legs is controlled according to the terrain. Section 5 describes impedance control to reduce the contact/impact force of the landing foot. Impedance parameters are changed in real time depending on the gait phase. Section 6 addresses walking experiments to verify the effectiveness of the mechanism of the WABIAN-series robots, the moment compensation control and the impedance control. Section 7 describes conclusions. Also, future works are discussed to make more reliable control systems.

2. Mechanisms of WABIAN-RII

An adult-sized humanoid robot should be constructed to properly simulate human motion. We have been developing WABIAN-series robots since 1996. In this section, the mechanical and electrical structure of WABIAN-RII is proposed.

2.1 Mechanical Mechanisms

WABIAN-RII consists of two 6-DOF legs, two 7-DOF arms, two 3-DOF hands, two 2-DOF eyes, a 4-DOF neck and a torso with a 3-DOF waist (Lim *et al.*, 2004a). Table 1 shows DOF of WABIAN-RII. Figure 3 shows a photo of WABIAN-RII. Its height is 1840 mm and its total weight is 127 kg. Table 2 and Table 3 show the weight and length distribution of each segment, respectively. The movable angle of the waist's roll and yaw is ± 20 deg and ± 45

deg, respectively. The movable angle of the waist's pitch is + 30 deg and -10 deg. The movable angle of the knee is + 90 deg and - 0 deg. Figure 4 shows the movable angles of the pitch, roll and yaw, respectively. Duralumin, GIGAS (YKK Corp.) and CFRP (Carbon Fiber Reinforced Plastic) are mainly employed as its structural materials.



Fig. 2. Photo of WL-12RIII.

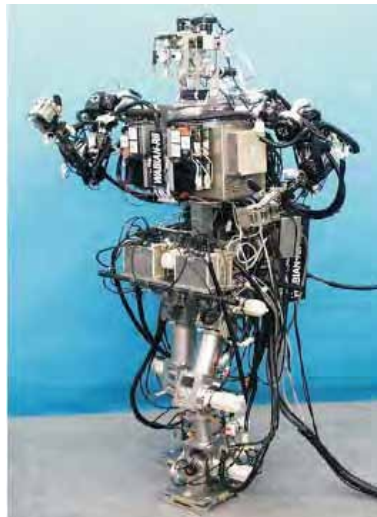


Fig. 3. Photo of WABIAN-RII.

2.2 Electrical systems

The trunk and legs of WABIAN-RII are driven by AC servo motors with reduction gears. The neck, hands and arms are actuated by DC servo motors with reduction gears. The eyes are driven by DC servo motors without reduction gears. Two CCD cameras that are attached to the head observe human environments. A force/torque sensor is attached to the wrist to detect interaction, and another force/torque sensor is attached between the ankle and the foot to measure ZMP. Also, a voice recognition system is installed.

WABIAN-RII is controlled by a PC/AT compatible computer PEAK-530 (Intel MMX Pentium 200 MHz CPU processor) run by MS-DOS 6.2/V (16-bit). TNT DOS-Extender SDK (Ver.6.1) is employed to extend the OS to a 32-bit. It has three counter boards, each with 24-bit 24 channels, three D/A converter boards, each with 12-bit 16 channels and an A/D converter board with differential 12-bit 16 channels to interface with sensors. The joint angles are sensed by incremental encoders attached to the joints, and the data are transferred to the computer through the counters. All the computations to control WABIAN-RII are carried out by the central computer, which runs the control program written in C language. The servo rate is 1 kHz. The computer system is mounted on the back of the waist and the servo driver modules are mounted on the upper part of the trunk. Electric power is the only external connection. Figure 5 shows the control system of WABIAN-RIL.

Parts	DOF
Eye	2 (1 pitch, 1 yaw) × 2
Neck	4 (2 pitch, 1 roll, 1 yaw)
Waist	3 (1 pitch, 1 roll, 1 yaw) × 2
Hip	3 (1 pitch, 1 roll, 1 yaw) × 2
Knee	1 (1 pitch) × 2
Ankle	2 (1 pitch, 1 roll) × 2
Shoulder	3 (1 pitch, 1 roll, 1 yaw) × 2
Elbow	1 (1 pitch)
Wrist	3 (1 pitch, 1 roll, 1 yaw) × 2
Hand	3 (thumb 1-DOF, index finger 1-DOF, the other fingers 1-DOF) × 2

Table 1. DOF of WABIAN-RII.

Parts	Weight (kg)	Parts	Weight (kg)
Foot	1.9 (×2)	Hand	0.5 (×2)
Ankle	7.8 (×2)	Wrist	0.5 (×2)
Knee	5.3 (×2)	Elbow	0.9 (×2)
Head	2.4	Shoulder	5.9 (×2)
Neck	6.4	Waist	20.6
Trunk	52.0		

Table 2. Segment weight of WABIAN-RII.

Parts	Length (mm)
Height	1840
Width	706
Depth	533
Between foot and ankle	142
Between ankle and knee	300
Between knee and hip	353
Between hip and waist	176
Between waist and neck	537
Between neck and head	332
Upper arm	263
Lower arm	247

Table 3. Segment length of WABIAN-RII.

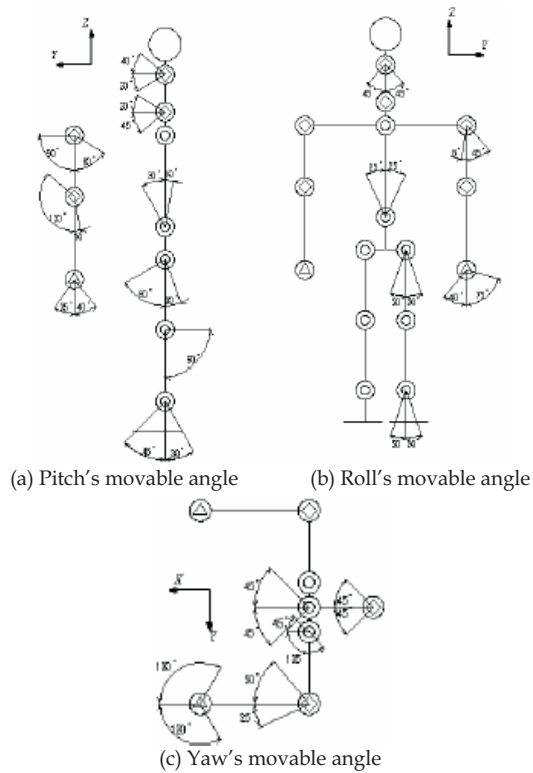


Fig. 4. Movable range of WABIAN-RII.

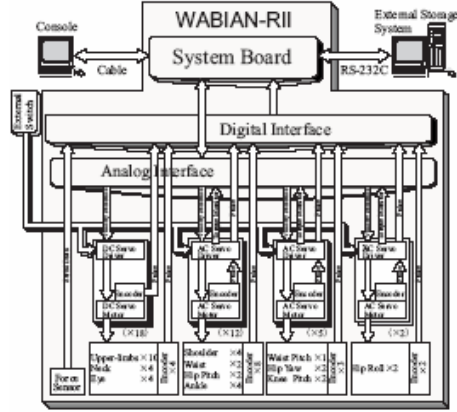


Fig. 5. Control system of WABIAN-RII.

3. Locomotion Pattern Generation

A biped locomotion phase can be classified into a contact, a single and a double supporting phase to generate a leg motion. During the single supporting phase, one foot is not constrained to the ground but the other foot is on the ground. As soon as the heel of the swinging foot reaches the ground, the single supporting phase is changed to the contact phase. If the toe and heel of the swinging foot contacts on the ground, the phase becomes a double supporting phase. Fig. 6 shows a locomotion phase for a leg pattern generation.

The smooth motion of the foot according to terrain conditions, x_f , can be generated by using a sixth degree polynomial considering the angle, the angular velocity, the angular acceleration and the height from the ground to the foot (Lim *et al.*, 2004b). Three constraints of the foot position and orientation come from the selection of initial and final values:

$$x_f(t_0) = x_0, x_f(t_f) = x_f, x_f(t_m) = x_m, \quad (1)$$

where t_0 , t_m and t_f are the initial, intermediate and final time of a step, respectively.

The position and orientation of the foot have an additional four constraints that are the zero initial and final velocity and acceleration:

$$\dot{x}_f(t_0) = \mathbf{0}, \dot{x}_f(t_f) = \mathbf{0}, \ddot{x}_f(t_0) = \mathbf{0}, \ddot{x}_f(t_f) = \mathbf{0}. \quad (2)$$

The sixth order polynomial for a foot motion is written as follows:

$$x_f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6, \quad (3)$$

and the velocity and acceleration along the path are clearly

$$\begin{aligned} \dot{x}_f(t) &= a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 + 6a_6 t^5, \\ \ddot{x}_f(t) &= 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 + 30a_6 t^4. \end{aligned} \quad (4)$$

Combining Equation (3) and Equation (4) with the seven constraints, the seven coefficients ($a_0 \cdots a_6$) can be obtained. Then, substituting these coefficients for Equation (3), the foot motion pattern is obtained. In addition, a knee and waist pattern is created by inverse kinematics.

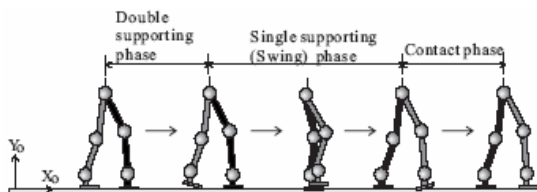


Fig. 6. Locomotion phase.

4. Moment Compensation Control

When a biped humanoid robot walks, moments generated by the motion of the legs make it unstable (Lim & Takanishi, 2005). To cancel the moments, a moment compensation control is discussed in this section.

4.1 Concept of Zero Moment Point

Consider a single supporting phase in which one foot is not constrained to the ground. As shown in Figure 7, the supporting polygon (stable region) is made by the contact points between the foot sole and the ground. To maintain the dynamic equilibrium of a biped humanoid robot, the ground reaction force should act at an appropriate point on the foot sole. It means that the ZMP (Zero Moment Point) exists at a point where the total forces and moments are equal to zero in the support polygon (Vukobratovic *et al.*, 1970). In this study, the ZMP is arbitrarily set in the stable region.



Fig. 7. ZMP in supporting polygon.

4.2 Moment Compensation

A world coordinate frame \mathcal{F} is fixed on the ground, and a moving coordinate frame \mathcal{F}' is attached to the center of the waist of a biped humanoid robot to define mathematical quantities. Considering that the humanoid robot is a set of particle model, the balancing moment around a contact point p in a supporting polygon can be expressed with respect to the world coordinate frame \mathcal{F} as

$$\sum_{i=1}^n m_i (r_i - r_p) \times (\ddot{r}_i + G) + T - \sum_{j=1}^n ((r_j - r_p) \times F_j + M_j) = \mathbf{0}, \quad (5)$$

where r_p is the position vector of the point p from the origin of the frame \mathcal{F} . m_i is the mass of the particle i of the biped humanoid robot. r_i and \ddot{r}_i denote the position and acceleration vectors of the particle i with respect to \mathcal{F} , respectively. G is the gravitational acceleration vector. T is the moment vector acting on the contact point p . r_j denotes the position vector of the particle j with respect to \mathcal{F} . F_j and M_j denote the force and the moment vectors acting on the particle j relative to the frame \mathcal{F} , respectively.

Let the ZMP be on the point p . The moment T will be a zero vector by the ZMP concept. Therefore, equation (5) can be rewritten relative to the moving frame \mathcal{F} as follows:

$$\begin{aligned} \sum_{i=1}^n m_i (\bar{r}_i - \bar{r}_{zmp}) \times (\ddot{\bar{r}}_i + \ddot{r}_q + G + \dot{\bar{\omega}} \times \bar{r}_i + 2\bar{\omega} \times \dot{\bar{r}}_i + \bar{\omega} \times (\bar{\omega} \times \bar{r}_i)) \\ - \sum_{j=1}^n ((\bar{r}_j - \bar{r}_{zmp}) \times \bar{F}_j + \bar{M}_j) = \mathbf{0}, \end{aligned} \quad (6)$$

where \bar{r}_{zmp} is the position vector of the ZMP with respect to \mathcal{F} . r_q is the position vector of the origin of \mathcal{F} from the origin of \mathcal{F} . $\bar{\omega}$ and $\dot{\bar{\omega}}$ denote the angular velocity and acceleration vectors, respectively.

In case that the upper body is modeled as a four-mass model as shown in Figure 8, three moment components generated by the motion of the particles of the legs, $M = [M_x \ M_y \ M_z]^T$, can be written as follows:

$$\begin{aligned} m_s (\bar{z}_s \ddot{\bar{x}}_s - \bar{x}_s \ddot{\bar{z}}_s) + m_t (\bar{z}_t - \bar{z}_{zmp}) (\ddot{\bar{x}}_t + \ddot{\bar{x}}_q + g_x) \\ - m_t (\bar{x}_t - \bar{x}_{zmp}) (\ddot{\bar{z}}_t + \ddot{\bar{z}}_q + g_z) \\ + m_w (\bar{z}_w - \bar{z}_{zmp}) (\ddot{\bar{x}}_w + \ddot{\bar{x}}_q + g_x) \end{aligned} \quad (7)$$

$$\begin{aligned} -m_w (\bar{x}_w - \bar{x}_{zmp}) (\ddot{\bar{z}}_w + \ddot{\bar{z}}_q + g_z) = -M_y(t), \\ m_s (\bar{y}_s \ddot{\bar{z}}_s - \bar{z}_s \ddot{\bar{y}}_s) + m_t (\bar{y}_t - \bar{y}_{zmp}) (\ddot{\bar{z}}_t + \ddot{\bar{z}}_q + g_z) \\ - m_t (\bar{z}_t - \bar{z}_{zmp}) (\ddot{\bar{y}}_t + \ddot{\bar{y}}_q + g_y) \\ + m_w (\bar{y}_w - \bar{y}_{zmp}) (\ddot{\bar{z}}_w + \ddot{\bar{z}}_q + g_z) \end{aligned} \quad (8)$$

$$\begin{aligned} -m_w (\bar{z}_w - \bar{z}_{zmp}) (\ddot{\bar{y}}_w + \ddot{\bar{y}}_q + g_y) = -M_x(t), \\ m_s (\bar{x}_s \ddot{\bar{y}}_s - \bar{y}_s \ddot{\bar{x}}_s) + 2(\bar{x}_s \dot{\bar{x}}_s + \bar{y}_s \dot{\bar{y}}_s) \bar{\omega}_z + (\dot{\bar{x}}_s^2 + \dot{\bar{y}}_s^2) \dot{\bar{\omega}}_z \\ + m_t (\bar{x}_t - \bar{x}_{zmp}) (\ddot{\bar{y}}_t + \ddot{\bar{y}}_q + g_y) \\ - m_t (\bar{y}_t - \bar{y}_{zmp}) (\ddot{\bar{x}}_t + \ddot{\bar{x}}_q + g_x) \\ + m_w (\bar{x}_w - \bar{x}_{zmp}) (\ddot{\bar{y}}_w + \ddot{\bar{y}}_q + g_y) \\ - m_w (\bar{y}_w - \bar{y}_{zmp}) (\ddot{\bar{x}}_w + \ddot{\bar{x}}_q + g_x) = -M_z(t), \end{aligned} \quad (9)$$

where m_s denotes the mass of both shoulders including the mass of the arms. m_t is the mass of the torso including the head, shoulders and arms, and m_w is the mass of the waist. $\bar{r}_t = [\bar{x}_t \ \bar{y}_t \ \bar{z}_t]^T$ and $\bar{r}_w = [\bar{x}_w \ \bar{y}_w \ \bar{z}_w]^T$ are the position vectors of the neck and the waist with respect to the \mathcal{F} , respectively. $\bar{r}_s = [\bar{x}_s \ \bar{y}_s \ \bar{z}_s]^T$ is the position vector of the shoulder with respect to the neck frame.

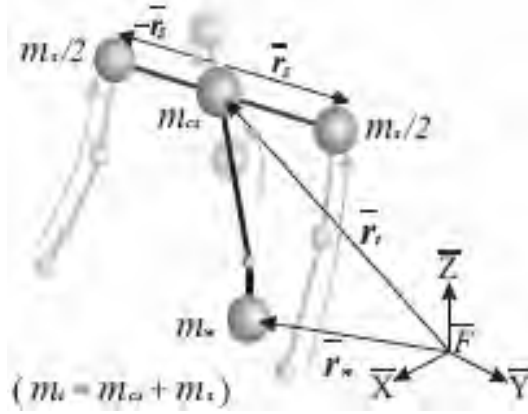


Fig. 8. Approximate model of upper body.

These moment equations are interferential and non-linear because each equation has the same variables, \bar{z}_w and \bar{z}_t . Thus, it is difficult to derive analytic solutions from equations (7), (8) and (9). We assume that neither the waist nor the trunk particles move vertically, and the trunk arm rotates on the horizontal plane only as

$$\ddot{\bar{z}}_t = 0, \ddot{\bar{z}}_w = 0, \ddot{\bar{z}}_q = 0, \bar{z}_t - \bar{z}_{zmp} = const., \bar{z}_w - \bar{z}_{zmp} = const. \quad (10)$$

We put the terms about the motion of the particles of the upper body on the left-hand side as unknown variables, and the terms about the moments generated by the particles of the lower-limbs on the right-hand side as known parameters. The decoupled and linearized ZMP equations can be written as

$$\hat{M}_{yt} + \hat{M}_{yw} = \hat{M}_y(t), \quad (11)$$

$$\hat{M}_{xt} + \hat{M}_{xw} = \hat{M}_x(t), \quad (12)$$

$$m_t l^2 \theta_t = \hat{M}_z(t), \quad (13)$$

where

$$\begin{aligned} \hat{M}_{yt} &= m_t(\bar{z}_t - \bar{z}_{zmp})\ddot{\bar{x}}_t - m_t g_z \bar{x}_t, \\ \hat{M}_{yw} &= m_w(\bar{z}_w - \bar{z}_{zmp})\ddot{\bar{x}}_w - m_w g_z \bar{x}_w, \\ \hat{M}_{xt} &= -m_t(\bar{z}_t - \bar{z}_{zmp})\ddot{\bar{y}}_t + m_t g_z \bar{y}_t, \\ \hat{M}_{xw} &= -m_w(\bar{z}_w - \bar{z}_{zmp})\ddot{\bar{y}}_w + m_w g_z \bar{y}_w, \\ \hat{M}_y(t) &= -M_y - (m_t(\bar{z}_t - \bar{z}_{zmp})\ddot{\bar{x}}_q + m_t g_z \bar{x}_{zmp} \\ &\quad + m_w(\bar{z}_w - \bar{z}_{zmp})\ddot{\bar{x}}_q + m_w g_z \bar{x}_{zmp}), \\ \hat{M}_x(t) &= -M_x - (-m_t(\bar{z}_t - \bar{z}_{zmp})\ddot{\bar{y}}_q - m_t g_z \bar{y}_{zmp} \end{aligned} \quad (14)$$

$$\begin{aligned} \hat{M}_z(t) = & -m_w(\bar{z}_w - \bar{z}_{zmp})\ddot{y}_q - m_w g_z \bar{y}_{zmp}, \\ & -m_t(\bar{x}_t - \bar{x}_{zmp})(\ddot{y}_t + \ddot{y}_q) \\ & -m_t(\ddot{y}_t - \ddot{y}_{zmp})(\ddot{x}_t + \ddot{x}_q) \\ & +m_w(\bar{x}_w - \bar{x}_{zmp})(\ddot{y}_w + \ddot{y}_q) \\ & -m_w(\ddot{y}_w - \ddot{y}_{zmp})(\ddot{x}_w + \ddot{x}_q), \end{aligned}$$

where θ_t is the vertical angle of the trunk. l is the length between the neck and the shoulder. \hat{M}_{xt} and \hat{M}_{yt} denote the roll's and the pitch's trunk components of the moments, respectively. \hat{M}_{xw} and \hat{M}_{yw} the moment components of the waist's roll and pitch, respectively. \hat{M}_{xw} , \hat{M}_{yw} , \hat{M}_{xt} and \hat{M}_{yt} become the known functions because they are calculated by the motion of the lower-limbs and the time trajectory of the ZMP. In steady walking, they are periodic functions because each particle of the biped humanoid robot and the ZMP move periodically with respect to the frame \mathcal{F} . Comparing the Fourier transform coefficients of both sides of each equation, the approximate periodic solutions of the pitch's and roll's waist and trunk, \bar{y}_w , \bar{x}_w , \bar{y}_t , \bar{x}_t and θ_t can be obtained.

4.3 Recursive Calculation

It is difficult to calculate the compensatory motion due to six unknown variables such as \bar{y}_w , \bar{x}_w , \bar{y}_t , θ_t and θ_w . Each equation is represented as a Fourier series. Using the Fourier transform coefficients, the approximate periodic solutions of the pitch's and roll's waist and trunk can be obtained. The approximate solutions are substituted into the moment equation (6), and the errors of moments generated by the waist and trunk motions are calculated according to the planned ZMP. These errors are accumulated to the right-hand side of equations (11), (12) and (13). The approximate solutions are computed again. These computations are repeated until the errors fall below a certain tolerance level. As a result, the strict periodic solutions of the nonlinear equations are obtained by a convergent regularity. The limit value of an accumulated moment error on each axis, E_n , is estimated as follows:

$$\begin{aligned} E_n &= \frac{2E_{n-1} + e_{n-1}}{2} \\ n &= 3, 4, 5, \dots, \quad E_1 = 0, \quad E_2 = e_1. \end{aligned} \quad (15)$$

where e_n is the n th moment error.

Figure 9 shows the block diagram for the compensatory motion of the waist and trunk. We numerically verified the convergence of the solutions using computer simulations. A 30 deg turning walking was simulated with the step time of 1.28 s/step and the step width of 0.2 m/step. Among the ten, the biped humanoid robot turned at the fifth and sixth step. Figure 10 shows the convergence of the turning walking. In this simulation, the errors of the pitch's, roll's and yaw's moments drop below 0.1 Nm after ten iterations.

Figure 11 shows the convergence of the trunk motion depending on the walking speed when the trunk length is 0.6[m]. In these simulations, the errors of the ZMP are dropped below 1.0[mm]. We can see that the recursive method does not take so much time in calculating the compensatory motion of the trunk and waist.

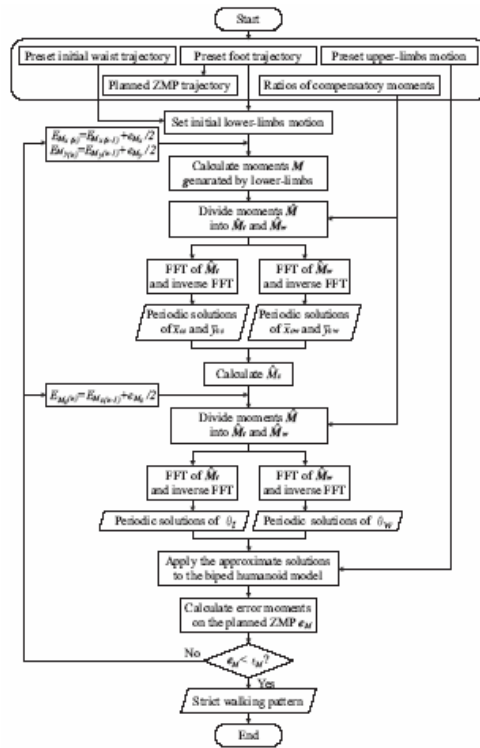


Fig. 9. Calculation of compensatory motion.

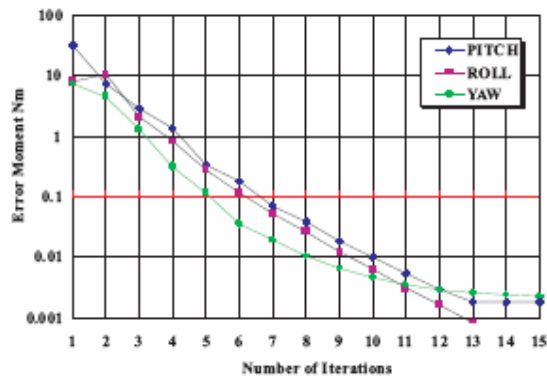


Fig. 10. Moment errors in turning walking.

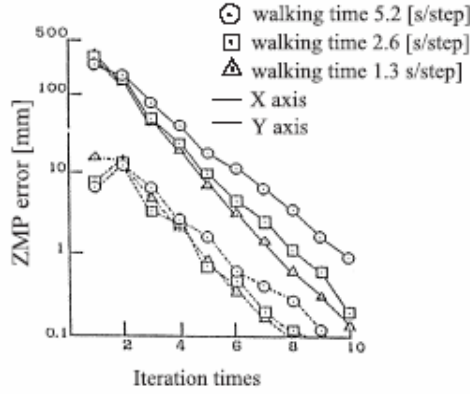


Fig. 11. Convergence of trunk

4.4 Trunk Motion for Complete Walking

This compensation control is also applicable to a complete walking. By regarding the complete walking motion as one periodic walking motion and applying the control method to it, the compensatory motion of the waist and trunk in steady and transitional walking can be derived. However, the biped humanoid robot must have a long period of standing time before starting and after stopping.

Assuming that the waist is immovable, the pitch's and the roll's moment components of equation (6) can be changed as

$$(\bar{z}_t - \bar{z}_{zmp})\ddot{\bar{x}}_t - g_z \bar{x}_t = -A(t), \quad (16)$$

$$(\bar{z}_t - \bar{z}_{zmp})\ddot{\bar{y}}_t + g_z \bar{y}_t = B(t), \quad (17)$$

where

$$A(t) = \frac{M_y(t) + m_w(\bar{z}_w - \bar{z}_{zmp})\ddot{\bar{x}}_w - m_w g_z \bar{x}_w}{m_t},$$

$$B(t) = \frac{M_x(t) - m_w(\bar{z}_w - \bar{z}_{zmp})\ddot{\bar{y}}_w - m_w g_z \bar{y}_w}{m_t} \quad (18)$$

Consider only the motion of the trunk around the pitch's axis to investigate the compensatory motion. The transfer function $\bar{y}_t(\omega)$ in the frequency domain can be expressed as

$$\bar{y}_t(\omega) = \frac{2p}{\omega^2 + p^2} q = \left(\frac{1}{p - j\omega} + \frac{1}{p + j\omega} \right) q, \quad (19)$$

where

$$p = \sqrt{\frac{g_z}{\bar{z}_t - \bar{z}_{zmp}}}, \quad q = -\frac{1}{2g_z} \sqrt{\frac{g_z}{\bar{z}_t - \bar{z}_{zmp}}}. \quad (20)$$

According to the ZMP concept, \bar{z}_{zmp} in equation (20) is zero. Equation (19) is generally known as Lorentz function, and its primitive function is written as

$$\bar{y}_t(t) = qe^{-p|t|}. \quad (21)$$

We can imagine from equation (21) that the casual law may not be applied anymore. If the walking speed of the biped humanoid robot is increased, it goes without saying that $\bar{y}_t(t)$ affects stability more badly. The trunk, therefore, should be in motion earlier than the shift of the ZMP on the ground to cancel the moments produced.

The relationship between the motion of the trunk and the applied force was simulated. When the biped humanoid robot was not in motion, an impulse moment 1000 Nm was applied to the trunk's pitch. Figure 12 shows the pitch's motion of the trunk. In this simulation, we can see that the compensatory motion of the trunk should begin with a view of balancing before and after the impulse moment is applied to the biped humanoid robot.

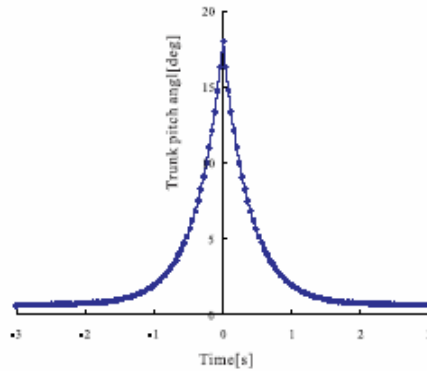


Fig. 12. Motion of trunk's pitch

5. Impedance Control

It is difficult for a biped humanoid robot to avoid instability caused by the impact/contact forces produced between the landing foot and the ground during dynamic walking. So, an impedance control is discussed in this section to guarantee higher stability with high speed walking (Lim *et al.*, 2004c)

When a human walks, he controls his leg muscles for shock absorption and posture. His muscles are relaxed to absorb the impact/contact force just before his landing foot makes an initial contact, while his muscles are hardened to maintain the balance after landing (Kazerooni & Waibel, 1988; Park & Kim, 1999). We pay attention to the elasticity of the biped humanoid robot like a human.

Suppose that the biped foot is connected to the ground with a spring and a damper model. A force/torque sensor is mounted between the foot and the ankle to measure the impact/contact force as shown in Figure 13. Also, an ankle coordinate frame F_a is fixed on the center of the ankle, and an end-effector coordinate frame F_z is attached at the ZMP.

The impact/contact force between the landing foot and the ground, $F \in \mathbb{R}^3$, can be written as

$$\begin{aligned} M_d \ddot{p} + D_d \dot{p}_e + K_d p_e &= F, \\ p_e &= p - p_d, \end{aligned} \quad (22)$$

where $M_d \in \mathbb{R}^{6 \times 6}$ is the desired mass matrix. $D_d \in \mathbb{R}^{6 \times 6}$ and $K_d \in \mathbb{R}^{6 \times 6}$ are the damping and stiffness matrices, respectively. $p \in \mathbb{R}^6$ and $p_d \in \mathbb{R}^6$ denote the actual and desired position vectors of the foot, respectively. The joint angles of the leg can be measured by the proper encoders attached to each joint. Then, p is calculated by forward kinematics. The reaction force F is measured by the force/torque sensor.

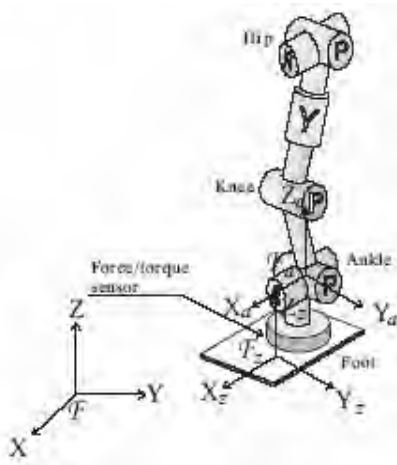


Fig. 13. Ankle and ZMP coordinate frames.

The angular velocity vector $\dot{\theta}_e$ that is commanded to the landing leg can be obtained as follows:

$$\dot{\theta}_e = J^{-1}(\theta) \dot{p} \quad (23)$$

where

$$\dot{p} = M_d^{-1} \int (F - D_d \dot{p}_e - K_d p_e). \quad (24)$$

For the absorption of the impact forces and the stable posture of the biped humanoid robot, an impedance control method is employed as shown in Figure 14. During the whole walking cycle, the position control based on the walking pattern is basically employed. The joint angles are measured by the encoders, and the actual position of the foot is calculated by using forward kinematics. The parameters in the impedance control are modulated depending on the gait phase as follows:

(1) in the swinging phase, only the position control is employed, (2) in the contact/double supporting phase, the impedance control with large viscosity is employed to reduce the impact/contact force, (3) in the first half of the single supporting phase, the impedance

control with large stiffness is applied to increase the momentum reduced, and (4) in the last half of the single supporting phase, the walking pattern changed by the impedance control is interpolated to the desired walking pattern.

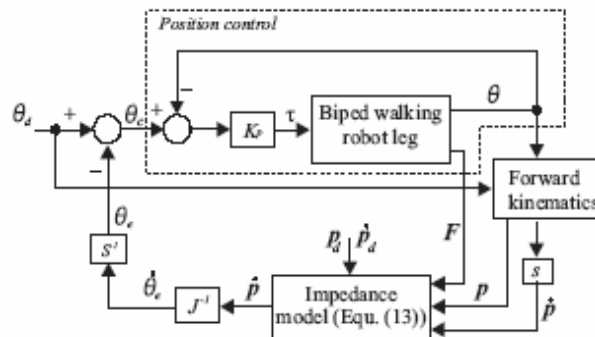


Fig. 14. Impedance control systems.

6. Experiments

For a biped humanoid robot to walk stably on terrains, the moment compensation control and the impedance control were described in the above sections. In this section, the effectiveness of the control methods is tested through dynamic walking experiments.

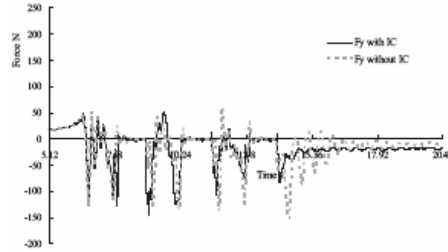
6.1 Impedance Control Test

The position control and the impedance control are experimented using a WABIAN serious robot. In this experiment, the impedance control was applied to the ankle joint. Dynamic walking is realized with the step speed of 1.28 s/step and the step length of 0.15 m/step (Lim *et al.*, 2004c).

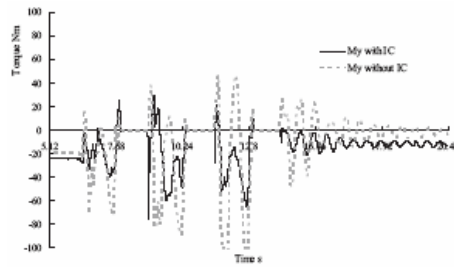
The contact forces and moments of the left foot are shown in Figure 15 (a)-(b), respectively. The experimental results are plotted after 5.12 s. The pitch's contact forces and moments are largely reduced in the last step. Figure 16 (a) and (b) illustrate the desired and actual x-and y-ZMP. The ZMP errors in the impedance control are small. So, we can say that the position-based impedance control is effective in suppressing the impact/contact forces and moments.

6.2 Compensation Control Test

The walking direction and length can be determined by a vision tracking and an auditory system (Lim *et al.*, 2004b). The vision system is initialized, and both hands of a human are specified as tracking points during the walking. The auditory system recognizes the walking length that is defined by compiling the text file of grammar rules. Using the locomotion pattern generator based on the visual and the auditory information, a locomotion pattern with 25 steps (four steps forwards, five steps to the right, four steps backwards, six steps to the left and six steps on the spot) is created.

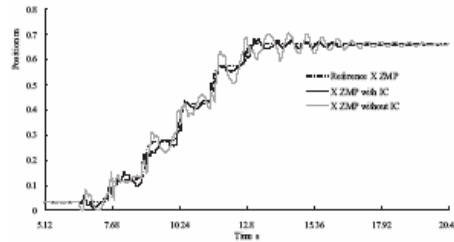


(b) Pitch's reaction force F_y .

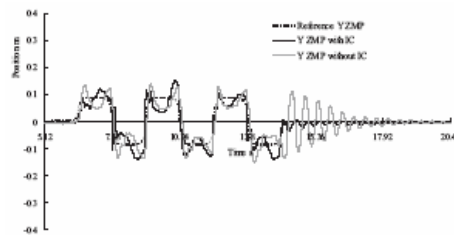


(a) Pitch's moment M_y .

Fig. 15. Reaction forces and Moments on the left foot. IC denotes the impedance control.



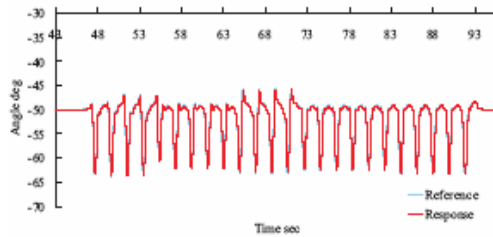
(a) X-ZMP



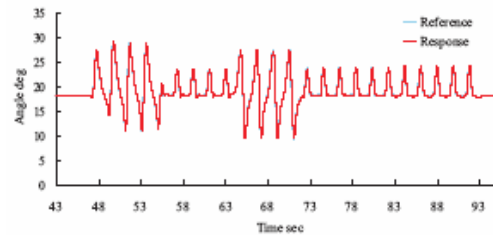
(b) Y-ZMP

Fig. 16. ZMP trajectories under two different controls. IC: Impedance control.

In the experiment, dynamic walking with 25 steps was achieved with the walking speed of 0.96 s/step using the moment compensation control. The response trajectories show a good accuracy of following the reference joint trajectories as shown in Figure 17. It means that the compensatory motion of the waist and trunk is used effectively.



(a) Pitch's angle of knee joint.



(b) Pitch's angle of hip joint.

Fig. 17. Angle response of the right leg.

7. Conclusion and Discussion

The mechanism of a 43 DOF biped humanoid robot WABIAN-R11 was described to simulate human motions was described. For the humanoid robot to walk stably on various terrains, its functions should be separated. Its waist and trunk carries out the function of stability, while its legs work out the function of the locomotion. A locomotion pattern generation was discussed which used a polynomial. In the pattern generation, the direction of the walking and the length of the step can be determined by visual and auditory information. Also, a moment compensation control was presented to balance its body, which was based on the motion of the waist and trunk. In addition, an impedance control method was discussed to absorb the impact/contact forces generated between the landing foot and the ground, which can adjust the impedance parameters to make relaxed and hardened motions like a human. The experimental results show that the impedance and the moment compensation control are effective for the sock absorption and stability. However, for a more reliable locomotion, biped humanoid robots must be able to adjust the time of the locomotion. If the walking time is suddenly changed, unnecessary acceleration is able to occur, and the errors of the ZMP can be increased. To solve these problems, a new control method will be introduced which uses the motion of the upper-limbs, and the impedance control should be considered in all the joints. Especially, the moment of the yaw should be compensated to deal with high speed walking.

7. References

- Kato, I.; Ohteru, S.; Kobayashi, H.; Shirai, K. & Uchiyama, A. (1973). Information-power machine with senses and limbs, *Proc. CISM-IFTToMM Symp. On Theory and Practice of Robots and Manipulators*, pp. 12-24
- Takanishi, A.; Ishida, M.; Yamazaki, Y. & Kato, I. (1985). The realization of dynamic walking by the biped walking robot, *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 459-466
- Takanishi, A.; Egusa, Y.; Tochizawa, M.; Takebayashi, T. & Kato, I. (1988). Realization of dynamic walking stabilized with trunk motion, *Proc. CISM-IFTToMM Symp. On Theory and Practice of Robots and Manipulators*, pp. 68-79
- Li, Q.; Takanishi, A. & Kato, I. (1992). Learning control of compensative trunk motion for biped walking robot based on ZMP stability criterion, *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 597-603
- Yamaguchi, J.; Takanishi, A. & Kato, I. (1995). Experimental development of a foot mechanism with shock absorbing material for acquisition of landing surface position information and stabilization of dynamic biped walking, *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2892-2899
- Takanishi, A.; Lim, H.; Tsuda M. & Kato I. (1990). Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface, *Proc. IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pp. 323-329
- Takanishi, A.; Kumeta, M.; Matsukuma, K.; Yamaguchi, J. & Kato, I. (1991). Development of control method for biped walking under unknown external force acting in lateral plane *RSJ Annual Conf. on Robotics*, pp. 321-3224
- Setiawan, S.; Hyon, S.; Yamaguchi, J. & Takanishi, A. (1999). Physical interaction between human and a bipedal humanoid robot: realization of human-follow walking *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 361-367
- Lim, H.; Ishii A. & Takanishi A. (2004a). Emotion-based biped walking, *Int. J. Information, Education and Research in Robotics and Artificial Intelligence*, Vol.22, No.5, pp.577-586
- Lim, H.; Kaneshima, Y. & Takanishi, A. (2004b). Online walking pattern and balance motion for a biped humanoid robot having a trunk, *Int. J. Human-friendly Welfare Robotic Systems*, Vol.5, No.1, pp. 26-35
- Lim, H. & Takanishi A. (2005). Compensatory motion control for a biped walking robot, *Proc. Int. J. Information, Education and Research in Robotics and Artificial Intelligence*, Vol.23, No.1, pp.1-11
- Vukobratovic M., Frank A. & Juricic D. (1970). On the stability of biped locomotion, *IEEE Trans. Biomedical Engineering*, vol. 17, no. 1, pp. 25-36
- Lim H., Setiawan S. & Takanishi A. (2004c). Position-based Impedance Control of Biped Humanoid Robot, *Int. J. Advanced Robotics*, Vol.18, No.4, pp.415-435
- Kazerooni K. & Waibel B. (1988). Theory and experiment on the stability of robot compliance control *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 71-87
- Park J. & Cho H, (2000). An online trajectory modifier for the base link of biped robots to enhance locomotion stability, *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3353-3358

Bio-mimetic Finger

Human like morphology, control & motion planning for intelligent robot & prosthesis

Emanuele Lindo Secco, Giovanni Magenes
University of Pavia, Bio-Engineering Laboratory - DIS - Italy
Emanuele@bioing.unipv.it, Giovanni@bioing.unipv.it

1. Introduction

The aim of this chapter is to summarize results we have obtained over the past five years of the bio-mimetic approach to robotics and/or prosthetic applications .

Focusing on multiple degrees of freedom (*dof*) and multi-articulated limbs we will suggest technical solutions for controlling and planning the motion of an artificial limb in such a way that it will match the natural pattern of 'natural movement'.

We considered three items when regarding the problem of moving a 3 *dof* artificial finger:

- First, we describe the morphology of an ideal bionic finger through imitation of the anatomical shape of the human finger as well as optimization criteria of the grasping function;
- Second, we introduce a mathematical model of multi-articular human limbs suggesting a solution to the problem of acting multiple-links with unimodal bell shape speed profiles;
- Third, we propose a natural planning of finger kinematics and dynamics. The plan will be based on desired and proprioceptive signals driving from a neural network (*nn*) controller: after a learning phase, the 'intelligent' outcome (AI system) will be able to reproduce the effective dynamics of natural movement. Furthermore the *nn* exhibits the capacity to generate 'never before seen' movements.

Future perspectives on control strategies will be discussed.

Abbreviations

neural network	<i>Nn</i>
degree of freedom	<i>Dof</i>

2. Bio-mimetic Morphology

2.1 Introduction

The design of an artificial finger, specifically the design of the three phalanges of an artificial finger, depends on its aim. In this work we have simplified the representation of the inter-segments by modeling the phalanges with cylindrical rods,. The main goal is to control

movement and not to grasp an object, which implies more complex concepts of contact mechanics. This approach simplifies the computation of the dynamic equations. Other authors have preferred different approaches, for example grasping analysis or modeling, in which the shape of the phalanges is more important.

In the design of an artificial finger we must first fix the dimensions of each phalanx. We started by considering a Fibonacci series, as the basis for computing the right proportional structure of the finger (Gupta et al., 1998). We also consulted the ergonomics literature of Garrett, 1971, Buchholz et al., 1992 and Pheasant, 1996. After fixing the lengths, we developed the shape.

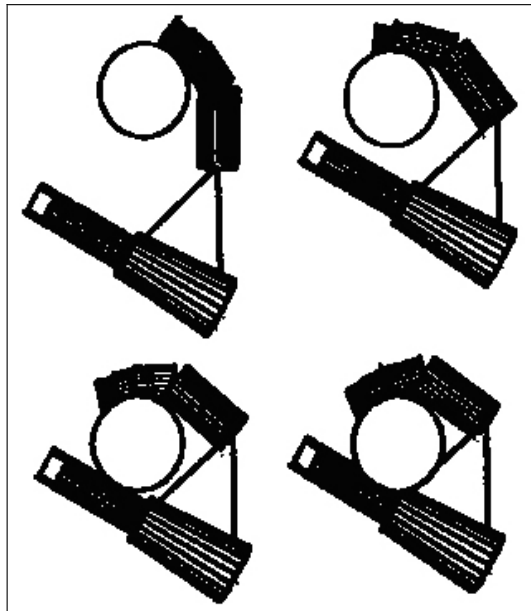


Fig. 1. The model proposed by Esteki & Mansour, 1997, includes Thumb, Index Finger, Palm and Inter-segmental Joints. In the sequence of the images, the cylinder is successfully grasped between the index finger, thumb and palm. Initial, final and two intermediate pictures of the hand motion are presented (figure from the cited paper).

2.2 Some Examples

We considered the work of Esteki & Mansour (1997) to model hand positioning and grasping force. They treat the hand as a jointed multi-body system. Thumb, Index Finger, Palm and Inter-segmental Joints are included in the model (Fig. 1). Each phalanx is represented by a cylindrical rigid body whose dimensions are measured for one subject. Segment masses and principal moments of inertia are computed on the basis of the volume and geometry of each segment, using a specific mass of 1.1 g/cm^3 . Moreover, it is assumed that the shape of the contacting thumb surface is a section of a sphere and that only the distal phalanx of the thumb is in contact with the object.

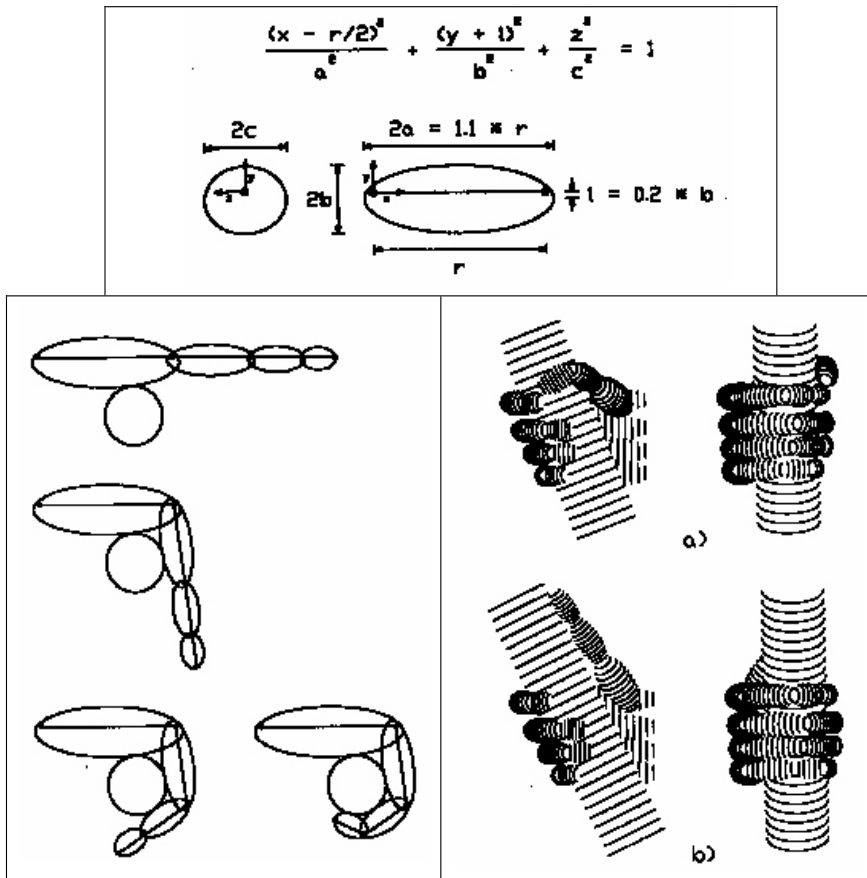


Fig. 2. TOP PANEL. Equations and graphical description of the three-dimensional geometry of the hand ellipsoidal segments; r = kinematic segment length, a = length semi-axis dimension, b = depth semi-axis dimension, c = breadth semi-axis dimension, l = ellipsoid depth offset. BOTTOM LEFT PANEL. Planar example of the power grasp algorithm from Buchholz & Armstrong, 1992; BOTTOM RIGHT PANEL. Example of the graphical display capabilities of the model: different view of the (a) transverse volar grasp and (b) diagonal volar grasp are shown (images from the cited paper).

Another interesting kinematical model was developed by Buchholz & Armstrong (1992) to simulate and predict the prehensile abilities of the human hand. The model is based on an algorithm that determines the contact between two ellipsoids (Fig. 2), which are used to approximate the geometry of the cutaneous surface of the hand segments.

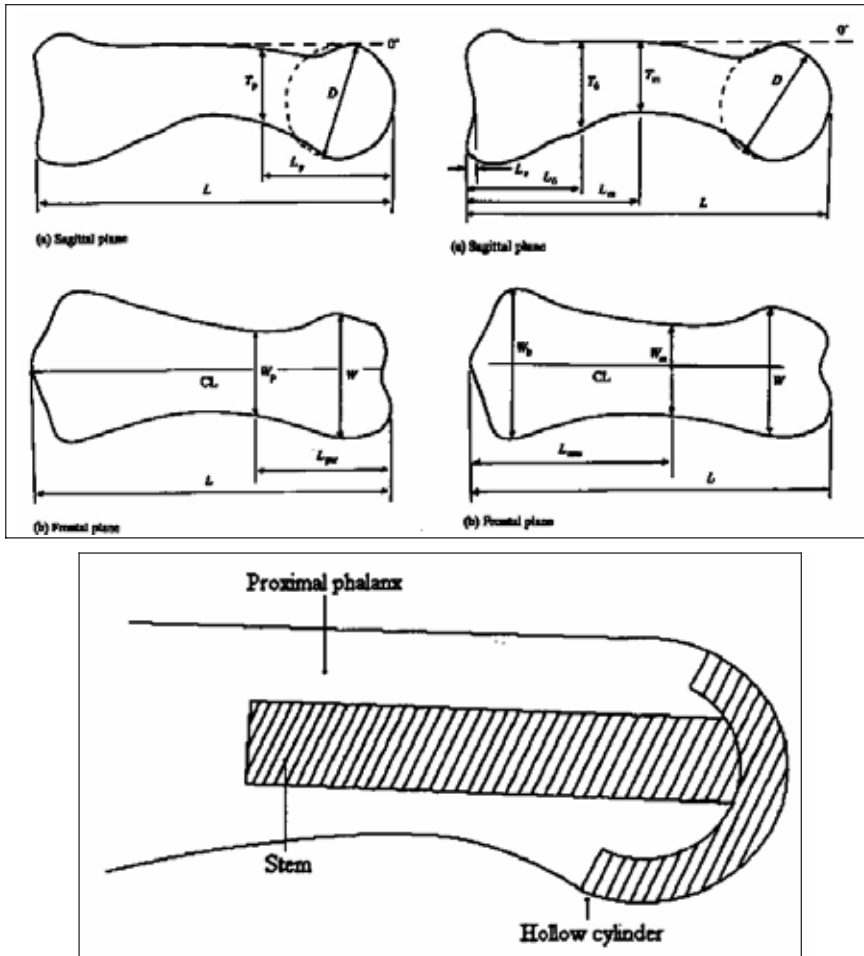


Fig. 3. TOP PANEL. MetacarpoPhalangeal (MP) dimensions and (dx) Proximal InterPhalangeal (PIP) dimensional parameters (imagine from). BOTTOM PANEL. Schema of a hollow PIP joint surface replacement prosthesis (images from Ash & Unsworth, 1996 & Ash et al., 2000).

Coefficients to estimate anthropometric parameters from hand length and breadth are incorporated in this model. The model was developed to predict or quantify the grip posture (Fig. 2, bottom panel). Moreover Buchholz collected data that would allow modeling the surface of each segment as a group of ellipsoids instead of a single ellipsoid.

Another approach to design artificial finger or portions of the finger came from the medical environment: phalanges bones from 83 *PIP joints* were dissected in order to determine the shape and size of the articular surface (Ash & Unsworth, 1996). Then, the dimensions were used in the design of surface replacement prosthesis for the PIP joint. Finally, from a statistical analysis of the parameters (Fig. 3, top panel), a cross-linked polyethylene (XLPE) surface replacement finger joint prosthesis (Fig. 3, bottom panel) was designed (Ash & Unsworth, 2000).

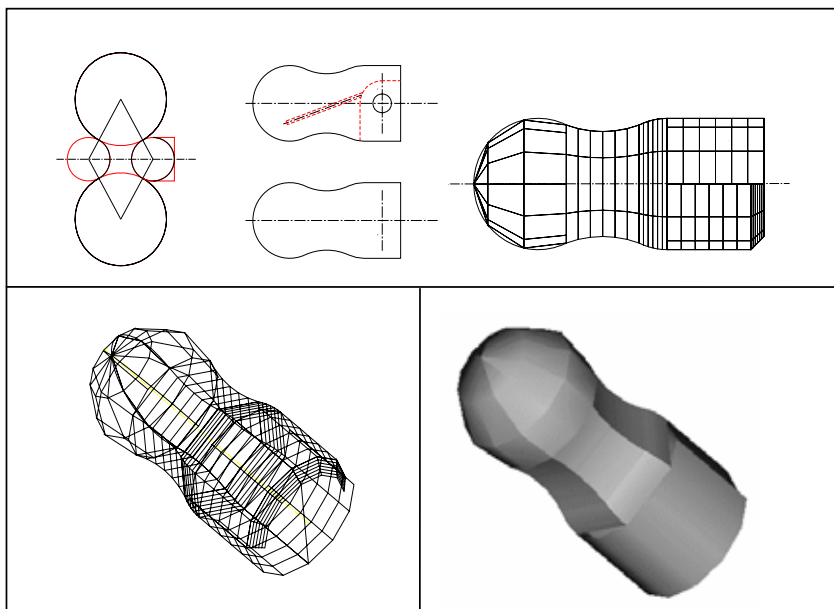


Fig. 4. TOP PANEL. Details on the rational design of the DIP phalanx for the artificial middle finger prosthesis. BOTTOM PANEL. The 3-D mesh (left) and rendering (right). Dimension is similar to the natural one (images from Secco, 1997).

PIP joint replacement is just a little portion of the entire finger and of the entire hand, whereas many attempts have been made by many laboratories for realizing artificial fingers and/or entire robotic or prosthetic hands in the last 10-20 years (for a rapid review see Rosheim, 1994).

In this context, we would like to introduce some proposals for the realization of a 3 *dof* robotic middle finger (Secco, 1997). The profiles of the phalanges imitate the human finger phalanges (Fig. 4), while the movement is guaranteed by the action of three wires (like human tendons) into the mechanical structure. Flexion is supported by 3 active tendons, extension is effected by pre-compressed springs into the three joints (Fig. 5).

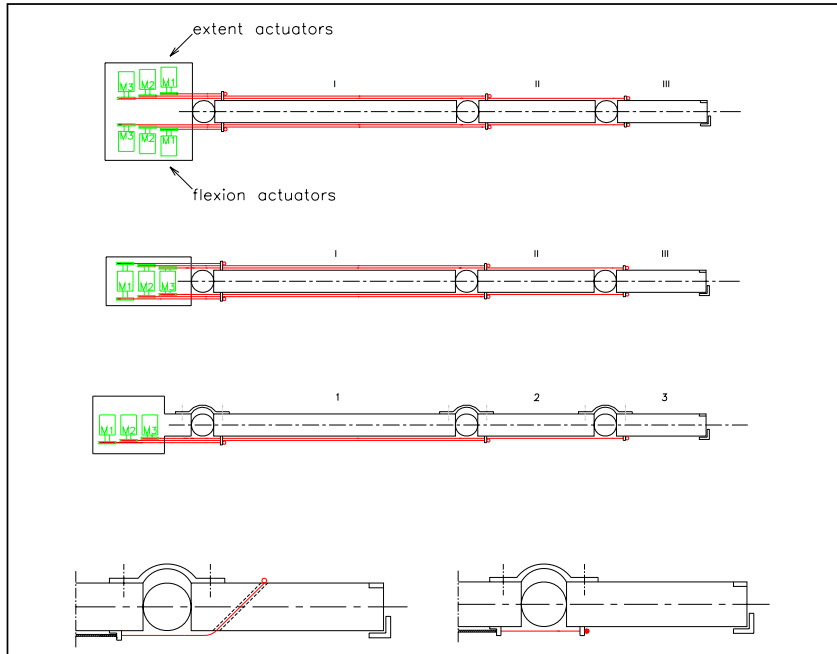


Fig. 5. Three potential motor configurations. TOP PANEL. Extent and flexion actuators (i.e. 6 motors for 3 independent phalanges). This configuration is expensive in terms of number of actuators and of mechanical complexity - adopted by the MIT Hand, i.e. 266 pulleys to drive all the tendons network - Jacobsen et al., 1986. MIDDLE PANEL. Extent & flexion are driven by the same motor for each phalanx. In this configuration the tendon excursions (flexion and extension) must be the same - this configuration is adopted, for example, by the UB Hand II - Melchiorri & Vassura, 1995). BOTTOM PANEL. Flexion is actively realized by motors, while extent follows by the spring responses. BELOW. Details over the tendon attachment on the third DIP phalanx.

Different motor configurations were analyzed to establish if flexion and extension should be activated by 3 (or 6) independent motors and/or by passive springs placed around each articulations (Fig. 5). Tendons attached to the finger's skeleton are also discussed, according to bio-mimetic analogies (Fig. 5, bottom panel - see also Kapandji, 1970).

The final configuration is based on three artificial muscles - Wittenstein Motion Control GmbH, link {1} - placed in the forearm that move the three phalanges independently. The final design of two (of the 3 phalanges) is shown in Fig. 6 where the dimensions of the mechanical phalanges are similar anatomically to the human middle finger.

It is important to note that the "imitation" approach has recently stimulated further discussion (Schaal, 1999) and it has been adopted at the A.I. Laboratory of MIT {2} (development of a human-like finger system), at the LIRA-Lab - {3} (the Babybot Project)

and at the Humanoid Robotic Group Research at MIT - {4} (see also the Biologically Inspired Robotics Laboratory - {5}).

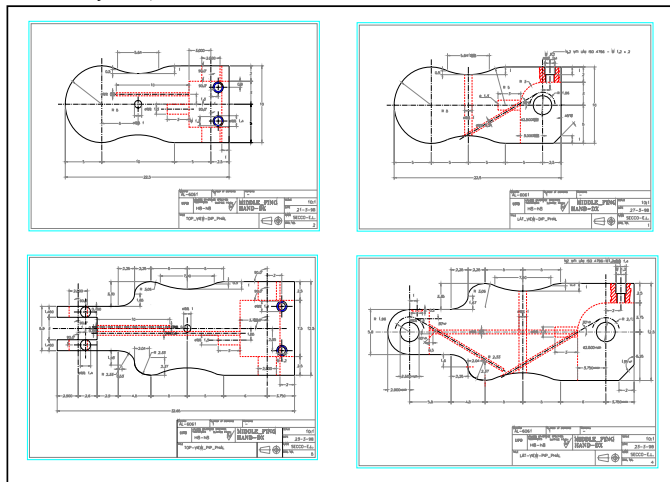


Fig. 6. Top and lateral view (respectively on the LEFT and on the RIGHT side) of the DIP and PIP phalanges (respectively on the TOP and on the BOTTOM). Quotes are expressed in [mm]; the mechanical components are dimensionally similar to the human phalanges morphology - from Secco, 1997.

3. Bio-Mimetic Motion Planning

We propose a method, based on both physiologic and engineering considerations, for the motion planning of a prosthetic finger. In particular, we exploit a minimum jerk approach to define the trajectory in Cartesian space. Then, cubic splines are adopted in the joint space. The redundancy problem arising from the presence of three links is solved by assuming that there is a constant ratio between the second and the third joint motion. The value of the proportional constant is determined by minimizing the maximum jerk in the joint space. It is found that this constant value can be suboptimally, but effectively, set to one for all the movements. This approach guarantees a natural movement of the finger as well as reduced vibrations in the mechanical structure and increased control performance.

3.1 Introduction

In the past few years there has been significant development of prosthetic hand devices and haptic perception. Many laboratories have produced multi *dof* artificial hands for robotic and/or prosthetic applications (see, e.g., Jacobsen et al., 1986; Rosheim, 1994; Melchiorri & Vassura, 1995). In this context, after reproducing the morphology, it is also of interest to reproduce the natural movement of the fingers in order to perform different tasks, such as exploring an unknown object with an artificial hand (for teleoperation use) or reaching and handling an object with prosthetic devices (for the support of handicapped patients). However, this has to be done while also taking into account the design constraints of the mechanical structure and of the control architecture.

From the physiological side, many studies have been published regarding the finger (Harris & Rutledge, 1972; Buchner et al., 1988; Lee & Rim, 1990; Hoff & Arbib, 1993; Hahn et al, 1995; Brook et al., 1995), arm (Massone & Bizzi, 1989; Okadome & Honda, 1999), and human movements in general (Nelson, 1983; Plamondon, 1995ab), with both kinematic and dynamic approaches. In general, they conclude that natural movements are planned and executed by following optimization principles. In particular, we highlight the contribution of Flash & Hogan (1985) who show that the upper limb movements in the proximal space are executed in order to minimize the square magnitude of the jerk (the derivative of the acceleration function) of the extremity in the Cartesian space over the entire movement. Following this result, Laczko et al. (2000) investigated, in multi-joint kinematic chains, the relative contribution of the velocities, accelerations, and jerks in the individual joints to the total endpoint jerk; they concluded that the term related to the individual joint jerk dominates over the others. From a robotic viewpoint, the problem of determining the motion of a multi joint finger has to face the redundancy of the system, i.e., there are an infinite number of joint configurations for a unique Cartesian fingertip position. In other words, the presence of the third joint/phalanx implies that we have three *dof*, namely one more than that necessary to address a motion in a two-dimensional plane. In general, the additional *dof* is exploited in order to minimize some objective function.

It is also recognized that the minimization of the joint jerk provides benefits to the mechanical structure of a robot manipulator, reducing the presence of vibrations and the joint wear and therefore increasing the robot life-span (Craig, 1989) as well as permitting the increase of trajectory control performances (Kyriakopoulos & Saridis, 1988; Secco & Magenes, 2002ab). In this context, different approaches to minimize the joint jerk in the trajectory planning of robot manipulators have been proposed in the literature (Simon, 1993; Piazzi & Visoli, 2000)

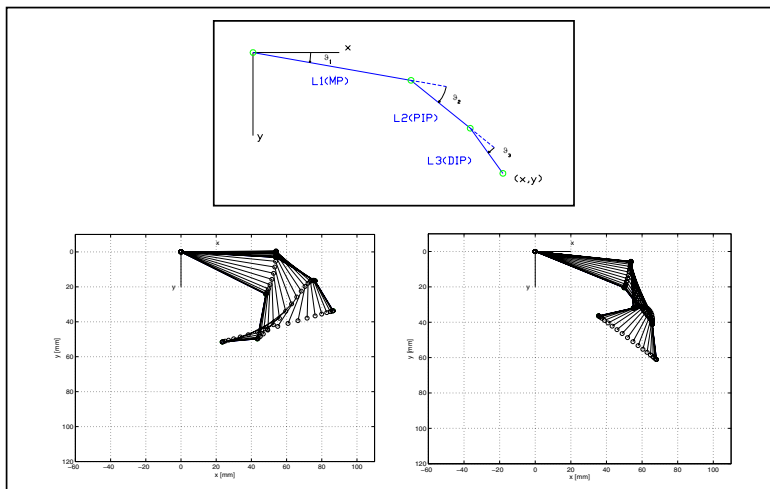


Fig. 7. TOP PANEL. Schematic representation of the prosthetic finger. BOTTOM PANEL. Evolution of 2 representative simulations of minimum jerk movements.

However, when the coordinated movement of a prosthetic arm, consisting of a multi-link/multi-joint kinematic chain, has to be programmed and controlled in order to reach a

determined position in the proximal space with the end-effector (the fingertip) for grasping, touching, or exploring an object, it seems useful to maintain a smooth motion of the fingertip along the straight line trajectory connecting its initial and final positions. This implies a smooth control of the fingertip in the Cartesian space, with null acceleration at the beginning and at the end of the motion. This should prevent accidental mechanical shocks of the fingertip on the object, due to possible minute errors in the joint actuators. Thus, in addition to physiological requirements, this represents an important mechanical feature, as a low impact is guaranteed in the fingertip approach to the object's surface.

Taking into account all these considerations, we propose a new approach for the motion planning of a prosthetic finger, based on the minimum-jerk principle. Basically, it consists of defining linear movements of the tip in the Cartesian space, defined in such a way to minimize the Cartesian jerk. Then, the inverse kinematic problem is solved by using cubic splines to interpolate the trajectory between the two known knots and by fixing a constant ratio between the second and the third joint angle. The value of the constant ratio is found by minimizing the maximum jerk of the three joints. The obtained results are consistent with physiological evaluations.

3.2 The Minimum Jerk Principle

3.2.1 Motion Planning in the Cartesian Space

We consider a human middle finger, shown in Fig. 7, where $L1 = 54$ mm, $L2 = 26$ mm, and $L3 = 20$ mm are the lengths of the phalanges¹ and ϑ_1 , ϑ_2 , and ϑ_3 are the Metacarpophalangeal (MP) joint, the Proximal-Inter-Phalangeal (PIP) joint, and the Distal-Inter-Phalangeal joint (DIP), respectively. From physiological considerations (Kapandji, 1970) it has been assumed that

$$-30^\circ \leq \vartheta_1 \leq 90^\circ, \quad 0^\circ \leq \vartheta_i \leq 90^\circ, \quad i = 2, 3 \quad (1)$$

Only straight movements in the X-Y plane are considered.

By applying the mathematical model developed by Flash and Hogan (1985) to the finger, it results that a natural movement from position (x_0, y_0) to position (x_f, y_f) starting at time $t_0 = 0$ and ending at time t_f has to minimize the following function, (the time integral of the square of the magnitude of jerk):

$$C = \frac{1}{2} \int_0^{t_f} \left[\left(\frac{d^3x}{dt^3} \right)^2 + \left(\frac{d^3y}{dt^3} \right)^2 \right] dt \quad (2)$$

where x and y are the time-varying coordinates of the fingertip position. By solving the optimization problem, we see that the trajectory of the fingertip in the Cartesian space is uniquely determined as follows:

$$\begin{aligned} x(t) &= x_0 + (x_f - x_0) \left[-6 \left(\frac{t}{t_f} \right)^5 + 15 \left(\frac{t}{t_f} \right)^4 - 10 \left(\frac{t}{t_f} \right)^3 \right] \\ y(t) &= y_0 + (y_f - y_0) \left[-6 \left(\frac{t}{t_f} \right)^5 + 15 \left(\frac{t}{t_f} \right)^4 - 10 \left(\frac{t}{t_f} \right)^3 \right] \end{aligned} \quad (3)$$

¹ We applied this study also with the average American female middle finger ($L1 = 42.11$ mm, $L2 = 20.27$ mm, and $L3 = 15.60$ mm) and with the average American male middle finger ($L1 = 46.22$ mm, $L2 = 22.26$ mm, and $L3 = 17.12$ mm - Garrett, 1971), obtaining the same results for all the three fingers. The length of the phalanges is not a critical issue as well (see details in Secco et al., 2002).

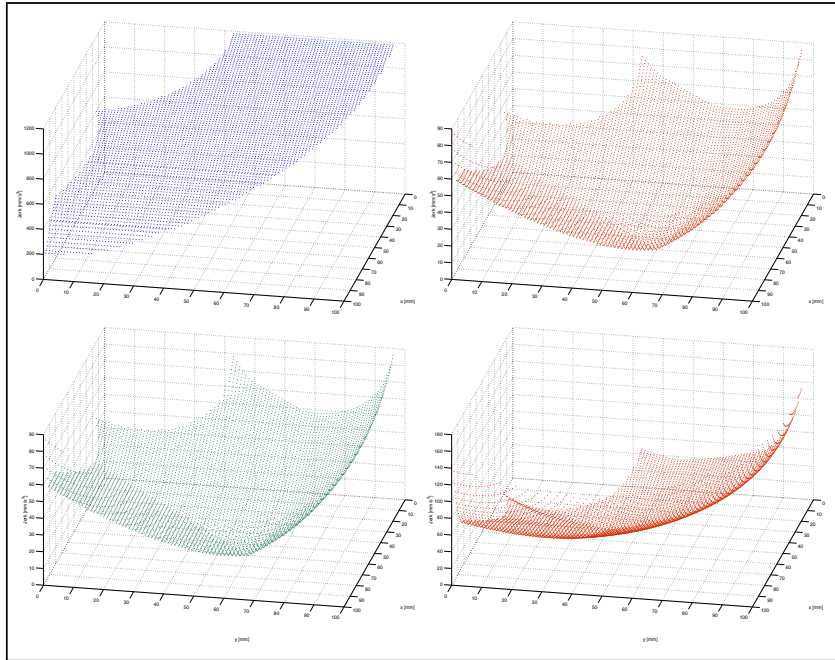


Fig. 8. Values of the maximum jerk (VMJ) obtained by using the value $K=1$ for motions starting from (100,0) - TOP LEFT PANEL - and from (60,60) - BOTTOM LEFT PANEL. VMJ by using the optimal value of K for motions starting from (60,60) - TOP RIGHT PANEL - and from (20,60) - BOTTOM RIGHT PANEL.

3.2.2 Motion Planning in the Joint Space

Once the movement of the fingertip in the Cartesian space has been determined, the inverse kinematics problem has to be solved in order to calculate the motion law in the joint space, which is then applied to the robotic finger by using appropriate actuators. A practical solution, often adopted in industrial environments, is to select a sufficient number m of equally spaced knots along the Cartesian trajectory.

For simplicity, we select the knots d_1, \dots, d_m , at time intervals equal to each other. Then, the corresponding joint configuration has to be determined by applying the inverse kinematics. In this context, the redundancy problem has to be faced. Namely, the presence of three joints causes that an infinite number of joint configurations solve the inverse kinematics problem. One of the simplest methods to effectively tackle the redundancy problem, consists of maintaining the angle of the third phalanx proportional to that of the second phalanx, namely,

$$\vartheta_3 = K\vartheta_2, K \geq 0 \quad (4)$$

Note that this assumption seems to be reasonable from a physiological point of view, since such proportionality - eq. (4) - drives the natural movement of the human finger (see, e.g.,

Harris & Rutledge, 1972; Hahn et al., 1995; Secco, 2001). Then, for each joint, cubic splines (Craig, 1989; Lin et al., 1983) are adopted to interpolate the resulting displacements for each joint. The salient feature of the cubic splines is that they assure the continuity of the velocity and acceleration functions and, being of low order, they prevent large overshoots (see details in Secco et al., 2002).

3.2.3 Optimization Problem

In the framework proposed in the previous subsections, the resulting trajectory in the joint space depends on the design parameter K . An explicit solution of the inverse kinematic problem that depends on K cannot be computed because of its high complexity. Therefore, the value of K has to be fixed before the inverse kinematics and the cubic splines approach are applied. In this context, the inverse kinematic problem can be solved by applying a standard Newton–Raphson algorithm (Ortega & Rheiboldt, 1970). Thus, once a value of K has been selected, taking into account the physical constraints on the joint angles (1), the inverse kinematic problem can be uniquely solved.

An appropriate method to find the value of K is to minimize the maximum jerk over the three joints. In other words, we have to solve the following constrained min max optimization problem:

$$\min_{k \geq 0} \max \{ |j_{ki}| : i = 1, \dots, n; k = 1, 2, 3 \} \quad (5)$$

subject to

$$-30^\circ \leq \vartheta_1 \leq 90^\circ, \quad 0^\circ \leq \vartheta_i \leq 90^\circ, \quad i = 2, 3.$$

From a practical point of view, an upper bound for the optimal value of K can be easily found by taking into account again physiological consideration (Kapandji, 1970). A conservative upper bound $K^+ = 1.5$ has been selected, while, as a lower bound, the value $K^- = 0$ has been retained. In order to find the optimal value K^* that minimizes (5), a tight gridding (with step equal to 0.01) over the interval $[K^-, K^+]$ has been performed, evaluating the maximum jerk for each value of K and then selecting the optimal one (see details in Secco et al., 2002).

3.3 Results

The algorithm presented in Section 2.3 has been applied to a large number of movements, with different motion times (Secco et al., 2001). Here, for the sake of clarity, we focus on some significant results that illustrate the conclusions we draw. In particular, we consider straight movements performed in an interval time of 1 s (i.e., $t_f = 1$ s) with $m = 20$. As example, different configurations of the finger for particular movements are reported in Fig. 7.

First, we select all movements starting in (100,0) - Fig. 7 - and ending in the points of the workspace. In all cases we obtain that the optimal jerk corresponds to a value of K close to 1. The resulting optimal jerk j^* for each movement is shown in Fig. 8 - top left panel. Slightly different results are obtained for different starting points. Namely, the solution of the optimization problem does not always yield to the value $K = 1$. In any case, it is of interest to compare the values of the jerk j^* achieved for the optimal values of K and the values of the jerk achieved by fixing $K = 1$. They have been reported in Fig. 8 - top right panel and Fig. 8 - bottom left panel, respectively, for movements starting in (60,60) and in Fig. 8 - bottom right panel and Fig. 9 - left panel, respectively, for movements starting in (20,60).

It appears that the difference between the values of the maximum jerk achieved by selecting the optimal value of K and $K = 1$ is not very significant. This is confirmed by the fact that the resulting optimal value of K is indeed $K = 1$ for many movements.

The reader can appreciate what is affirmed through the illustrative example of Fig. 9 – right panel, where the final points of the movements (for the case $x_0 = 20$ and $y_0 = 60$) for which the optimal K is equal to one are indicated. For a better evaluation of the results from an analytical point of view, the mean value of the maximum jerk and its standard deviation for the considered movements are reported in Table 1.

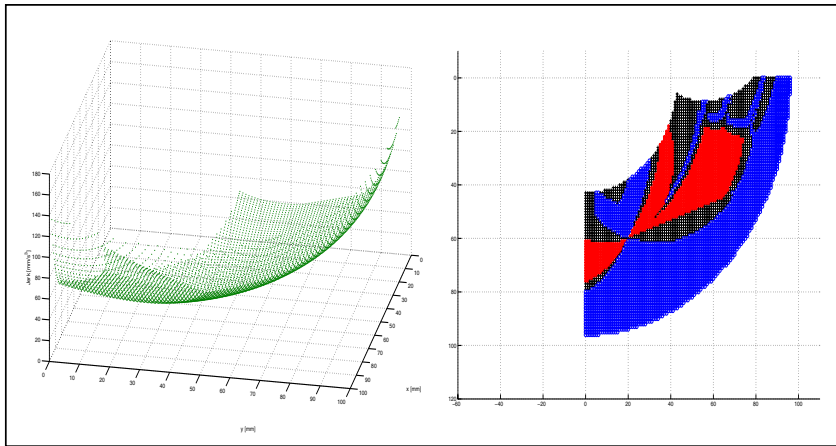


Fig. 9. LEFT PANEL. Values of the maximum jerk obtained by using the value $K=1$ for motions starting from $(20,60)$. RIGHT PANEL. Final positions of motions starting from $(20,60)$ for which $K=1$ (indicated with blue “+”). Red dots (“o”) represents the $K=1.5$ class, black cross (“+”) the uncertainty zone. The percentage range permitted to belong to each class is 10%.

3.4 Discussion

The obtained results show that a suitable choice to solve the redundancy problem in the motion planning of the prosthetic finger consists of choosing a constant ratio equal to one between the PIP and the DIP joints. Although the proposed solution is theoretically suboptimal, from a practical point of view it is easy to implement and preserves the minimization of the maximum jerk in the joint space. In addition, the value $K = 1$ is appropriate also from physiological considerations, as it reflects the behavior of a real human finger (Harris & Rutledge, 1972; Secco et al., 2005; Sumbre et al., 2005).

It also worthnoting that for $K = 1$ the inverse kinematic problem can be solved analytically (details in Secco et al., 2002).

Hence, the overall proposed methodology satisfies both physiological and engineering requirements as the planned movement reflects that of a human finger, both in Cartesian and in the joint space, and it ensures the prevention of vibrations and the increasing of the control performances by keeping the maximum jerk of the joints at a low level.

Starting Point	Optimal K		K=1	
	mean	std	mean	std
(100,0)	812.8	238.3	812.8	238.3
(60,60)	33.4	15.2	33.8	15.4
(20,60)	40.8	27.8	41.5	27.9

Table 1. Mean value and standard deviation of the maximum jerk for different motions with the optimal value of K and with K=1.

3.5 Conclusions

This strategy ensures both a smooth approach to the object along a straight line trajectory (with a low shock of the fingertip when touching the object) and low jerk at the joint level, as required by physiological and mechanical considerations. The redundancy robotic problem is solved by selecting a constant ratio (equal to one) between the PIP and the DIP joint, in order to minimize the maximum of the jerk functions of the three joints. This solution is suitable to be applied in a practical context since it satisfies engineering requirements (easiness of implementation, reduction of vibrations, increasing of the control performances, etc.) and at the same time it allows a physiological smooth natural movement.

4. Bio-Mimetic Control: Teaching a Robot With Human Natural Movements

After reproducing the morphology and the kinematics, we face a human inspired sensorimotor approach to robotic systems, to solve the problem of designing adaptive and learning robots for widely versatile tasks. Hence we present a *nn* controller for driving the 3 *dof* finger to desired positions in space. The controller is taught with human movements of the corresponding finger. At the end of the training process, it is able to closely imitate the physiological control and the motion planning strategy of the human being. Generalization properties are shown after the training process (that is the capacity to move the device in different directions and places never seen during the teaching phase). This approach seems promising for controlling artificial prosthetic and robotic upper limbs in general.

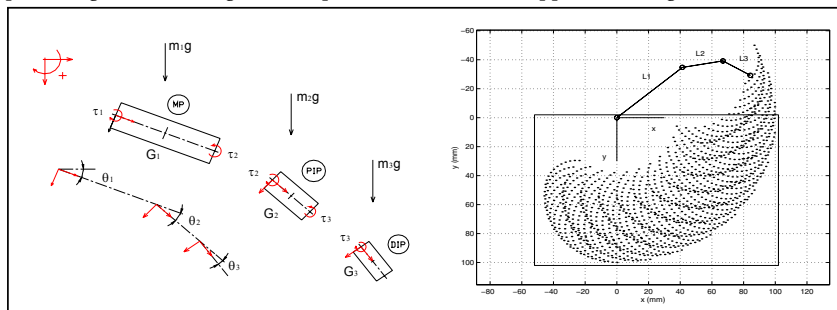


Fig. 10. LEFT PANEL. Conventions of the signs, the external forces and the motor torques applied to the three phalanges - τ_1 , τ_2 and τ_3 are the MP, PIP and DIP Phalanx torques. G_1 , G_2 and G_3 are the 3 barycenters; m_1 , m_2 , m_3 , θ_1 , θ_2 , θ_3 are masses and angular displacements respectively - g represents the gravitational contribute. RIGHT PANEL. Schematized representation of the geometrical model adopted for the middle finger in the vertical plane: the rectangular box represents the workspace domain of the *nn* controller.

4.1 Introduction

The problem of designing and producing a humanoid robot, able to interact intelligently with the environment, has recently been inspired by the behavior-based approach that avoids high level cognitive representation and relies upon a tight connection between sensory and motor systems (Brooks & Stein, 1994). Among the evident advantages that could derive from a human inspired sensorimotor approach to robotic systems are the reproduction of the outstanding fault tolerance, the adaptability to changed environments, the learning ability and the inherent versatility of human beings.

In this last paragraph we present a *nn* controller for driving the 3 *dof* finger to a desired position in its proximal space. The controller is taught with human movements of the corresponding finger and, at the end of the training process, it is able to closely imitate the physiological control and the motion planning strategy in human beings.

Both for robotic and prosthetic applications, such control system driving multi *dof* artificial limbs in a smooth natural manner could represent a noticeable improvement. Moreover it is compulsory that the natural movement, instead of being controlled by overwhelming equations (see par. 2), can be executed in a straightforward manner and can be learned by examples from human experimental measurements. Thus, a possible methodological tool capable of achieving the desired specifications is represented by artificial *nn*. They allow us to setup the control system bypassing the computation of the explicit solution of the inverse dynamics problem.

The control law is obtained by optimizing the weights of the *nn* through a *supervised training procedure* which utilizes a set of input-desired output pairs, taken from natural movements measurements.

4.2 Methods

The robotic finger is mechanically simplified as consisting of three links (the MP, PIP & DIP phalanges) and three planar joints (the 3 articulations), on which three torques can be applied by means of small torque motors. A mechanical model of the artificial finger, a kinematic model of its movement and a dynamic model are developed. A similar approach could be easily readapted to the analysis and control of the upper limb too.

4.2.1 The Plant and the Controller

The *mechanical properties* of the finger have been previously characterized by the mass, the friction and the forces acting on each phalanx (Secco & Mageses, 2002a). Each joint torque motor is provided by a rotational encoder giving the angular position and velocity of each phalanx with respect to the preceding one. The finger is acting on the vertical plane under the gravity action. Specifically we consider a planar model of the middle finger consisting of 3 rods (the 3 phalanges) and 3 hinges - the MP, PIP and DIP joints. The choice of neglecting lateral movement is supported both by the need of simplifying the problem and by the design constraints of a prosthetic/artificial middle finger. The finger configuration is defined by the three angles θ_1 , θ_2 and θ_3 as illustrated in Fig. 10 - according to the robotic Denavit-Hartenberg notation. The anatomical constraints and the dynamic phalanx properties are selected according to the anatomical behavior of the homologous human finger. These conditions univocally determine the workspace of the artificial fingertip (x,y) - L_1 , L_2 and L_3 are defined as the MP, PIP and DIP phalanx lengths, θ_1 , θ_2 and θ_3 the respective relative angular dispositions.

Whereas we need to check the behavior of the network by supposing a virtual movement and not using the real acquired movements, the adopted *kinematic model* follows a simple approach; we fix a smooth natural movement mathematically illustrated by the two conditions expressed in Eq. (2) & (4) with $K = 1$. The first equation comes from the minimum jerk theory which we are extending to the finger, while the second hypothesis stems from the necessity to support the first, because of the three *dof* redundancy (see the details in par. 2). By means of the imposed conditions we can describe a rectilinear and smooth movement of the fingertip (as reported on the background of Fig. 11 - top panel), providing an unimodal bell speed profile.

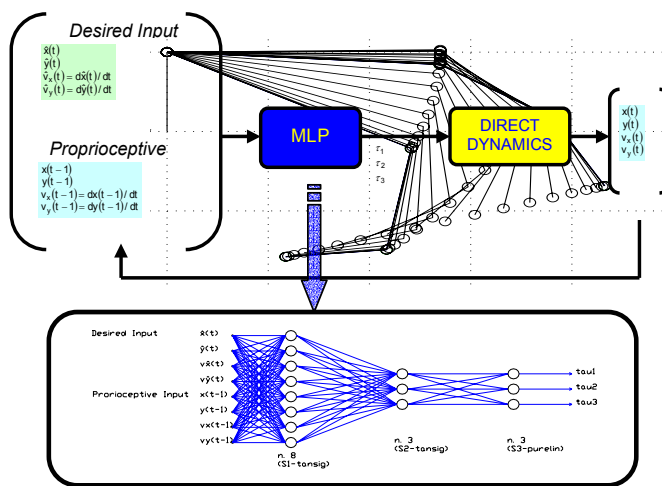


Fig. 11. TOP PANEL. The closed loop control system: the controller (MLP) is fed back by the proprioception of the muscle spindles. A direct dynamic block determines the effective movement due to the applied torques. BOTTOM PANEL. The Multi Layer Perceptron (MLP) nn: the structure of the I/O training vector, the activation functions and the number of neurons for each layer. In the configuration of the figure we have 8, 3 and 3 neurons on the 1st, 2nd, and 3rd layer respectively. Activation functions are tansig, tansig and purelin respectively. Because of the type of activation functions, the three outputs are normalized between [-1,1]. Others structures were tested applying different architectures of the network with 3 and 4 layers and different numbers of neurons. The present configuration and the training set dimension are the results of an optimized compromise (see for details Secco & Magenes, 2002a; see also Haykin, 1999).

The *dynamics* is governed by the kinematics. The kinematics is the real movement or the ideal movement that we have described above. In order to set up a dynamic model we define the properties of the artificial finger by establishing that a) the finger is moving on a vertical plane and it is subject to gravity, b) the phalanges are treated as homogeneous rods, c) the torques are provided by the motors positioned on the joints. Following Lagrange's approach we could write out the expressions of the three motor torques (Secco and Magenes, 2002ab).

Finally, a Multi Layers Perceptron (MLP) nn is setup. The input of the controller are the desired position and velocity of the fingertip (*sensorial input*) and the actual position and velocity (*proprioceptive input*). The output produces the joint torques. The MLP is fed back with the actual positions and velocities by closing the sensorimotor loop through the dynamics of the finger: the real movement due to the applied torques is sent back to update the proprioceptive input. The training performance is the minimization of the SSE (Sum Squared Error) of the output normalized torque. The three generalized forces are previously normalized between $[-1,1]$ - three factors of normalization are applied (one for each phalanx) - to be compatible with the domain of the nn activation functions. The resulting performance function is:

$$SSE = \sum_{ijk} (\tau_{ijk} - \hat{\tau}_{ijk})^2 \quad (6)$$

where:

τ_{ijk} = simulated normalized value of the torque from the nn output

$\hat{\tau}_{ijk}$ = desired normalized value of the torque from the dynamic equations

$i=1, 2, 3$ = number of phalanges

$j=1, 2, \dots, 7$ = number of movements

$k=1, 2, \dots, 23, 24$ = number of steps/temporal passes for each movement

The adopted network has 8-5-3 neurons on the first, hidden and output layer respectively (Fig. 11 - bottom panel). Thanks to the dynamic model we are able to compute the necessary torque for each required movement and to train the controller. The learning procedure is *back-propagation with adaptive learning rate and momentum term in batch mode*². Seven human movements are applied with the respective computed torque vectors for the learning phase

4.2.2 Teaching the movement

We performed some experimental measurements on the human movements of flexion and extension of the human finger. More precisely, we prepared an experimental setup to acquire the free flex-extension movements of the right index finger of human subject.

Before describing the apparatus, the measures and the results, we want to stress why it was important to perform this experimental phase.

- First, the obtained data could be used to train the controller with actual natural movements. Through this procedure we could bypass the identification of the natural movement model. Moreover, we could get closer to the realization of a robotic/prosthetic device, which is trained by the observed movements (Da Cunha et al., 2000).
- Second, the analysis of the data would have allowed us to improve/verify the kinematic model we apply to prepare the virtual movements (that is the kinematic model - Secco et al., 2002).

² Details on the structure's optimization and on the training procedure are reachable on Secco & Magenes, 2002a.

- Third, we could control the effective torque patterns with respect to the training ones. We mean that we could apply the acquired movements to the artificial phalanges, deducing the consequent 'real' motor torques to move in this natural manner.

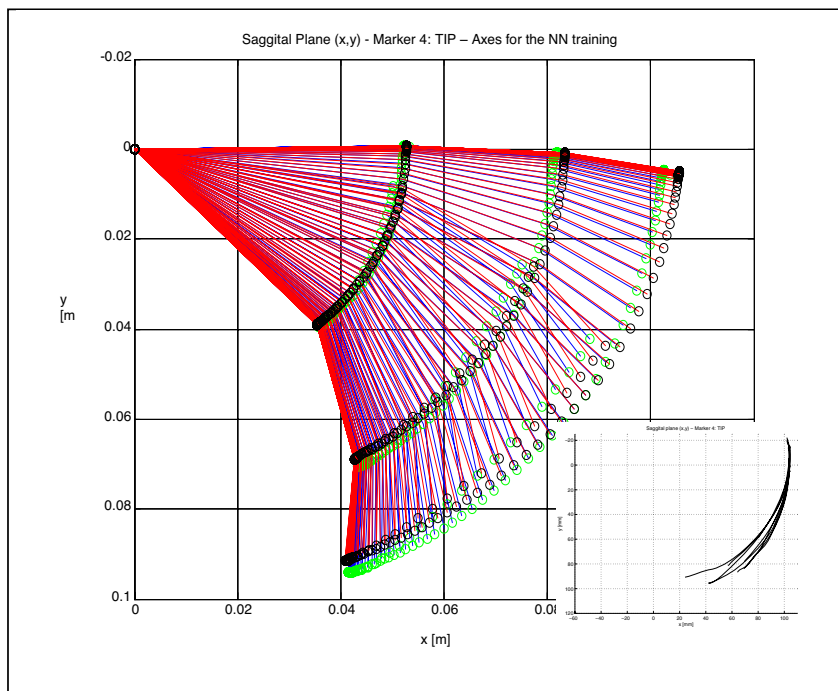


Fig. 12. MAIN PANEL. Finger path reconstruction during a partial flexion-partial extension movement and Fingertip (TIP) multiple path acquisitions - RIGHT BOTTOM PANEL.

A passive infrared markers system was used (Elite® System). We registered the movements by two cameras at a 100 Hz frequency. After the calibration and the estimation of the position error (less than 1%), we applied 4 markers on the lateral side of the right index finger of 11 healthy volunteers. We acquired the movement on the saggital plane (we call it the xy plane). We fixed the transducers to the MP articulation (Marker n. 1), PIP articulation (Marker n. 2) and DIP (Marker n. 3) one. A fourth marker describing the Fingertip movement was fixed on a thimble that the volunteer was wearing during the acquisition. The palm was placed on a support, while the finger freely moved into the calibrated volume in front of the two cameras. The four markers were always visible from both the cameras. Approximately the finger moved on the xy plane (apart from the movement of the palm on the hand support and the disturbances due to the movement of laterality - not inhibited).

At the beginning the support was placed in such a manner that the palm distance from the flat was 50 mm. In a second phase the support was rotated and the distance between the palm and the flat was increased to 66 mm. For each phase and each person we asked to move the finger from a unique stretch limb configuration to a final fingertip position on the flat. The finger passed through the lighted band of a photocell: this was connected with a trigger that started the acquisition. Then the finger moved to one of the final possible positions on the flat: there were five possible final positions, equally spaced of 20 mm on a horizontal straight line. In the best case, the total number of acquisitions for each of the 11 volunteers was 10 (5 movements with the support on the 'position 50 mm' and other 5 movements in the 'position 66 mm').

We can describe a single training movement as a sequence of multiple micro-movements. The controller structure (that is the *mm* configuration and the training set dimension) was optimized for a 200 micro-movements set-up (Secco & Magenes, 2002a; Haykin, 1999). Moreover we obtained not more than 8 movements of the same subject that were sufficiently extended in time. Thus we caught 7 of them for the training phase and one for the validation one. Considering the optimal number of micro-movements was 200 and the number of entire available movements was 7, then we divided each available acquisition in 24 successive positions of the fingertip at equal temporal intervals $\Delta t = 30$ ms, from an initial fingertip position (x_i, y_i) at time $t_i = 0$ s to a final one (x_f, y_f) at time $t_f = 30 \text{ ms} * 24 = 0.72$ s. In this way 24 *steps/μ-movements* were used for each movement. The marker positions were previously filtered (low pass filter with a cut-off frequency of 10 Hz). The markers positions were then corrected by imposing a constant length of the phalanges during the motion. The Cartesian fingertip velocity was computed, filtered with a smooth process and used to define the beginning of the movement. Then the resulted vector was resampled with a 30 ms period (Vicini, 2003). Then we computed the angular position, speed and acceleration of the three links to calculate the three torques for executing each μ -movement. The torques were normalized and the entire set of vectors - containing the 7 movements - was setup for the training phase of the controller.. Finally a 200 thousand epochs training history performs the final error $SSE = 0.8$.

4.3 Results

After training the MLP with the 7 real movements, we tested the same movements as input and we observed the normalized outputs. Here we show the results through graph bars and time patterns.

4.3.1 Open Loop Control (i.e. no proprioception)

For each movement, the *SSE* error of each temporal step is represented (24 temporal steps imply 24 torque values τ , where $\tau(t)$ was applied for a moving finger from time $t-1$ to time t). The sum of the *SSE* of the three torque values (MP, PIP and DIP) for each pass is shown in the relevant bar (Fig. 13 - top left panel). The results are referring to the normalized torques. It is not manifested what happens to each phalanx torque in this representation. We need to look at each single phalanx torque computed by the *mm* in the simulation phase, and to compare it with the expected value. Fig. 13 - top right panel refers to the temporal torque patterns of the training movement n. 5: *theoretical torques* are compared with *simulation values*

obtained by the trained MLP. All the remaining 6 training movements show an equivalent behavior.

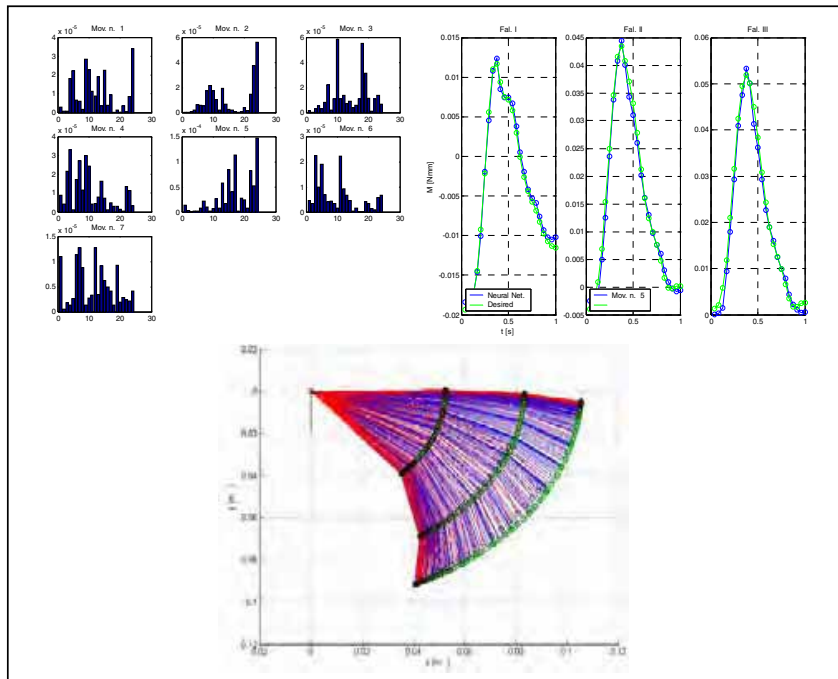


Fig. 13. TOP LEFT PANEL. The SSE of the normalized torques for each of the 7 training movements (step and $[N \cdot m / N \cdot m]^2$). Each bar represents the total SSE of the three torque values for each single step of the examined movements. TOP RIGHT PANEL. Course of the de-normalized torques of the training movement n. 5/7, '-' green solid lines are the desired/theoretical value, '-' blue solid lines represent nn simulation values. BOTTOM PANEL. We apply the integrator to the three patterns of motor torques. The torques are given as input to the integrator (the Direct Dynamic block of Fig. 11) as constant values on each interval of 30 ms for the entire execution time (0.72 s): the obtained final integrated movement, '-' blue lines, is compared with the real acquired one, '-' red lines.

It is possible to analyze and to represent the real/de-normalized torque values too, but we have to consider that:

- real de-normalized values do not give information about the success of the training phase: differences of SSE could be caused by different orders of magnitude of the real de-normalized phalanx torques values involved;
- de-normalized values trend is an homothetic transformation of the normalized ones because of the three applied factors of normalization (one for each phalanx).

We applied the simulated and de-normalized torque to the Direct Dynamic block. The results give an answer about the effective precision of the *nn* in executing the movement. We gave the generalization movement as input of this new configuration. We asked to the *nn* to execute the 8th never seen before movement. The graphical result is reported on Fig. 13 - bottom panel. It is compared with the real 8th acquired movement. The two final fingertip positions are not coincident: the distance between the two positions (less than 10% with respect to the entire fingertip movement) comes from the cumulative error of the three simulated torques integration.

The final result depends on the integration of the highly non-linear dynamic equations. Every final trajectory of the training movements is well reproduced. Also the *nn* seems to be able to simulate never seen movement during the training phase. Therefore we could conclude that all the training movements are re-executed by the MLP in a very satisfying manner and that the *nn* can "implicitly learn" the dynamics of the system.

4.3.2 Closed Loop Control (i.e. proprioceptive feedback)

The dynamic control system was initially implemented as a sampled open-loop chain (feed-forward configuration). We did not closed the loop by up-dating the proprioceptive input during the motion. Actually, the feedback configuration of the controller represents the real natural condition. Thus we decided to realize the feedback procedure: we required the execution of a movement., then the movement was prepared and converted into a sequence of 24 different desired inputs and proprioceptive ones. For each step of motion from time $t-1$ to time t (as usual, $t_f = 0.72$ s, $t_0 = 0$ s, $\Delta t = 30$ ms) a couple of inputs (desired + proprioceptive) was given; due to the input, the MLP computed the torques at time $t-1$ and time t . Interpolating the three values between $t-1$ and t , the three torques patterns were finally integrated. The new final fingertip position at time t and its speed were fed back to update the proprioceptive input. This was the new state at time $t-1$ for the next step of motion from $t-1$ to t . The procedure was repeated until the 24th step.

The same generalized movement was simulated by closing the loop (feedback configuration). The final trajectory is reported in Fig. 14 - top panel.

Finally, we wanted to verify the *nn* ability of executing whatever movement by maintaining the same dynamics. We stressed the system by asking to execute a virtual rectilinear movement (as reported on Fig. 11 - top panel) by following the above defined kinematic model. We tested this in both the feed-forward and feed-back configuration. Graphical results are reported on Fig. 14 - bottom panel: we see that the feedback configuration (right side of the mentioned figure) is clearly the best approach to imitate the biological control. We notice that both the curvilinear and the rectilinear movements are flexions. Extensions were tested with bad results.

However, even if the controller is not able to execute reverse (or not-forward) movements, it seems able to generalize quite well (also in terms of initial conditions). Considering that we could prepare more complete training sets (in terms of forward and reverse movements) with a consequent bigger capacity of moving in each direction, these results are very promising. Moreover, the application of such a system to the control of a device having a simpler dynamic - as a 2 *dof* arm - could be even more efficient.

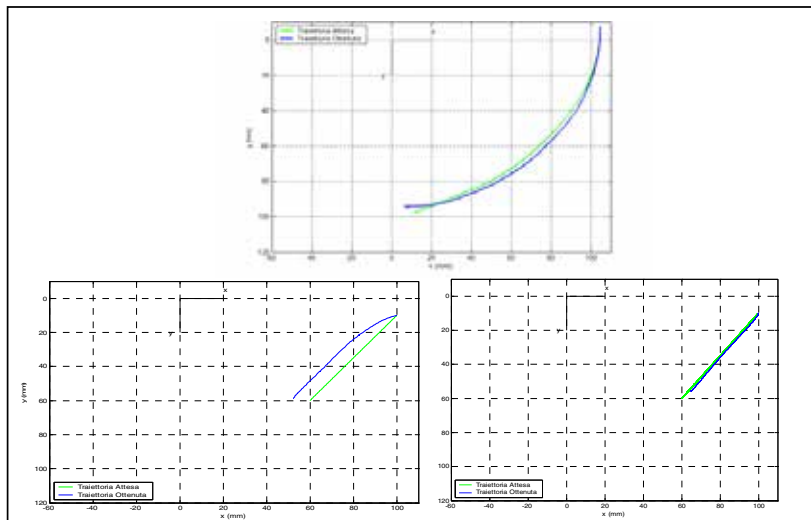


Fig. 14. TOP PANEL. The final fingertip real movement, ‘-’ green line, is compared with the integrated one, ‘-’ blue line, by closing the loop control chain. The ‘in time’ up-dating of the proprioceptive information is more visible in the final configuration. The trajectories are deduced by the integration of the three ‘real’ and simulated torques. BOTTOM PANEL. Fingertip ‘real’, ‘-’ green line, and simulated, ‘-’ blue line, trajectories obtained by the feed-forward configuration controller (LEFT) and by the feedback configuration (RIGHT). The rectilinear movement is prepared by taking into account the kinematic model described in Par. 2, that is an extension of the minimum jerk theory (Flash & Hogan, 1985) – details on (Secco et al., 2002).

4.4 Discussion

It has to be underlined that an optimization procedure must be carried out on the training set depending on the network structure. We mean that the next step could be to improve the number of *Macro-Movements* in the set of training examples, without reducing the number of μ -*Movements*. This operation will require a re-modelling of the m , due to the above optimization procedure (Haykin, 1999; Secco & Magenes, 2002a).

A further step would be to define a performance parameter that is representative of the graphical results and of the final fingertip position. For instance the Euclidean distance between the desired final configuration and the obtained one could be a good proposal. Furthermore, it would be useful to define a performance index before the integration. In fact, despite the compensative and positive effect of the feedback, the torques computation of the m is controlled by the *SSE*, while the integration error is not ‘controlled’ yet. An intelligent solution could be to evaluate the area extension between the desired and obtained torques pattern. This is highly correlated with the integration. Finally, because of this correlation, we note (without the feedback) an increasing error on the fingertip position during the final part of the trajectory. An interesting development could be to train the

network in function of a new Performance Parameter that depends on the integrated area between the MLP computed torques and the desired ones. However it has to be taken into account that the TIP reported trajectories represent the result of all the three torque integration, that is the final behaviour of a 3 *dof* mechanical chain. Due to the good results we would apply the same acquisition-learning-moving plant to the human arm, that could be initially treated as a more simple 2 *dof* planar limb.

The contents of such approach seem to have a high potential. By teaching the controller with few real movements, the system is able to 'generalize' and to learn the dynamics implicitly. We like imaging robots and/or prosthetic devices learning by the integration of different sensory information in a near future. A synergic fusion between proprioceptive and visual information has been proved a winning strategy. Further integration could be done by extending the application to the upper limb and, for what concerns the finger case, by involving tactile perception data (Caiti et al., 1995) in the device. The research trend on humanoid robots and/or helpful devices - prosthesis - has to be oriented towards this type of approach, that is the bio-mimetic approach - we mean imitating the biological system to build artifacts that better adapt to human needs (Magenes et al., 2002).

5. References

- Ash, H. E. & Unsworth, A. (1996). Proximal Interphalangeal Joint Dimensions for the Design of a Surface Replacement Prosthesis. *Proc Instn Mech Engrs*, 210, 95-108.
- Ash, H. E. & Unsworth, A. (2000). Design of a Surface Replacement Prosthesis for the Proximal Interphalangeal Joint. *Proc Instn Mech Engrs*, 214, 151-163.
- Brook, N., Mizrahi, J., Shoham, M., Dayan, J. (1995). A biomechanical model of index finger dynamics. *Med Eng Phys*, 17, 54-63.
- Brooks, R.A. & Stein, L.A. (1994). Building brains for bodies. *Autonomous Robots*, 1 (1), 7-25.
- Buchholz, B., Armstrong, T. J., Goldstein, S. A. (1992). Anthropometric Data for Describing the Kinematics of the Human Hand. *Ergonomics*, 35 (3), 261-273.
- Buchholz, B. & Armstrong, T. J. (1992). A Kinematic Model of the Human Hand to Evaluate Its Prehensile Capabilities. *J. Biomechanics*, 25 (2), 149-162.
- Buchner, H.J., Hines, M.J., Hemami, H. (1988). A dynamic model for finger interphalangeal coordination, *J Biomech*, 21:(6), 459-468.
- Caiti, A., Canepa, G., De Rossi, D., Germagnoli, F., Magenes, G., Parisini, T. (1995). Towards the realization of an artificial tactile system: fine-form discrimination by a tensorial tactile sensor array and neural inversion algorithms. *IEEE SMC Transaction*, 25 (6), 933 - 946.
- Da Cunha, F.L., Schneebeil, H.A., Dymnikov, V.I. (2000). Development of anthropomorphic upper limb prostheses with human-like interphalangeal and interdigital couplings. *Artificial Organs*, 24 (3), 193-197.
- Esteki, A. & Mansour, J. M. (1997). A Dynamic Model of the Hand with Application in Functional Neuromuscular Stimulation. *Annals of Biomedical Engineering*, 25, 440-451.
- Flash, T. & Hogan, N. (1985). The coordination of arm movements: an experimentally confirmed mathematical model. *J Neurosci*, 5 (7), 1688-1703.
- Garrett, J.W. (1971). The adult human hand: some anthropometric and biomechanical considerations. *Human Factors*, 13, 117-131.
- Gupta, A., Rash, G.S., Somia, N.N., Wachowiak, M.P., Jones, J., Desoky, A. (1998). The Motion Path of the Digits. *J. Hand Surg [Am]*, 23 (6), 1038-1042.

- Hahn, P., Krimmer, H., Hradetzky, A., Lanz, U. (1995). Quantitative analysis of the linkage between the interphalangeal joints of the index finger. *J Hand Surgery*, 20B, 696-699.
- Harris, C. & Rutledge, G.L. (1972). The functional anatomy of the extensor mechanism of the finger. *J Bone Joint Surgery*, 54 (1), 713-726.
- Haykin, S. (1999). *Neural Networks - A Comprehensive Foundation*, Prentice Hall, 2nd Ed
- Hoff, B. & Arbib, M. (1993). Models of trajectory formation and temporal interaction of reach and grasp. *J Motor Behav*, 25, 175-192.
- Jacobsen, S., Knutti, D.F., Johnson, R.T., Biggers, K.B. (1986). Design of the UTAH/MIT dexterous hand. *Proc IEEE Int Conf on Robotics and Automation*, Raleigh NC, 1520-1532.
- Kapandji, I. A. (1970). *The Physiology of the Joints*. E & S Livingstone, Edinburgh & London, 2nd Ed, vol 1
- Kyriakopoulos, K.J. & Saridis, G.N. (1988). Minimum jerk path generation. *Proc IEEE Int Conf on Robotics and Automation*, 364-369, Philadelphia.
- Laczko, J., Jaric, S., Tihanyi, J., Zatsiorsky, V.M., Latash, M. (2000). Components of the end-effector jerk during voluntary arm movements. *J Appl Biomech*, 16, 14-25.
- Lee, J.W. & Rim, K. (1990). Maximum finger force prediction using a planar simulation of the middle finger. *Proc Instit Mech Eng*, 204, 169-178.
- Lin, C.-S., Chang, P.-R., Luh, J.Y.S. (1983). Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Trans Autom Control*, AC-23 (12), 1066-1074.
- Magenes, G., Ramat, S., Secco, E.L. (2002). Life-like sensorimotor control: from biological systems to artifacts. *Current Psychology of Cognition*, 21 (4-5), 565-596.
- Massone, L. & Bizzi, E. (1989). A neural network model for limb trajectory formation. *Biol Cybern*, 61, 417- 425.
- Melchiorri, C. & Vassura, G. (1995). Implementation of whole hand manipulation capability in the U.B. Hand system design. *J Adv Robot*, 9 (5), 547-560.
- Nelson, L.W. (1983). Physical principles for economies of skilled movements. *Biol Cybern*, 46, 135-147.
- Okadome, T. & Honda, M. (1999). Kinematic construction on the trajectory of sequential arm movements. *Biol Cybern*, 80, 157-169.
- Ortega, J.M. & Rheiboldt, W.C. (1970). *Iterative solution of nonlinear equations in several variables*, Academic Press, New York.
- Pheasant, S. (1996). *Bodyspace: Anthropometry, Ergonomics and the Design of Work*, Taylor and Francis, London. 2nd Edition.
- Piazzini, A. & Visioli, A. (2000). Global minimum-jerk trajectory planning of robot manipulators. *IEEE Trans Ind Electron*, 47 (1), 140-149.
- Plamondon, R. (1995a). A kinematic theory of rapid human movements: part I. Movement representation and generation. *Biol Cybern*, 72, 295-307.
- Plamondon, R. (1995b). A kinematic theory of rapid human movements: part II. Movement time and control. *Biol Cybern*, 72, 309-320.
- Rosheim, M.E. (1994). *Robot evolution*, Wiley, New York.
- Craig, J.J. (1989). *Introduction to robotics: mechanics and control*, Addison-Wesley, 2nd Ed, New York.
- Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3 (6), 233-242.

- Secco, E.L. (1997). *Studi per la Realizzazione di una Mano Artificiale: Progettazione di un Dito a 3 gdl Ottimizzato per la Funzione di Presa*. Master Thesis, University of Padova, Italy.
- Secco, E.L., (2001). *Movement Control of a 3 d.o.f. artificial finger: dynamic learning and execution of the natural movement*. PhD Thesis, University of Pavia, Italy.
- Secco, E.L., Visioli, A., Magenes, G. (2001). A Minimum Jerk Approach for the Motion Planning of a Prosthetic Finger. *Proc. International Society of Biomechanics*, Zurich.
- Secco, E.L. & Magenes, G. (2002a). A feedforward neural network controlling the movement of a 3 degree of freedom finger. *IEEE SMC Transaction Part A*, 32 (3), 437-445.
- Secco, E.L., Magenes, G. (2002b). A Life-Like Control Algorithm for the Natural Movement of a 3 dof Finger. *Proc. Int. Joint Conf. BMES EMBS*, 2375-2376, Houston.
- Secco, E.L., Visioli, A., Magenes, G. (2002). Minimum jerk motion planning for a prosthetic finger. *Journal of Robotic Systems*, 21(7), 361-368.
- Secco, E.L., Valandro, L., Caimmi, R., Magenes, G., Salvato, B. (2005). Optimization of Two-Joint Arm Movements: A Model Technique or a Result of Natural Selection ?. *Biological Cybernetics*, 93 (4), 288-306.
- Simon, D. (1993). The application of neural networks to optimal robot trajectory planning. *Robot Auton Syst*, 11, 23-34.
- Sumbre, G., Fiorito, G., Flash, T., Hochner, B. (2005). Motor Control of Flexible Octopus Arms. *Nature*, 433, 595.
- Vicini, O. (2003). *Controllo neurale di un dito robotico allenato tramite movimenti di un dito umano*. Master Thesis, University of Pavia, Italy.

6. Links

- {1} Wittenstein Motion Control GmbH
<http://www.wittenstein.de/>
- {2} J. Lauren Banks
<http://www.ai.mit.edu/people/jessical>
- {3} The Babybot Project, LIRA-lab
http://www.lira.dist.unige.it/Projects/Projects/Babybot/Baby_bot.htm
- {4} The Humanoid Robotic Group
<http://www.ai.mit.edu/projects/humanoid-robotics-group/>
- {5} The Bio-Robots
<http://biorobots.cwru.edu/>

7. Acknowledgment

We would like to thank professor Antonio Visioli for scientific advice.

Vision Based Control of Model Helicopters

Erdoğan Altuğ¹, James P. Ostrowski², Camillo J. Taylor³

¹*Istanbul Technical University, Turkey*

²*Evolution Robotics, USA*

³*University of Pennsylvania, USA*

1. Introduction

The purpose of this paper is to explore control methodologies and vision algorithms to develop an autonomous unmanned aerial vehicle (UAV). UAVs include unmanned aircrafts, helicopters, blimps and other flying vehicles. An autonomous UAV brings enormous benefits and is suitable for applications like search and rescue, surveillance, remote inspection, military applications, therefore saving time, reducing costs and keeping human pilots away from dangerous flight conditions. UAVs are especially useful when (i) the working environment is inaccessible or hard to reach (planetary environments), (ii) flight is dangerous (due to war, contaminated environmental conditions), (iii) flight is monotonous, (vi) flight time is extended (atmospheric observations, data relay), (v) flight is not possible even by a skilled pilot (movie making, flight of experimental vehicles).

Various unmanned vehicles are in service in military and civilian areas. Fixed-wing vehicles have long-range since they are energy efficient, but they lack the maneuverability required for many UAV tasks. Blimps are easy to control when there are fewer disturbances like wind, and lift comes from natural buoyancy, but they lack maneuverability as well. The rotary-wing aerial vehicles (also called rotorcraft) - such as helicopters - have distinct advantages over conventional fixed-wing aircraft and blimps on surveillance and inspection tasks, since they can take-off and land in limited space and can easily hover above any target. Moreover, helicopters have the advantage of superior maneuverability. Unfortunately, this advantage comes from the dynamically unstable nature of the vehicle and it makes helicopters very hard to control. Sophisticated sensors, fast on-board computation, and suitable control methods are required to make rotorcraft based UAV stable.

Autonomy defined as “the quality or state of being self-governing”. Most of the commercial UAVs involve little or no autonomy. The goal is to increase the level of autonomy to fully autonomous operation including take-off, landing, hover (if platform capable of), way point navigation to more advanced autonomy modes such as searching, avoiding danger, combat, refueling, returning to base, etc. In addition, autonomy requires cooperation and communication with other vehicles. Currently a military UAV is supported with almost a dozen personnel to perform piloting, communications, and to control payload systems. The goal in the future is to reduce the personnel/UAV ratio (which is greater-equal to one currently) to lower than one, meaning that one personnel controlling various UAVs. Moreover, the vehicles will be autonomous to control their actions, interact with other vehicles to perform missions. This can be a commander controlling a fleet of military vehicles for combat, or a group of fire fighting UAVs commanded to extinguish a fire.

In order to create an autonomous UAV, precise knowledge of the helicopter position and orientation is needed. In the previous work involving autonomous flying vehicles, this information was obtained from Inertial Navigation Systems (INS), Global Positioning Systems (GPS) or other sensors like sonar sensor. Typically, multiple sensors are used to overcome limitations of individual sensors, thereby increasing the reliability and decreasing the errors. Vision sensors are primarily used for estimating the relative positions of some target, like a landing site or a ground vehicle. In other words, vision is used to answer the question "Where is the target?". The basic reason for this is the fact that special objects can be easily identified on the visual data relatively fast. Unfortunately, the vision system is not as fast as a gyro, and it is not as reliable as other sensors due to sensitivity to changes in lighting conditions. Vision is used for various research projects involving flying vehicles (Amidi, 1996; Shim, 2000; Sharp et al., 2001). In these papers, vision systems consisting of a single on-board camera or stereo cameras have been used, and the estimates were obtained by combining image data with readings from the inertial navigation systems, GPS or gyros. Our primary goal is to investigate the possibility of a purely vision-based controller. Limited payload capacity may not permit the use of heavy navigation systems or GPS. Moreover, GPS does not work in indoor and clustered environments. One can still setup an indoor GPS system with beacons or use small navigation systems though, cost considerations limit the use of these systems. Vision information can also be used to stabilize, to hover the helicopter, and also to track a moving object. In other words, we are asking the questions "Where is it?" and "Where am I?" at the same time. This study utilizes a two-camera system for pose estimation. Unlike previous work that utilizes either monocular views or stereo pairs, our two cameras are set to see each other. A ground camera with pan-tilt capabilities and an on-board camera are used to get accurate pose information.

There are various configurations of rotorcrafts. Conventional main rotor/tail rotor configuration; single rotor configuration; coaxial twin rotor configuration; side-by-side twin rotor configuration and multi-rotor configuration. The most popular configuration is the conventional main rotor/tail rotor configuration. This configuration has good controllability and maneuverability. However, the mechanical structure is complex and it requires a large rotor and a long tail boom (Castillo et al., 2005). Our interest in this paper will be on multi-rotor rotorcrafts. We have selected a remote-controlled, commercially available multi-rotor helicopter as our test bed. A quadrotor is a four-rotor helicopter, shown in Figure 1. The first full-scale quadrotor has been built by De Bothezat in 1921 (Gessow & Myers, 1967). It is an under-actuated, dynamic vehicle with four input forces and six output coordinates. Unlike regular helicopters that have variable pitch angle rotors, a quadrotor helicopter has four fixed-pitch angle rotors. Advantages of using a multi-rotor helicopter are the increased payload capacity and high maneuverability. However, multi-rotor helicopters are disadvantaged due to the increased helicopter weight and increased energy consumption due to the extra motors. The basic motions of a quadrotor are generated by varying the rotor speeds of all four rotors, thereby changing the lift forces. The helicopter tilts towards the direction of the slow spinning rotor, which enables acceleration along that direction. Therefore, control of the tilt angles and the motion of the helicopter are closely related and estimation of orientation (roll and pitch) is critical. Spinning directions of the rotors are set to balance the moments and eliminate the need for a tail rotor. This principle is also used to produce the desired yaw motions. A good controller should properly arrange the speed of each rotor so that only the desired states change.



Fig. 1. A commercially available four-rotor rotorcraft, Quadrotor.

Recent work in quadrotor design and control includes the quadrotor (Altuğ, 2003), and X4-Flyer (Hamel et al., 2002). Moreover, related models for controlling the VTOL aircraft are studied by Hauser et al. (1992), and Martin et al. (1996). The main concentration of this study is to use non-linear control techniques to stabilize and perform output-tracking control of a helicopter using vision based pose estimation.

2. Computer Vision

The estimation of motion (relative 3D position, orientation, and velocities) between two frames is an important problem in robotics. For autonomous helicopters, estimation of the motion of objects relative to the helicopter is important as well as estimation of the motion of the helicopter relative to a reference frame. This information is critical for surveillance and remote inspection tasks or for autonomous landing - taking off from a site. This information can be obtained using on-board sensors (like INS, GPS) or cameras. Usually the best sensor can be chosen based on the specific application. For a pose estimation in space for docking operations a camera system would be necessary since, other sensors like INS or GPS are not functional at space. Similarly, for a surveillance UAV used for military purposes, the estimation should not depend entirely on GPS or active sensors that could be manipulated, detected, or disturbed by the enemy.

The pose estimation problem has been a subject of many research projects for many years. The methods proposed use single-vision cameras, stereo cameras or direct 3D measuring techniques such as sonar sensors or laser range finders. Most of the pose estimation techniques are image based and they fall into these two categories: (i) point-based methods and (ii) model-based methods. Point-based methods use the feature points identified on a 2D image while model-based methods use the geometric models (e.g. lines, curves) and its image to estimate the motion. Moreover, the image based pose estimation (IBPE) methods that are point based can also be divided into two categories based on the number of the

cameras used: i) Single-cam methods and ii) Dual-camera methods. In this paper, we will describe the direct method, which is a single-cam method, and the two-camera method, which is a dual-camera method.

For our project, the goal is to obtain the pose from vision rather than complex navigation systems, INS or GPS. We are interested in point-based techniques that are real-time. For this purpose, pair of color cameras are being used to track the image features. These cameras track multi-color blobs located under the helicopter and ground. These blobs are located on a known geometric shape as shown in Figure 2. A blob-tracking algorithm is used to obtain the positions and areas of the blobs on the image planes. Therefore the purpose of the pose estimation algorithm is to obtain (x, y, z) positions, tilt angles (θ, ψ) , the yaw angle (ϕ) and the velocities of the helicopter in real-time relative to the ground camera frame.

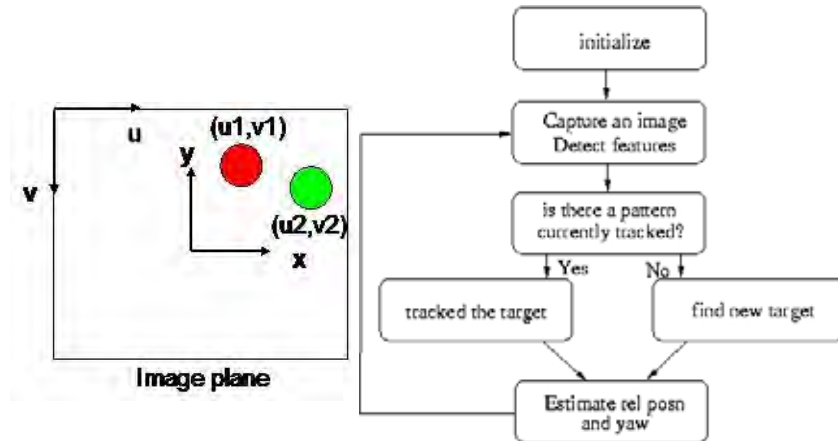


Fig. 2. Feature based pose estimation using color blobs (left). Tracking algorithm to estimate relative motion (right).

For the estimation of the pose $(x, y, z,$ and heading) of a flying robot, such as a blimp or a helicopter, an on-board camera, and multi-color blobs that are equally spaced grids on the floor can be used. A ground camera can also be used for the pose estimation of a flying vehicle. If the pitch and the roll angles are approximately zero, in that case two blobs will be enough for successful pose estimation. The way such a pose estimation method works is, the blobs are tracked with a ground camera and the blob separation on image plane is compared to distance L , the real blob separation, to estimate the altitude. Tracking is the action where a particular blob's whereabouts are known by successfully identifying it all time steps. Estimation of the relative motion and the absolute position and yaw needs a systematic approach; estimate the position of the pattern at each time step and update the absolute position of the pattern based on the estimated motion of the pattern. The biggest disadvantage of such a ground based pose estimation method is the fact that the estimation is limited to camera view area. A pan/tilt camera can be used to not only estimate the pose but also track the pattern as it moves. This increases the limited view area of the ground

camera. With the input of the pan and tilt angles, measured from the camera, the estimated relative position values should be translated due to the motion of the camera system. The pose estimation can be defined as finding a rotation Matrix, R , $R \in \text{SO}(3)$, defining the body fixed frame of the helicopter with respect to the fixed frame located at the ground camera frame, where $R^T R = I$, $\det(R) = 1$, and also the relative position $p \in \mathbb{R}^3$ of the helicopter with respect to the ground camera and also the velocities w and V of the helicopter as shown in Figure 3.

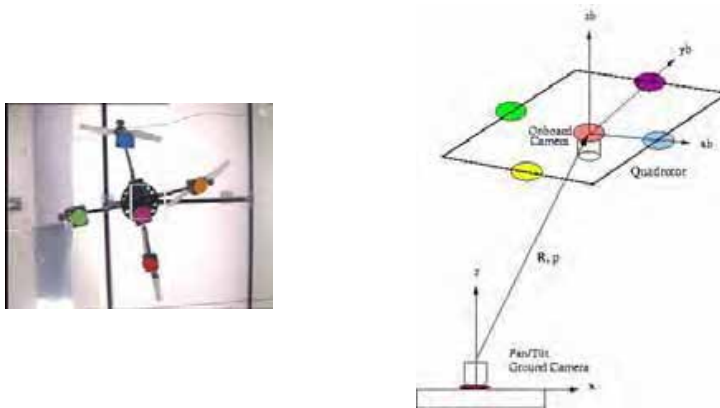


Fig. 3. Quadrotor helicopter pose estimation and tracking using color blobs.

In this section, we will introduce two methods to estimate pose of the helicopter in real-time. These methods are the direct method and the two-camera method. The methods will then be compared in simulations.

2.1. Direct Method

The purpose of the direct pose estimation algorithm is to obtain (x, y, z) positions, pitch angles (θ, ψ) and the yaw angle (ϕ) of the helicopter in real-time relative to the camera frame. Four different color blobs can be placed as a square pattern under the helicopter as shown in Figure 4. A ground camera is used to track the blobs to estimate the helicopter pose. Input of the camera intrinsic parameters (f_x, f_y, O_x, O_y) and image coordinates of the blobs are required. Moreover, the blob size and blob separation L is predetermined. A blob-tracking algorithm can be used to get the positions and areas of the blobs on the image plane (u_i, v_i, A_i) . The position of each blob with respect to fixed frame is calculated as

$$z_i = \frac{(f_x + f_y)\sqrt{C\pi}}{2\sqrt{A_i}}, \quad x_i = (u_i - O_x)\frac{z_i}{f_x}, \quad y_i = (v_i - O_y)\frac{z_i}{f_y} \tag{1}$$

where C is the number of pixels per unit area. The position of the helicopter is estimated by averaging the four blob positions. Normalization is performed using the real center difference between blobs. The yaw angle, ϕ , can be obtained from blob positions and the tilt angles can be estimated from the height differences of the blobs.

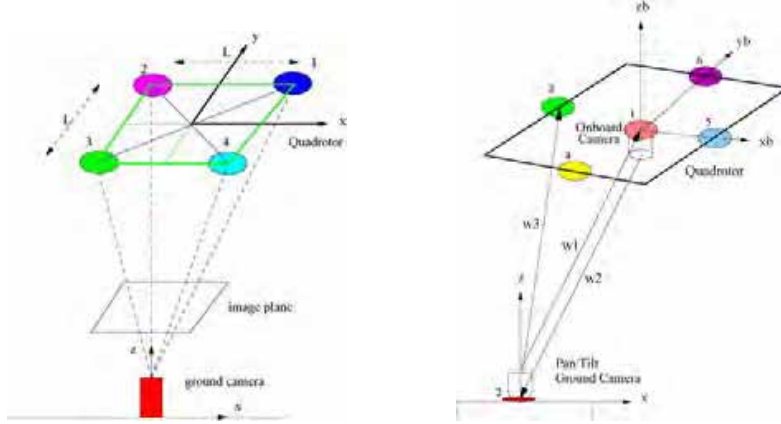


Fig. 4. Direct pose estimation method (left) and two-camera pose estimation method (right)

$$\phi = \arctan(y_1 - y_3 / x_1 - x_3), \quad \psi = \arcsin(z_3 - z_2 / L), \quad \theta = \arcsin(z_2 - z_1 / L) \quad (2)$$

These estimates depend on area calculations; therefore, they are sensitive to noise. This method works well on estimating yaw angle and positions of the helicopter, but it suffers greatly on estimating the tilt angles.

2.1. Two-camera Pose Estimation Method

The two-camera pose estimation method involves the use of two cameras that are set to see each other. One of the cameras is located at the ground and the other is an on-board camera looking downwards. This method is useful for autonomous take-off or landing, especially when the relative motion information is critical, such as landing on a ship at rough seas. Colored blobs are attached to the bottom of the quadrotor and to the ground camera as shown in Figure 4. Tracking two blobs on the quadrotor image plane and one blob on the ground image frame is found to be enough for accurate pose estimation. To minimize the error as much as possible, five blobs are placed on the quadrotor and a single blob is located on the ground camera. The blob tracking algorithm tracks the blobs and returns image values (u_i, v_i) for all of the features. The cameras have matrices of intrinsic parameters, A_1 and A_2 . Let be \bar{w}_i a unit vector from each camera to the blobs, and λ_i unknown scalars.

$$\bar{w}_i = \text{inv}(A_1) \cdot [u_i \quad v_i \quad 1]^T, \quad \bar{w}_i = \bar{w}_i / \text{norm}(\bar{w}_i) \quad \text{for } i=1,3,4,5,6 \quad (3)$$

$$\bar{w}_2 = \text{inv}(A_2) \cdot [u_2 \quad v_2 \quad 1]^T, \quad \bar{w}_2 = \bar{w}_2 / \text{norm}(\bar{w}_2) \quad (4)$$

Let \bar{L}_a be the vector pointing from blob-1 to blob-3 in Figure 4. Vectors \bar{w}_1 and \bar{w}_3 are related by

$$\lambda_3 \bar{w}_3 = \lambda_1 \bar{w}_1 + R \bar{L}_a \quad (5)$$

To simplify, let us take the cross product of this equation with \bar{w}_3

$$\lambda_1(\mathbf{w}_3 \times \mathbf{w}_1) = (R\bar{\mathbf{L}}_a \times \mathbf{w}_3) \quad (6)$$

This can be rewritten as

$$(\mathbf{w}_3 \times \mathbf{w}_1) \times (R\bar{\mathbf{L}}_a \times \mathbf{w}_3) = 0 \quad (7)$$

In order to solve the above equation, let the rotation matrix R be composed of two rotations: the rotation of θ degrees around the vector formed by the cross product of \mathbf{w}_1 and \mathbf{w}_2 and the rotation of α degrees around $\bar{\mathbf{W}}_1$. In other words

$$R = \text{Rot}(\bar{\mathbf{W}}_1 \times \mathbf{w}_2, \theta) \cdot \text{Rot}(\bar{\mathbf{W}}_1, \alpha) \quad (8)$$

where $\text{Rot}(\mathbf{a}, b)$ means the rotation of b degrees around the unit vector \mathbf{a} . The value of θ can be found from $\theta = \arccos(\bar{\mathbf{W}}_1 \cdot \mathbf{w}_2)$. Alternatively, one can use the cross product of \mathbf{w}_1 and \mathbf{w}_2 to solve θ angle. The only unknown left in Equation 8 is the angle α . Rewriting Equation 7 gives

$$(\mathbf{w}_3 \times \mathbf{w}_1) \times (\mathbf{w}_3 \times (\text{Rot}(\bar{\mathbf{W}}_1 \times \mathbf{w}_2, \theta) \cdot \text{Rot}(\bar{\mathbf{W}}_1, \alpha))\bar{\mathbf{L}}_a) = 0. \quad (9)$$

Let M be given as

$$M = (\mathbf{w}_3 \times \mathbf{w}_1) \times \{\mathbf{w}_3 \times [R(\bar{\mathbf{W}}_1 \times \mathbf{w}_2, \theta)]\} = 0. \quad (10)$$

Using Rodrigues' formula, $\text{Rot}(\bar{\mathbf{W}}_1, \alpha)$ can be written as

$$\text{Rot}(\bar{\mathbf{W}}_1, \alpha) = I + \hat{\mathbf{W}}_1 \sin \alpha + \hat{\mathbf{W}}_1^2 (1 - \cos \alpha) \quad (11)$$

Pre-multiplying Equation 11 with M and post-multiplying it with $\bar{\mathbf{L}}_a$ gives the simplified version of the Equation 9

$$M \cdot \bar{\mathbf{L}}_a + \sin \alpha \cdot M \hat{\mathbf{W}}_1 \cdot \bar{\mathbf{L}}_a + (1 - \cos \alpha) \cdot M \cdot (\hat{\mathbf{W}}_1)^2 \cdot \bar{\mathbf{L}}_a = 0. \quad (12)$$

This is a set of three equations in the form of $A \cos \alpha + B \sin \alpha = C$, which can be solved by

$$\alpha = \arcsin \frac{B \cdot C \pm \sqrt{(B^2 \cdot C^2 - (A^2 + B^2) \cdot (C^2 - A^2))}}{A^2 + B^2}. \quad (13)$$

One problem here is that $\alpha \in [\pi/2, -\pi/2]$ because of the \arcsin function. Therefore, one must check the unit vector formed by two blobs to find the heading, and pick the correct α value. The estimated rotation matrix will be found from Equation 8. Euler angles (ϕ, θ, ψ) defining the orientation of the quadrotor can be obtained from the rotation matrix, R . In order to find the relative position of the helicopter with respect to the inertial frame located at the ground camera frame, we need to find scalars λ_i . The λ_1 can be found using Equation 6. The other λ_i values can be found from the relation of the blob positions

$$\lambda_i \bar{\mathbf{W}}_i = \lambda_1 \bar{\mathbf{W}}_1 + R\bar{\mathbf{L}}_i \quad (14)$$

where $\bar{\mathbf{L}}_i$ is the position vector of the blob i , in body-fixed frame. To reduce the errors, λ_i values are normalized using the blob separation, L . The center of the quadrotor will be

$$[X \ Y \ Z]' = [\lambda_3 \bar{\mathbf{W}}_3 + \lambda_4 \bar{\mathbf{W}}_4 + \lambda_5 \bar{\mathbf{W}}_5 + \lambda_6 \bar{\mathbf{W}}_6] / 4. \quad (15)$$

2.3 Comparing the Pose Estimation Methods

The proposed direct method and the two-camera pose estimation methods are compared to other methods using a MATLAB simulation. Other methods used were a four-point algorithm (Ansar et al., 2001), a state estimation algorithm (Sharp et al., 2001), and a stereo pose estimation method that uses two ground cameras that are separated by a distance d . The errors are calculated using angular and positional distances, given as

$$e_{\text{ang}} = |\log(R^{-1} \cdot R^{\text{est}})| \quad e_{\text{pos}} = |\mathbf{p} - \mathbf{p}^{\text{est}}|. \quad (16)$$

R^{est} and \mathbf{p}^{est} are the estimated rotational matrix and the position vector. Angular error is the amount of rotation about a unit vector that transfers R to R^{est} . In order to compare the pose estimation methods, a random error up to five pixels was added on image values. The blob areas were also added a random error of magnitude ± 2 . During the simulation helicopter moves from the point (22, 22, 104) to (60, 60, 180) cm, while (θ, ψ, ϕ) angles change from (0.7, 0.9, 2) to (14, 18, 40) degrees. The comparison of the pose estimation methods and the average angular and positional errors are given on Table 1. The values correspond to average errors throughout the motion of the helicopter.

Method	Angular Error (degree)	Position Error (cm)
Direct	10.2166	1.5575
Four Point	3.0429	3.0807
Two-camera	1.2232	1.2668
Linear	4.3700	1.8731
Stereo	6.5467	1.1681

Table 1. Comparison of the Pose Estimation Methods using the angular and positional distances.

It can be seen from Table 1 that, the estimation of orientation is more sensitive to errors than position estimation. The direct method uses the blob areas, which leads to poor pose estimates due to noisy blob area readings. For the stereo method, the value of the baseline is important for pose estimation. The need for a large baseline for stereo pairs is the drawback of the stereo method. Based on the simulations, we can conclude that the two-camera method is more effective for pose estimation especially when there are errors on the image plane.

3. Helicopter Model

It is not an easy task to model a complex helicopter such as a quadrotor. In this section, our goal is to model a four-rotor helicopter as realistically as possible, so that we can derive control methodologies that would stabilize and control its motions. As shown in Figure 5, quadrotor helicopter has two clock-wise rotating and two counter-clock-wise rotating rotors, which eliminates the need for a tail rotor. Basic motions of the quadrotor are achieved by the trusts generated by its four rotors.

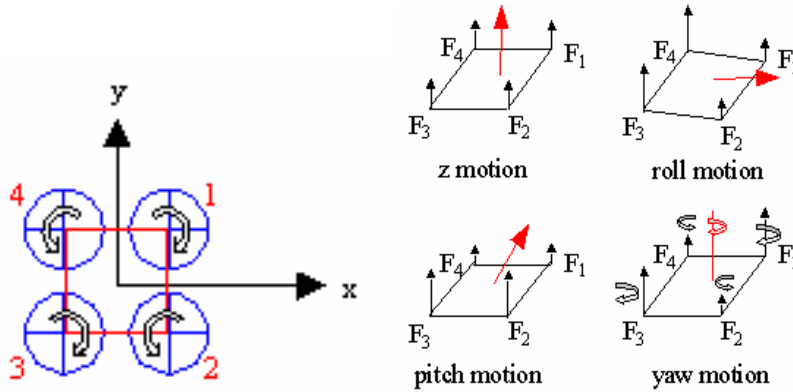


Fig. 5. Quadrotor helicopter can be controlled by individually controlling the rotor thrusts.

For a rigid body model of a 3D quadrotor given in Figure 6, a body fixed frame (frame B) is assumed to be at the center of gravity of the quadrotor, where the z-axis is pointing upwards. This body axis is related to the inertial frame by a position vector $\rho=[x \ y \ z]^T \in \mathbb{O}$ where O is the inertial frame and a rotation matrix $R:\mathbb{O} \rightarrow \mathbb{B}$, where $R \in \text{SO}(3)$. A ZYX (Fick angles) Euler angle representation has been chosen for the representation of the rotations, which is composed of three Euler angles, (ϕ, θ, ψ) , representing yaw, pitch, and roll respectively.

$$\text{RPY}(\phi, \theta, \psi) = \text{Rot}(z, \phi) \cdot \text{Rot}(y, \theta) \cdot \text{Rot}(x, \psi) \tag{17}$$

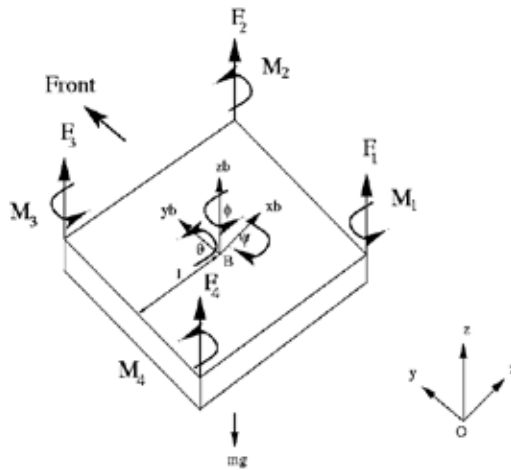


Fig. 6. 3D quadrotor helicopter rigid body model.

Let \bar{V} and $\bar{w} \in \mathbf{O}$ represent the linear and angular velocities of the rigid body with respect to the inertial frame. Similarly, let \bar{V}^b and $\bar{w}^b \in \mathbf{B}$ represent the linear and angular velocities of the rigid body with respect to the body-fixed frame. Let $\bar{\zeta}$ be the vector of Euler angles, $\bar{\zeta} = [\psi \ \theta \ \phi]^T$ and $\bar{w}^b = [p \ q \ r]^T$. The body angular velocity is related to Euler angular velocity by $\bar{w}^b = \text{unskew}(\mathbf{R}^T \dot{\mathbf{R}})$, where $\text{unskew}()$ term, represents obtaining vector \bar{w}^b from skew symmetric matrix; $\text{skew}(\bar{w}^b)$. The $\text{skew}(\bar{w}) \in \mathfrak{so}(3)$ is the skew symmetric matrix of \bar{w} . The Euler angle velocities to body angular velocities are mapped by, $\bar{\zeta} = \mathbf{J}\bar{w}^b$

$$\begin{pmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} 1 & S_\psi T_\theta & C_\psi T_\theta \\ 0 & C_\psi & -S_\psi \\ 0 & S_\psi / C_\theta & C_\psi / C_\theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \quad (18)$$

where S_ϕ denotes $\text{Sin}(\phi)$, C_ϕ denotes $\text{Cos}(\phi)$, and T_θ denotes $\text{Tan}(\theta)$. Using the Newton-Euler equations, we can represent the dynamics of the quadrotor as follows

$$\bar{V} = \frac{1}{m} \mathbf{F}_{\text{ext}} - \bar{w}^b \times \bar{V}^b, \quad \mathbf{I}_b \dot{\bar{w}}^b = \bar{M}_{\text{ext}} - \bar{w}^b \times \mathbf{I}_b \bar{w}^b, \quad \bar{\zeta} = \mathbf{J}\bar{w}^b \quad (19)$$

where \mathbf{I}_b is the inertia matrix, \mathbf{F}_{ext} and \mathbf{M}_{ext} are the external forces and moments on the body fixed frame given as

$$\mathbf{F}_{\text{ext}} = -\text{drag}_x \bar{i} - \text{drag}_y \bar{j} + (T - \text{drag}_z) \bar{k} - R \cdot \text{mg} \bar{k}, \quad \bar{M}_{\text{ext}} = M_x \bar{i} + M_y \bar{j} + M_z \bar{k}. \quad (20)$$

In this equation, T is the total thrust, M_x , M_y , and M_z are the body moments, \bar{i} , \bar{j} , \bar{k} are the unit vectors along x , y and z axes respectively. A drag force acts on a moving body opposite to the direction it moves. The terms drag_x , drag_y , drag_z are the drag forces along the appropriate axis. Let ρ be the density of air, A the frontal area perpendicular to the axis of motion, C_d the drag coefficient and V the velocity, then the drag force on a moving object is $\text{drag} = \frac{1}{2} C_d \rho V^2 A$. Assuming the density of air is constant then, the constants at above equation can be collected, and the equation can be written as $\text{drag} = C_d V^2$. The total thrust is (Prouty, 1995)

$$\mathbf{F} = bL = \frac{\rho}{4} \omega^2 R^3 abc (\theta_t - \phi_t) \quad (21)$$

where a is the slope of the airfoil lift curve, b is the number of blades on a rotor, c is the lift coefficient, L is the lift of a single blade, θ_t is the pitch at the blade tip, ϕ_t is the inflow angle at the tip, ω is the rotor speed, and R is the rotor radius. Note that the angle θ_t is constant for a quadrotor helicopter that has fixed pitch rotors. In addition, we assume that $\phi_t = 0$, implying that we ignore the change in direction of the airflow due to the motion of the quadrotor through the air. By collecting the constant terms as D , for hover or near-hover flight conditions this equation simplifies to $F_i = D \omega_i^2$. Successful control of the helicopter requires direct control of the rotor speeds, ω . Rotor speeds can be controlled by controlling

the motor torque. The torque of motor i , M_i , is related to the rotor speed ω_i as $M_i = I_r \omega_i^2 + K \omega_i^2$, where I_r is the rotational inertia of rotor i , K is the reactive torque due to the drag terms. For simplicity, we assume that the inertia of the rotor is small compared to the drag terms, so that the moment generated by the rotor is proportional to the lift force, i.e., $M_i = C F_i$, where C is the force-to-moment scaling factor. For simulations, a suitable C value has been experimentally calculated. The total thrust force T and the body moments M_x , M_y , and M_z are related to the individual rotor forces through

$$\begin{pmatrix} T \\ M_x \\ M_y \\ M_z \end{pmatrix} = \begin{pmatrix} l & l & l & l \\ -l & -l & l & l \\ -l & l & l & -l \\ C & -C & C & -C \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{pmatrix} \quad (22)$$

where F_i 's are the forces generated by the rotors. The matrix above which we denote by $N \in \mathbb{R}^{4 \times 4}$ is full rank for $l, C \neq 0$. This is logical since $C = 0$ would imply that the moment around z -axis is zero, making the yaw axis control impossible. When $l = 0$, this corresponds to moving the rotors to the center of gravity, which eliminates the possibility of controlling the tilt angles, which again implies a lack of control over the quadrotor states.

In summary, to move the quadrotor, motor torques M_i should be selected to produce the desired rotor velocities ω_i , which will change the thrust and the body moments in Equation 22. This will change the external forces and moments in Equation 20. This will lead to the desired body velocities and accelerations as given in Equation 19.

4. Helicopter Control

Unmanned aerial vehicles bring enormous benefits to applications like search and rescue, surveillance, remote inspection, military applications and saving human pilots from dangerous flight conditions. To achieve these goals, however, autonomous control is needed. The control of helicopters is difficult due to the unstable, complex, non-linear, and time-varying dynamics of rotorcrafts. Rotor dynamics, engine dynamics, and non-linear variations with airspeed make the system complex. This instability is desired to achieve the set of motions that could not be achieved by a more stable aircraft. In this work, our goal is to use external and on-board cameras as the primary sensors and use onboard gyros to obtain the tilt angles and stabilize the helicopter in an inner control loop. Due to the weight limitations, we can not add GPS or other accelerometers on the system. The controller should be able to obtain the relative positions and velocities from the cameras only. The selection of suitable control method for an UAV requires careful consideration of which states need to be observed and controlled, which sensors are needed and the rate of sensors. In this paper, we will introduce 3D quadrotor model and explain the control algorithms that are developed for these vehicles.

The helicopter model given in the previous section is a complicated, non-linear system. It includes rotor dynamics, Newton-Euler equations, dynamical effects, and drag. One can under some assumptions simplify the above model. Such a simplified model will be useful for derivation of the controllers.

Let us assume that

- The higher order terms can be ignored ($\vec{F}_{ext} \gg m\dot{w}^b \times \vec{r}^b$ and $\vec{M}_{ext} \gg \dot{w}^b \times I_b \dot{w}^b$)
- The inertia matrix I_b is diagonal
- The pitch (ψ) and roll (θ) angles are small, so that J in Equation 19 is the identity matrix

This leads to the following dynamical equations

$$\vec{V} = \frac{1}{m} \vec{F}_{ext}, \quad I_b \vec{w}^b = \vec{M}_{ext}, \quad \vec{\zeta} = \vec{w}^b. \quad (23)$$

The equations of motion can be written using the force and moment balance on the inertial frame.

$$\begin{aligned} \ddot{x} &= \left[\left(\sum_{i=1}^4 F_i \right) (C_\psi S_\theta C_\psi + S_\psi S_\theta) - K_x \dot{x} \right] / m \\ \ddot{y} &= \left[\left(\sum_{i=1}^4 F_i \right) (S_\psi S_\theta C_\psi - C_\psi S_\theta) - K_y \dot{y} \right] / m \\ \ddot{z} &= \left[\left(\sum_{i=1}^4 F_i \right) (C_\theta C_\psi) - mg - K_z \dot{z} \right] / m \end{aligned} \quad (24)$$

$$\ddot{\theta} = \frac{1}{J_1} (-F_1 - F_2 + F_3 + F_4), \quad \ddot{\psi} = \frac{1}{J_2} (-F_1 + F_2 + F_3 - F_4), \quad \ddot{\phi} = \frac{1}{J_3} (M_1 - M_2 + M_3 - M_4) \quad (25)$$

The J_i 's given above are the moments of inertia with respect to the corresponding axes, and the K_i 's are the drag coefficients. In the following, we assume the drag is zero, since drag is negligible at low speeds.

For convenience, we will define the inputs to be

$$\begin{aligned} u_1 &= (F_1 + F_2 + F_3 + F_4) / m = T / m \\ u_2 &= (-F_1 - F_2 + F_3 + F_4) / J_1 = T_x / J_1 \\ u_3 &= (-F_1 + F_2 + F_3 - F_4) / J_2 = T_y / J_2 \\ u_4 &= C(F_1 - F_2 + F_3 - F_4) / J_3 = T_z / J_3 \end{aligned} \quad (26)$$

where C is the force-to-moment scaling factor. The u_1 represents a total thrust/mass on the body in the z -axis, u_2 and u_3 are the pitch and roll inputs and u_4 is the input to control yawing motion. Therefore, the equations of motion become

$$\begin{aligned} \ddot{x} &= u_1 (C_\psi S_\theta C_\psi + S_\psi S_\theta) & \ddot{y} &= u_1 (S_\psi S_\theta C_\psi - C_\psi S_\theta) & \ddot{z} &= u_1 C_\theta C_\psi \\ \ddot{\theta} &= u_2 & \ddot{\psi} &= u_3 & \ddot{\phi} &= u_4 \end{aligned} \quad (27)$$

Noticing that the motion along the y -axis is related to the ψ tilt angle, similarly motion along the x -axis is related to the θ angle, one can design a backstepping controller. Backstepping controllers (Sasthy, 1999) are especially useful when some states are controlled through other states. Similar ideas of using backstepping with visual servoing have also been developed for a traditional helicopter (Hamel & Mahony, 2000).

Considering Equation 27, the use of a small angle assumption on ψ in the \ddot{x} term, and a small angle assumption on θ in the \ddot{y} term gives $\ddot{x} = u_1 C_\theta S_\psi$, $\ddot{y} = -u_1 C_\theta S_\psi$. From this equation, backstepping controllers for u_2 and u_3 can be derived (Altuğ et al., 2005). Controller u_2 controls angle θ in order to control x motions and controller u_3 controls angle ψ in order to control motions along the y -axis.

$$u_2 = \frac{1}{u_1 C_\theta C_\psi} (-5\dot{x} - 10\ddot{x} - 9u_1 S_\theta C_\psi - 4u_1 \dot{\theta} C_\theta C_\psi + u_1 \dot{\theta}^2 S_\theta C_\psi + 2u_1 \dot{\phi} S_\theta S_\psi - u_1 \dot{\phi}^2 S_\theta C_\psi) \quad (28)$$

$$u_3 = \frac{1}{u_1 C_\psi C_\theta} (-5\dot{y} - 10\ddot{y} - 9u_1 S_\psi C_\theta - 4u_1 \dot{\psi} C_\theta C_\psi + u_1 \dot{\psi}^2 S_\psi C_\theta + 2u_1 \dot{\psi} S_\psi S_\theta - u_1 \dot{\psi}^2 S_\psi C_\psi) \quad (29)$$

The sequential derivation of the backstepping controller involves finding suitable Lyapunov functions therefore; the controllers are guaranteed to exponentially stabilize the helicopter.

PD controllers on the other hand, can control the altitude and the yaw.

$$u_1 = \frac{g + K_{p1}(z_u - z) + K_{d1}(\dot{z}_u - \dot{z})}{C_\theta C_\psi}, \quad u_4 = K_{p2}(\phi_d - \phi) + K_{d2}(\dot{\phi}_d - \dot{\phi}) \quad (30)$$

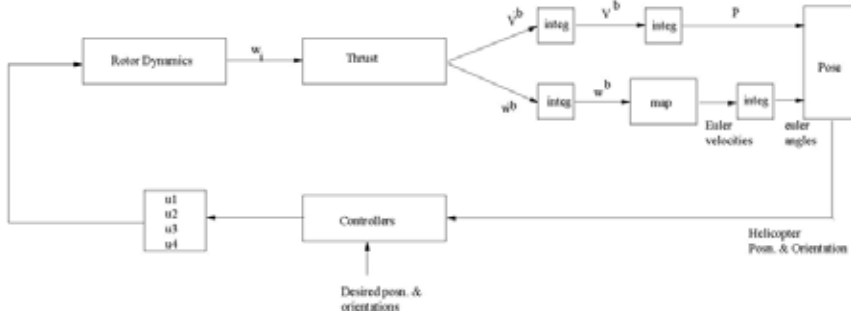


Fig. 7. Helicopter simulation model developed in MATLAB Simulink.

The proposed controllers are implemented on a MATLAB, Simulink simulation as shown in Figure 7. The helicopter model is based on the full non-linear model given by Equation 19. The following values are used for the simulation: The force to moment ratio, C , was found experimentally to be 1.3. The length between rotors and center of gravity, l , was taken as 21 cm. The inertia matrix elements are calculated with a point mass analysis as; $I_x = 0.0142$ kg/m², $I_y = 0.0142$ kg/m² and $I_z = 0.0071$ kg/m². Mass of the helicopter is taken as 0.56 kg. The drag coefficients are taken as $C_x = 0.6$, $C_y = 0.6$ and $C_z = 0.9$. Gravity is $g = 9.81$ m/s². The thrust forces in real flying vehicles are limited. Therefore, the maximum and minimum inputs are defined by

$$\frac{-F_{\max}}{m} \leq u_1 \leq \frac{4F_{\max}}{m}, \quad \frac{-2F_{\max}}{l} \leq u_2 \leq \frac{2F_{\max}}{l}, \quad \frac{-2F_{\max}}{l} \leq u_3 \leq \frac{2F_{\max}}{l}, \quad -2CF_{\max} \leq u_4 \leq 2CF_{\max} \quad (31)$$

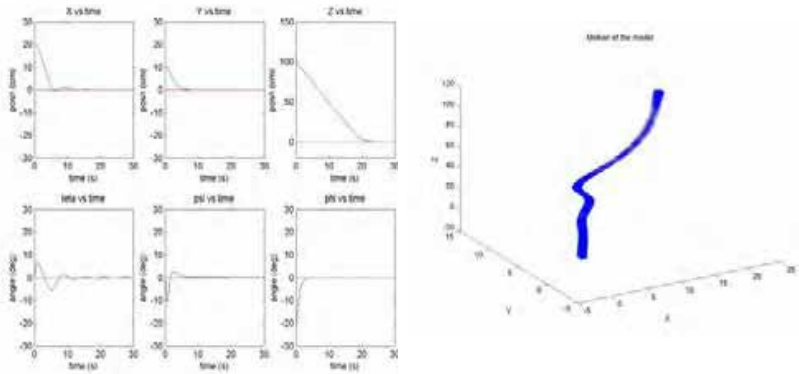


Fig. 8. Helicopter control simulation, using backstepping and PD controllers, and helicopter motion during this simulation.

In the first simulation, helicopter is being controlled with the proposed PD and backstepping controllers. The simulation results in Figure 8 shows the motion of the quadrotor from position (20, 10, 100) to the origin, while reducing the yaw angle from -20 to zero degrees.

The controller should be strong enough to handle random errors that are occurring because of the pose estimation error and disturbances. In order to simulate the robustness of the controllers to error, random error have been introduced on x , y , z , and yaw values. Error on x , y was with variance of 0.5 cm., error on z was with variance of 2 cm., and error on yaw was with variance of 1.5 degrees. In the simulation helicopter moves from 100 cm. to 150 cm. while reducing the yaw angle from 30 degrees to zero as shown in Figure 9. Although there were considerable error on the states, controllers were able to stabilize the helicopter. The mean and standard deviation are found to be 150 cm. and 1.7 cm. for z and 2.4 and 10.1 degrees for yaw respectively.

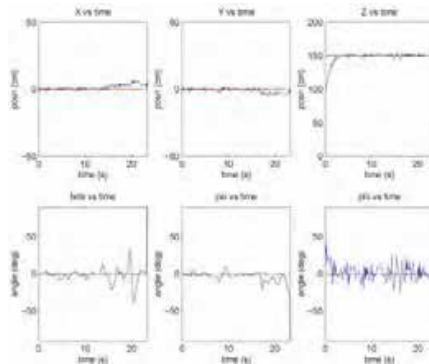


Fig. 9. Backstepping controller simulation with random noise at x , y , z and ϕ values.

One of the biggest problems in vision-based control is the fact that the vision system is not a continuous feedback device. Unlike sensors that have much higher rate than the vision updates such as accelerometer, or potentiometer, the readings - images - has to be captured, transferred and analyzed. Therefore, to simulate the discrete nature of the feedback system, this problem has to be included in the model. Usually the frame rates of many cameras are 20 to 30 Hz. A frame rate of 15 Hz. will be used for the overall sampling rate of this sensory system. The Figure 10 shows the results of the simulation, where the x, y and z positions are sampled at 15 Hz. The controllers are robust enough to handling the discrete inputs. A simple comparison of the plots shows that discrete sampling causes an increased settling time.

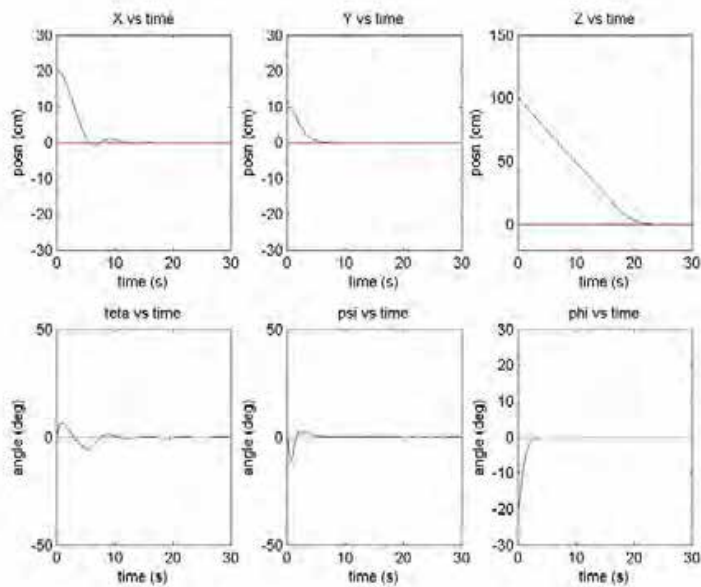


Fig. 10. The effect of the delay on the simulation.

Considering the simulations performed, PD control was successful to control altitude and heading of the helicopter. The results on PD controllers depend on the correct selection of gains K_p and K_d . Considering the settling time, and the ability to perform with noisy or delayed data, the backstepping controllers are much better than the PD controllers. Moreover, the backstepping controllers are guaranteed to exponentially stabilize the helicopter.

5. Experiments

This section discusses the applications of the image based pose estimation and control methods. One logical starting point is to decide where to put cameras, how many cameras

to use, location of the vision computer and computation time. If a remote computer will process the images, transferring images to this computer and transfer of commands to the helicopter will be required. The use of an onboard camera and processing images locally not only eliminates the need of information transfer, but also is very useful for other task that are usually required by the vehicle, such as locating a landing pad. The disadvantage of this is the increased vehicle weight, the need of more powerful computers onboard the vehicle.

The proposed algorithms implemented on a computer vision system. We used off the shelf hardware components for the system. Vision computer is a Pentium 4, 2 GHz machine that had an Imagination PXC200 color frame grabbers. Images can be captured at 640×480 resolution at 30 Hz. The camera used for the experiments was a Sony EVI-D30 pan/tilt/zoom color camera. The algorithm depends heavily on the detection of the color blobs on the image. When considering color images from CCD cameras, there are a few color spaces that are common, such as RGB, HSV, and YUV. The YUV space has been chosen for our application. The gray scale information is encoded in the Y channel, while the color information is transmitted through the U and V channel. Color tables are generated for each color in MATLAB. Multiple images and various lighting have to be used to generate the color tables, to reduce the effect of lighting condition changes. The ability to locate and track various blobs is critical. We use the blob tracker routines. The blob tracking routings use the images and the pregenerated color tables to identify the color blobs in real-time. It returns the image coordinates of all color blobs as well as the sizes of the blobs. It can track up to eight different blobs at a speed depending on the camera, computer and frame grabber. Our system could be able to track the blobs at about 20 Hz.

To make a helicopter fully autonomous, we need a flight controller as shown in Figure 11. An off-board controller receives camera images, processes them, and sends control inputs to the on-board processor. On board processor stabilizes the model by checking the gyroscopes and listens for the commands sent from the off-board controller. The rotor speeds are set accordingly to achieve the desired positions and orientations.

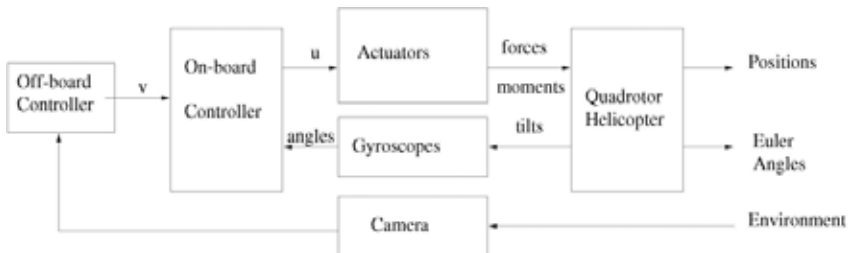


Fig. 11. Diagram of the on-board controller.

The off-board controller (the ground system) is responsible for the main computation. It processes the images, sets the goal positions, and sends them to the on-board controller using the remote controller transmitter as shown in Figure 12.

The proposed controllers and the pose estimation algorithms have been implemented on a remote-controlled battery-powered helicopter shown in Figure 13. It is a commercially

available model helicopter called HMX-4. It is about 0.7 kg, 76 cm. long between rotor tips and has about three minutes flight time. This helicopter has three gyros on board to stabilize itself. An experimental setup shown in Figure 13 was prepared to prevent the helicopter from moving too much on the x-y plane, while enabling it to turn and ascend/descend freely. Vision based stabilization experiments were performed using two different methods; direct method, which is using a single ground camera, and the two-camera pose estimation method.

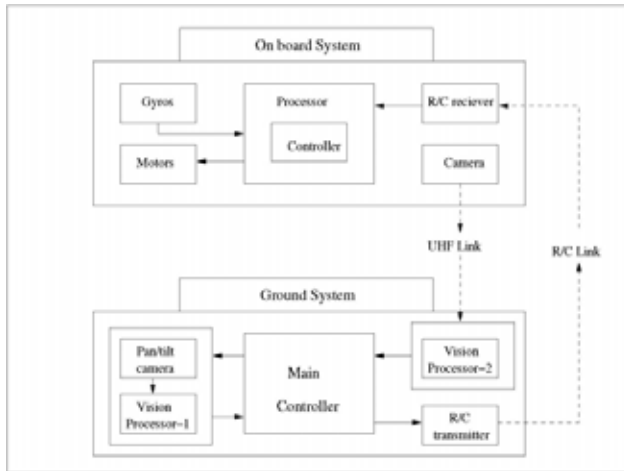


Fig. 12. Experimentation system block diagram, including the helicopter and the ground station.

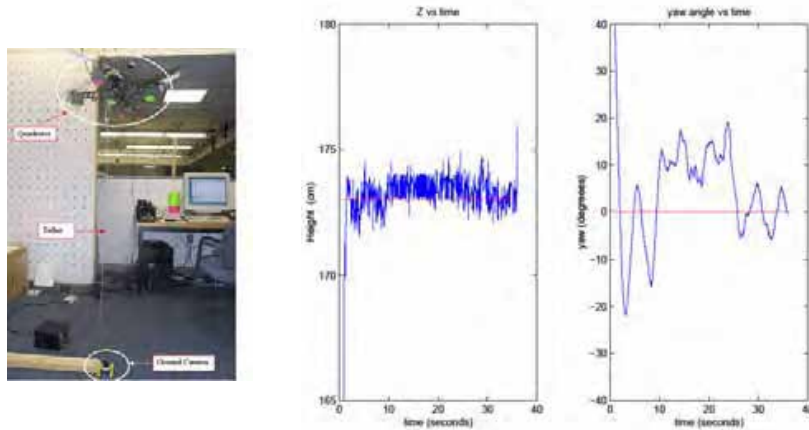


Fig. 13. Altitude and yaw control experiment performed with a single ground camera based on direct pose estimation method, and the results.

The first experiment involves the use of a single ground camera and the direct pose estimation method. The helicopter pose is estimated using image features, as well as areas of the image features. The goal is to control the helicopter at the desired altitude and the desired heading. In this experiment altitude and yaw angle are being controlled with PD controllers. Figure 13 shows the results of the altitude and yaw control experiment using this method.

The second experiment is the control of the quadrotor helicopter using the two-camera pose estimation method. In this experiment, two separate computers were used. Each camera was connected to separate computers that were responsible for performing blob tracking. PC-1 was responsible for image processing of the on-board camera. The information is then sent to PC-2 via the network. PC-2 was responsible for the ground pan/tilt camera control, image processing, and calculation of the control signals for the helicopter. These signals were then sent to the helicopter with a remote control device that uses the parallel port. The backstepping controllers for x and y motions and PD controllers for altitude and heading were implemented for the experiment. Figure 14 shows the results of this experiment using the two-camera pose estimation method. The mean and standard deviation are found to be 106 cm. and 17.4 cm. for z , 4.96 degrees, and 18.3 degrees for heading respectively. The results from the plots show that the proposed controllers do an acceptable job despite the pose estimation errors and errors introduced by the tether.

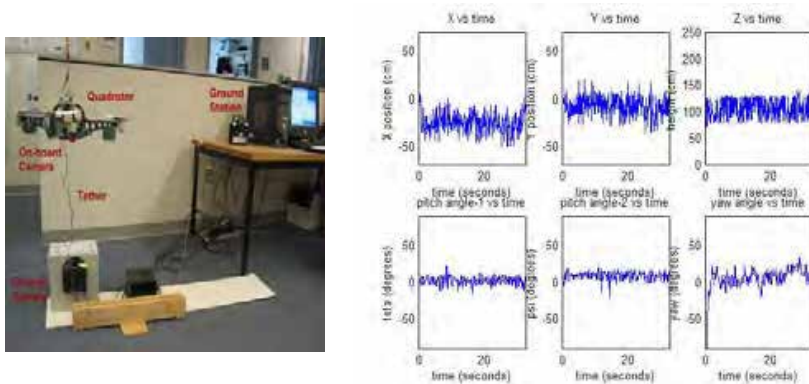


Fig. 14. The experimental setup and the results of the height x , y and yaw control experiment with two-camera pose estimation method.

6. Conclusions and Future Work

In this work, we have presented pose estimation algorithms and non-linear control techniques to build and control autonomous helicopters. We introduce a novel two-camera method for helicopter pose estimation. The method has been compared to other pose estimation algorithms and shown to be more effective, especially when there are errors on the image plane. A three dimensional quadrotor rotorcraft model has been developed. Nonlinear backstepping and PD controllers have been used to stabilize and

perform output-tracking control on the 3D quadrotor model. With the simulations performed on MATLAB - Simulink, controllers shown to be effective even when there are errors in the estimated vehicle states. Even in the presence of large errors in camera calibration parameters or in image values, global convergence can be achieved by the controllers. The proposed controllers and the pose estimation methods have been implemented on a remote control, battery-powered, model helicopter. Experiments on a tethered system show that the vision-based control is effective in controlling the helicopter.

Vision system is the most critical sensory system for an aerial robot. No other sensor system can supply relative position and relative orientation information like it. Especially tracking a moving target can only be possible with a vision system. One of the drawbacks of the vision system is that, it is not reliable when the lighting on the scene changes, and it is sensitive to vibration. In addition, weight and power consumption are other important parameters limiting the use of vision in mini UAVs. With recent advances in microcomputers and cameras, it will become further possible to achieve real-time feature extraction and control with commercial of the shelf parts. Our future work will concentrate on development of a bigger helicopter UAV for outdoor flight. A Vario model helicopter has already been integrated with a suite of sensors including IMU, GPS, barometric pressure altimeter, sonar, camera, and a Motorola MPC-555 controller. A helicopter ground test platform has been developed to test the helicopter system. Our future work will be to further explore control and vision algorithms using this helicopter. It is expected that further improvement of the control and the vision methods will lead to highly autonomous aerial robots that will eventually be an important part of our daily lives.

7. References

- Altuğ, E. (2003). Vision Based Control of Unmanned Aerial Vehicles with Applications to an Autonomous Four Rotor Helicopter, Quadrotor. Ph.D. Thesis, University of Pennsylvania, USA
- Altuğ, E.; Ostrowski, J. P. & Taylor, C. J. (2005). Control of a Quadrotor Helicopter Using Dual Camera Visual Feedback. *The International Journal of Robotics Research*, Vol. 24, No. 5, May 2005, pp. 329-341
- Amidi, O. (1996). An Autonomous Vision Guided Helicopter. Ph.D. Thesis, Carnegie Mellon University, USA
- Ansar, A.; Rodrigues, D.; Desai, J.; Daniilidis, K.; Kumar, V. & Campos, M. (2001). Visual and Haptic Collaborative Tele-presence. *Computer and Graphics, Special Issue on Mixed Realities Beyond Convention*, Vol. 25, No. 5, pp. 789-798
- Castillo, P.; Lozano, R. & Dzul, A. (2005). *Modelling and Control of Mini-Flying Machines*, Springer-Verlag London Limited
- Hamel, T.; Mahony, R. (2000). Visual Servoing of a class of under actuated dynamic rigid-body systems. *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, Vol. 4, pp. 3933-3938
- Hamel, T.; Mahony, R. & Chrietie, A. (2002). Visual Servo Trajectory Tracking for a Four Rotor VTOL Aerial Vehicle. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, D.C., pp. 2781-2786
- Hauser, J.; Sastry, S. & Meyer, G. (1992). Nonlinear Control Design for Slightly Non-Minimum Phase Systems: Application to V/STOL Aircraft. *Automatica*, vol. 28, No:4, pp. 665-679

- Gessow, A. & Myers, G. (1967). *Aerodynamics of the helicopter*, Frederick Ungar Publishing Co, New York, Third Edition
- Martin, P.; Devasia, S. & Paden, B. (1996). A Different Look at Output Tracking Control of a VTOL Aircraft, *Automatica*, vol. 32, pp. 101-107
- Prouty, R. *Helicopter Performance, Stability, and Control*. (1995). Krieger Publishing Company
- Sastry, S. *Nonlinear Systems; Analysis, Stability and Control*. (1999). Springer-Verlag, New-York
- Sharp, C. S.; Shakernia, O. & Sastry, S. S. (2001). A Vision System for Landing an Unmanned Aerial Vehicle, *IEEE Conference on Robotics and Automation*, Seoul, Korea
- Shim, H. (2000). Hierarchical Flight Control System Synthesis for Rotorcraft-based Unmanned Aerial Vehicles. Ph.D. Thesis, University of California, Berkeley

Multi – Agent System Concepts Theory and Application Phases

Adel Al-Jumaily, Moha'med Al-Jaafreh
*University of Technology Sydney
Australia*

1. Introduction

This chapter presents a recent research studies in multi-agent system, which has been using in many areas to improve performance and quality as well as reduce cost and save time by using available agents to solve many problems. Pursuit problem will be discussed to demonstrate the cooperation of multi agent systems as multi agent robots. Different pursuit algorithms are implemented for acquiring a dynamic target in unknown environment.

The multi-agent system concepts appeared recently and it is extremely distributed in all research areas; to solve problems by many agents cooperation. These agents, which have been using for multi-agent system, are defined as an entity; software routine, robot, sensor, process or person, which performs actions, works and makes decision (Arenas & Sanabria, 2003). In human society's concepts, the cooperation means "an intricate and subtle activity, which has defied many attempts to formalize it" (D'Inverno et al., 1997). Artificial and real social activity in social systems is the paradigm examples of Cooperation. In multi-agent concepts side, there are many definitions for cooperation; the most popular definitions are

Definition 1: "The multi-agents working together for doing something that creates a progressive result such increasing performance or saving time" (Gustafson & Matson, 2003).

Definition 2: "One agent adopts the goal of another agent. Its hypothesis is that the two agents have been designed in advance and, there is no conflict goal between them, furthermore, one agent only adopts another agent's aim passively."

Definition 3: "One autonomous agent adopts another autonomous agent's goal. Its hypothesis is that cooperation only occurs between the agents, which have the ability of rejecting or accepting the cooperation" (Changhong et al., 2002).

The multi-agents system is divided to theory and application phases (Changhong et al., 2002). Cooperation taxonomy, cooperation structure, cooperation forming procedure and others are related to theory phase. For application phases, the mobile agent cooperation, information gathering, sensor information and communication and others have been studied. The following sections will show the latest researches of multi-agent system from both theory and application phases.

2. Theory of Multi-Agent

The theory of any science forms the core and facilitates understand ability and documentary of that science (e.g. multi agent system). To explain multi-agents cooperation theory the

Cooperation structure, Cooperative problem solving, Evolution of cooperation, Negotiation, Coalition and Cooperation taxonomy, will be discussed in detail in next sections.

2.1 Cooperation Structure

Cooperation in multi agent system is divided to complete structure and incomplete structure depending on the goal dividing. These complete and incomplete cooperation structures include the following cooperation structures:

- Cooperation of one agent to one agent coalition (CATC).
- Cooperation of one agent coalition to one agent (CCTA). (e.g. goal needs a group of agents to complete together, or group of agents can complete more effectively at less cost than one agent.)
- Cooperation of one agent coalition to another agent coalition (CCTC).
- Cooperation of two agent coalitions on each other's goal exchanging (CGE). (e.g. two agents (or agent groups) adopt each other's goal).

These structures are illustrated in figure (1).

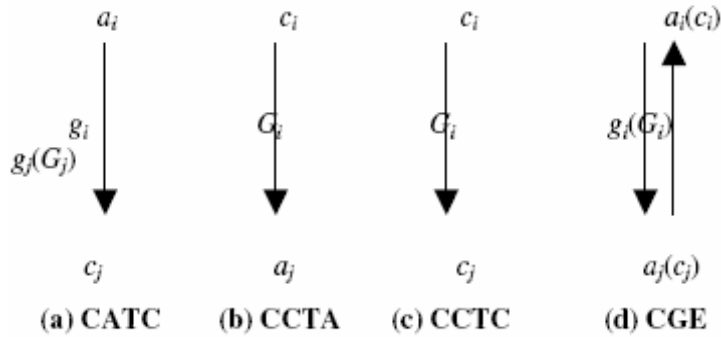


Fig. 1. Types of multi-agents cooperation structure.

Where a_i donates agent i , g_i denotes the goal of a_i , c_i donates agent coalition i , G_i donates the goal of c_i . The cooperation between agents is implemented through different communication techniques. The Speech Act is the basis communication technique of multi-agent interaction and it can be isolated from the content that speech embodied (Changhong et al., 2002). Speech act technique includes many communication primitives such as request cooperation, reply to the request, receive affirm, reject and other primitives.

In other hand, the cooperation structure is classified to implicit cooperation, explicit cooperation, and dynamic cooperation according to three dimensions of taxonomy (Gustafson & Matson, 2003):

- The amount of information that an agent has about other agents.
- The type of communications between agents.
- The amount of knowledge that agent has about the goals.

There are different states of taxonomy dimensions; which determine the cooperation types and facilitate the distribution of tasks between agents. These states are illustrated in table (1).

Information	Communication	Goal knowledge
Local state information	Implicit communication	Agent knows about its goal
Neighbourhood state information	Explicit communication "broadcast message"	Agent knows about its goal and the neighbours' goals
Global state information	Send and receive messages between robots	Agent knows about all robots' goals

Table 1. States of taxonomy dimensions.

From states of taxonomy dimensions table:

- 1) The implicit cooperation type consists of agents know about their goals or about neighbours' goals, use implicit communication and have information of local or neighbourhood agents' states.
- 2) The explicit cooperation type consists of agents know about all agents' goals, send messages to specific agent and have information of neighbourhood or global information.
- 3) The dynamic cooperation type consists of agents know about all agents' goals, send and receive messages between agents and have information of global state.

2.2 Cooperative problem solving

Performing cooperative problem solving requests a great deal of knowledge between agents. Agents decompose problem to sub-problems; which facilitates allocating tasks between them. There are many conditions to achieve cooperation between agents.

P. M. Jones & Jacobs listed these conditions (Jones & Jacobs, 2000):

- 1) There must be two or more reasoning agents.
- 2) Agents must be sharing the same environment, so that in some sense, their actions or the effects of their actions can be mutually perceived.
- 3) Each agent must be contributing by some productive elements, and they must be working together
- 4) Availability of local goals.

Jones's theory assumed some tenets of human-computer cooperative solving. These tenets are human in charge, mutual intelligibility, openness and honesty, multiple perspectives and management of trouble. In addition, some degrees of mutual understanding or shared meaning is required for human-computer cooperative problem solving; as example the computer "partner" is based on a normative model of task requirements. (Jones & Mitchell, 1991)

Cooperative Problem Solving concepts are metaphors, functional purpose, general functions, general and specific material, social mechanisms for cooperative problem solving, and cooperative mechanisms. Table (2) shows the concepts and their elaboration and examples.

Models of cooperative problem require a domain of knowledge to facilitate communications of goals, information and the coordination of activity. Bainbridge argues for the notion of contextual models; which highlights active seeking and structuring of information for building up a "structure of inference" about the current and future potential states of the system; such as prediction, planning, anticipation, flexibility, adaptability, organization of behaviour and management of multiple concurrent activities (Bainbridge, 1993).

Cooperative problem solving concept	Elaboration and Examples
Metaphors	Information processing system, culture and machine
Functional purpose	Augmentative, integrative and debative
General functions	Sharing goals or information and coordination activity (articulation work, redirection of attention)
General and specific material and social mechanisms for cooperative problem solving	Space: remote or proximate Time: synchronous or asynchronous Directness: Direct or machine-mediated Autonomy: distributed or collective e-mails, systems, artefacts, computer software and social organisation
Cooperative mechanisms	Request-based, inference-based and structurally-based

Table 2. Cooperative problem solving concept.

Operator function model (OFM) plan and goal graph contextual control are used to model human-machine interaction and assist in the design of user interfaces, intelligent associate systems and intelligent tutoring systems (Mitchell, 1999). OFM is a model of heterarchic-hierarchical network of nodes, which embodies how a human operator manages multiple synchronized activities in a dynamic event-driven world.

In OFM, Expert systems control intelligent associate systems; which are intended to act as intelligent resources for competent, rather than off-line consultants of presumably novice users. There are many examples of intelligent associate systems; such as the expert critic approach, IDEA, pilot's associate system and ISAM-CoSMO-ASPIRE family.

2.3 Evolution of cooperation

The competitive evolution is type of cooperation evolution; which used to facilitate emergent behaviour practically cooperation. This type is employed in pursuit and evasion domain as well as interrelated predator-pray systems. There are different artificial approaches of competitive evolution applied in pursuit domain; single pool approach, plasticity approach and multiple pools approach. A system of robots uses a single genotype illustrates a single pool approach, a system of robots uses learning mechanism characterizes plasticity approach and multiple pools approach is used for robots that share different genotypes (Nitschke, 2003).

Floreano et al. appraised a competitive evolutionary of cooperation in predator-pray scenario by using two mobile robots in evolutionary robotics experiments. Then they compared their procedure with single agent evolution. Finally, a fast comparative evolution of different behavioural strategies was observed with competitive co-evolution (Floreano et al., 1998).

G. Nitschke employed cooperative co-evolution in his research of pursuit evasion game. Briefly, His research is about a team of three robots "pursuers" cooperating to halt one of other three robots "evaders" (Nitschke, 2003). He used the multiple pools competitive evolution approach and yielded great performance compared with single pool and plasticity approaches. The multiple pools present that performance; because each robot of "pursuers and evaders" assigned to genotype from different population, and this encourages behavioural specialization of the team. In the other hand, single pool approach provides simplicity in calculation and behavioural encoding of team fitness as well as plasticity approach allows specialization of team member behaviour.

Nitschke used three strategies of cooperative pursuit by using three pursuers in each strategy:

- 1) Encirclement strategy: pursuers encircle and in close to an evader to force it to move in same direction then spin in its current position till tired.
- 2) Entrapment strategy: two pursuers moved in the same direction of evader different sides and third pursuer blocks from front to mobilize evader.
- 3) Role-switch strategy: two pursuers moved in the same direction of evader different sides and third pursuer moved around the other pursuers to mobilize evader in triangular formation.

2.4 Negotiation in Multi-Agents

The multi-agent cooperation was defined in third definition as “The multi-agents working together for doing something” (Gustafson & Matson, 2003). The vital member in multi-agents technology is group working; which needs a communication and negotiation between agents. Negotiation means “A key form of interaction that enables groups of agents to arrive at a mutual agreement regarding some belief, goal or plan” (Beer et al., 1998).

The negotiation between agents is implemented by different types, such as argumentation, protocols in the style of the contract net and auctions. The selection of negotiation type depends on the environment of problem, which has to be solved (D’Inverno et al., 1997).

The work group at IWMAS’98 suggested that the cases of negotiation can be distinguished by:

1. Challenges occur in managing agents interactions.
2. The larger processes where agents contribute.
3. Mechanisms will be used to maintain consistency of the challenges.
4. The complexity of negotiation.

Negotiations, according to their parent “going concern”, are characterized:

1. Negotiation techniques differ in the level of “common knowledge” with reference to the context that they presume; such as cooperative negotiations,.
2. Negotiation techniques differ in the side effects and impose on the context. The course of the negotiation in a peace negotiation or labour discussions can significantly effect the subsequent direction of the talks.
3. Negotiation episodes may be connected with another or with other actions in different ways to support the larger context (Parunak, 1999).

Coherent mechanisms dimension examines specific mechanisms; which are used by agents to ensure coherence in pursuit of objectives. Some mechanisms depend on structures in individual participants only “Solipsistic Mechanisms”, while others rely on structures shared among participants “Communal Mechanisms” (Parunak, 1999).

Moreover, Negotiation is classified according to goals’ classes. The next distinctions are specified to clear goals’ classes:

1. Static goals need simple and bounded negotiations.
2. Dynamic goals require renegotiation or meta-level negotiation.
3. An agent negotiating of future action has to address issues of time management which don’t arise in negotiations of present action.
4. Contentious goals need negotiation of mechanisms guarding against harmful threats.

Besides, Communal Mechanisms requires for agents participated in certain and common things, take the form of a common language, this language may exist at different levels of difficulty:

1. First and simplest level is a common ontology, a contract on how to divide up the world, supported by a currency with general agreed-upon valuation and a shared vocabulary.
2. Second level is meta-level which needs a contract to support some types of utterances, such as structured templates for various classes of discourse or rudimentary types of speech acts.
3. Third level is a static protocol which outlines a fixed order of utterances of different categories.
4. Forth and complex level of negotiation is occurred when agents can response to the received stream of speech acts and decide dynamically suitable type of utterances.

The main objective of negotiation is to achieve communication between agents in multi-agent project by protocols. The protocols of the community are used to assess the complexity of a negotiation mechanism (Parunak, 1999).

Several classes of the negotiation protocol are arranged in table (3) in a cline form ascending from less complexity to more complexity.

Class of Protocol	Description	Minimal Language
Reactive	Sense, then Act	Environment
Command	Master agent sends unilateral instructions to servant	Symbolic
Voting	One-shot quantitative statement of interest	Currency
Fixed Protocol	Back-and -front: symbolic and quantitative	Message and Protocol (Contract Net)
Conversation	Arbitrary interchange	Speech Acts (statement, request, commitment and command e.g. KQML)

Table 3. Negotiation protocols' classes.

The conclusions from this discussion of multi-agent negotiation are:

- Negotiation's issues provide many ways to explore in multi-agent researches.
- Negotiation processes are divided up to protocols, communication languages, standards and others (Beer et al., 1998).

2.5 Coalition in Multi-Agent

In "Cooperation structure" section, all types of the cooperation structures CATC, CCTA, CCTC and CGE include agent coalition; so the agent coalition is the most important part in multi-agent cooperation systems. Operations of coalition have to configure legacy or foreign systems (Allsoop et al., 2002).

Agent coalition is special type of agents, which concentrate on the coordination and the communication among agents to collaboratively accomplish tasks. For an example, the

power stations' extension, which costs a lot of money and time, can be solved if many power stations work together as form of agent coalition. In this example, agents in this system are owners of power stations, groups of customers and coordinators. The objective of the multi-agent system is to derive effective with gainful coalitions under the fair play practice subject to the constraints also requirements of power generation and transmission.

The modern industries needed a more efficient approach to facilitate a stable searching for new partners, coalition formation and a fair system used to identify the contribution from each participant (Yen et al., 1998). Some game theory models can be borrowed to improve the theoretical foundation for the multi-agent system.

The recent coalition operations are effected by many factors, such as data overload, starvation of information, labour-intensive information collection and coordination. The agent-based computing presents a new promising approach to effective coalition operations; since this approach embraces the coalition environment's open, heterogeneous diverse dispersed nature (Allsopp et al., 2002).

J. Yen et al. stated "The coalition formation in the multi-agent system is a hill climbing process." The payoff for each agent in the coalition formation should not be worse than the payoff of the previous method to solve same problem. However, such requirement may not be able to find the best solution for all the agents, it may arrested in local minimum (Yen et al., 1998).

Allsopp et al. stated "The coalition agents experiment (CoAX) is an international collaboration carried out under the auspices of DARPA's control of Agent-Based Systems program (CoABS)."

Some research hypotheses are suggested:

- Agents present a useful metaphor for dealing with the difficulty of real-world systems.
- An agent-based on command and control (C2) framework can support agile, robust coalition operations.
- Software agents can enable interoperability between incompatible systems.
- The CoABS Grid can rapidly integrate many different agents and systems, permitting rapid creation of virtual organizations.
- Domain policies can enforce coalition policies and compose agent relationships.
- Intelligent task and process management advance agent collaboration.
- Agent interoperability between distinct coalition controls systems can be improved by semantic web technology.

As an application on Multi-Agent coalition; CoAX team has built a software agent tested and based on the control of Agent-Based Systems program Grid. In this project, a demonstration series were being conducting within increasing difficulty in a stylized yet practical peace-enforcement scenario situated in Binni, a fictitious African state. These demonstration series use agent technologies to compose a coalition command system for intelligence gathering; visualization, campaign, battle, mission planning and execution (Allsopp et al., 2002).

Another application on Multi-Agent coalition; six-bus system of electric distribution has been used to illustrate the planning process of network expansion. The limits of power transmission and power generation are shown on the figure (2). The heuristic approach used to rank the possible locations to add new lines to an existing system. The heuristic approach is a quadratic linear programming problem, which used to identify the degree of solution feasibility (Yen et al., 1998).

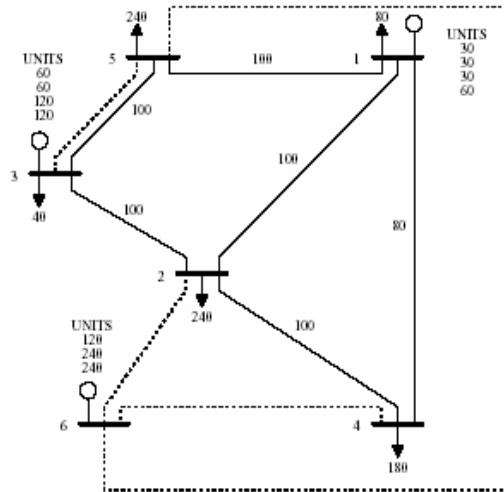


Fig. 2. Six-bus system of electric distribution (Yen et al., 1998).

The conclusions from these applications are:

1. The multi-agent is capable of making decisions for coalition formation and cost allocation, with very limited coordination and synchronization provided by the coordinator, in a fully decentralized environment.
2. Users do not need to rent dedicated lines to support the communications, because this implement is executed on the Internet.
3. Multi-agent systems can easily be applied to solve the problems where formation of coalition is essential and the environment is geographically dispersed, such as, global logistics planning or coalition formation of shipping and transportation firms.

3. Applications of Multi-Agent

There are many applications of Multi-Agent systems in all fields. The popular applications are information gathering, mobile agent cooperation and sensor information and communication. The overview of these applications will be presented in following sections.

3.1 Information Gathering

The most important issue in the applications related to information gathering is making an informed decision from a huge amount of information. One of the most recent researches in this area is a system of information gathering (Li et al., 2002).

This system consists of two agents:

1. Query processing agent.
2. Information filtering agent.

The objectives of information gathering system are:

1. Retrieving relevant information from World Wide Web (WWW).
2. Query the relevant information.
3. Filtering information from vast of information.

The structure of this system is shown in figure (3).

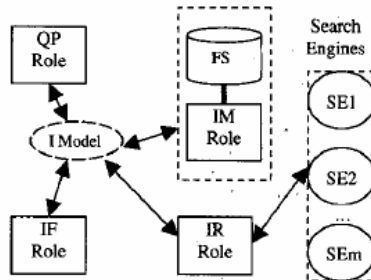


Fig. 3. Information Gathering System (Li et al., 2002).

The information collected from WWW is multi-modal, unorganized and distributed on collections overall world; therefore gathering right information is very difficult. Information of WWW is uncertain. WWW search engines provide users with overload sites or document for certain keywords or expressions. WWW services and documents are extremely changing. These system achievements are depended on the training documents; because documents contain uncertain and unorganized information.

3.2 Mobile Agent cooperation

Flexibility and efficiency in solutions of distributed systems are the mean objectives of Multi-Mobile Agents (Multi-MA). There are many research studies are presented in this area.

A theoretic model has been suggested to solve some key problems such as fuzzy belief composing and contradiction coordinating. In addition, new ideas of Multi-Mobile Agents on fuzzy knowledge exchange and representation are presented.

Mobile node and mobile agent form this theoretic model. In addition, this model concerns on two important topics the architecture and agent's behavioral interesting aspects. The diagram of this model is shown in figure (4):

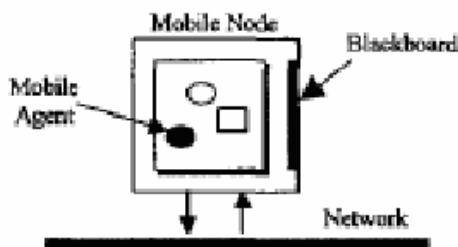


Fig. 4. The reference model of multi-MAs cooperating System (Miniy 2000).

The black board in the model allows agents' interactions; when agent passed the node the information on the blackboard is updated. This information influences on agent's behaviours in the same time. The basic elements of agent are Run-time behavioural code, roaming trial, belief (fuzzy knowledge) and other parameters.

There are two types of agent's migration behaviours:

1. Passive migration; when agent is controlled by other agents.
2. Active migration; when agent can decide by itself and start its movement according to suitable condition (Minyi et al. 2000).

Another application on Multi-Mobile Agents is the Multi-hop ad-hoc wireless Network (MANET). The MANET is defined as "a network architecture where each mobile host is treated as a router and makes a peer-to-peer communication without any base-stations" (Onishi et al., 2001).

The advantages of using mobile agents for control packets are:

1. Instillation new routing system is easy, because the process to instillation is "collect old routing agents and release new agents over the network". This easiness facilitates routing algorithms to adapt themselves to valuable cases.
2. Several services of routing are locally embedded in agents. In addition, in supported agents the heterogeneous networks can be connected together.
3. The used resources will be released after agents left.

In contract, the disadvantages of using mobile agents for control packets are:

1. Malicious hosts can easily attack agents.
2. Deploying a mobile-agent system, which effectively protects a machine against malicious agent by restricting resources, is possible.
3. Agents' codes are usually written in relatively slow interpreted languages and slightly gain weight (Onishi et al., 2001).

3.3 Sensor information and communication

The sensors are used to measure signals, temperature, pressure and other variables. Multi-agent systems have been used in this field to achieve reliable, processed, accurate and accepted measurements through merge different methods (agents) that used to measure one or more variables have common relationships.

Complex maintenance-type assignments for mobile Jet turbofan engine have been executed in coordinated style through cooperative of simple autonomous mobile sensor platforms. These platforms have been developed from a general archetype. The mains platforms of this system are:

1. Original platform collects decision, controls and returns sound, video and other measured variables.
2. Network platform connects all platforms and provides a communication between all platforms.

This system is based on swarm algorithms; which provide architecture scalability, units' flexibility and robustness through redundancy and simplicity. Tethers and necessity for an efficient representation of engine surface are essential keys for this model; since tethers create restriction on agents' movement and need a path planning, and the necessity for an efficient representation of engine is due to limited memory capacity of agents (Litt et al., 2003).

A multi-agent real-time simulation framework allows high-fidelity virtual sensor models, which incorporate in hardware-in-the-loop (HIL) experiments. Multi-agent real-time framework results are:

1. Full control of the environment.
2. Reproducibility.
3. Easy merge of real/virtual components in model experiment.

Multi-agent real-time framework has been illustrated through a laser range finder sensor and a pre-crash control solution (Papp et al., 2003)

4. Pursuit Problem Demonstration

In this section the pursuit problem will be used to demonstrate the working of multi agent systems as multi agent robots.

Pursuit problem is one of the punch problems in robotics fields as it deals with dynamic targets going in unknown directions. This section takes the notion of pursuit and implements different algorithms for acquiring a dynamic target in unknown environment, the algorithms test without using any localization devices such as the GPS devices or laser reflectors (Beacons). The agents are work in the same environment, negotiate together to decide on the way the target is acquired. The robots will roam in the environment and preserve their formations until the dynamic target is spotted. When the target is spotted the robots will decide who will acquire the target and will use its algorithm to acquire the target.

Robots can be used to carry out many tasks with a wide range of complexity: from simple tasks that can be performed by one agent, to other complicated tasks that need a group of robots collaborating together to reach their goal. The multiagents implementation in the domain of Pursuit problem is concerned with designing and implementing algorithms that allow robots to cooperate together to catch a hider robot. Negotiations and other anti flocking algorithms will take place to make the searching algorithms of different groups more effective. Since our work considering using behaviors based robotics, robots control can benefit from animal behavior studies. Robotics can draw from these behaviors models to make similar forms of behavior in machines. To Study behavior-based robotics it is essential to start with the biological behavior of animals in terms of ethology, and learn the relationship between animal behavior and multiagents behaviors. The study of ethology is very important for multiagents robotic systems in terms of display behavior (which involves the information signaling by changes in activity). These displays are usually generated by fixed action patterns and may be electrical, visible or audible. The displays include color changing in fish, birdsong, leg waiving in spiders, etc. These displays may benefit some activities such as escaping a predator. [Arkin, 1998]. The "ecological niche" is one of the most important concepts taken from the study of ethology that benefit behavior-based robotics. The ecological niche is the status of an animal in its own community in terms of food and surviving. A successful robotic system is a system that is autonomous and can compete with its environmental surroundings, that's why the ecological niche is very important to robotics. A system that does not find a stable niche will be unsuccessful. Robots must compete with their natural world with static or dynamic objects and the niche allows them to survive with their competitors. When an effective robotic system is to be designed it must be targeted towards some niche. [Arkin, 1998]

4.1 Pursuit in robotics

The contest of pursuit is among the most widespread, challenging, and important optimization problems that face mobile agents, and represents some of the most important

potential applications for robots. In a typical contest of this sort, a predator chases a prey animal around until the prey is captured.

Pursuit contests are usually difficult to handle, because they deal with dynamic targets. Agents that pursue must maintain the sensory information of both the physical environment and the hostile opponent. An effective pursuit may often require prediction and “mind-reading”. With this problem, recognizing other robots is very important when designing multiagent systems. A robot has to distinguish other robots from many environmental features and also when working with teams recognizing a hider robot from a seeker robot is very important as well. [Arkin, 1998]

The most commonly used techniques adapted to deal with pursuits are:

1. *Classical calculus of Variations and Optimal Control Technique*: This technique provides a strong tool of analysis and design. This technique has the advantage of giving a real time solution, whenever it exists, since the system and the constraints are represented by a set of differential equations. However this technique has not been widely used in pursuits because it gets complicated with the increase in the number of players.
2. *Dynamic Programming*: A very efficient technique that mainly deals with discrete systems with a value function that needs to be optimized.
3. *Reinforced Machine Learning*: Reinforcement learning is a technique of learning how to map situations to actions.

The learner is not told which actions to take, instead he must decide on the rewarding actions by trying them. Actions may not affect only the immediate reward, but also the next state and, through that, all subsequent rewards. The two characteristics of trial and error search and delayed reward, are the most important features of reinforcement learning. The basic idea is simply to capture the most important aspects of the real problem facing a learning agent interacting with its environment to achieve a goal. This goal is related to the state of the environment. Three aspects are included: sensation, action, and goal.

Many researchers have tackle the pursuit problem, Cassimatis, Trafton, Schultz, Bugajska and Adams from the Naval Research Laboratory in Washington [Staugaard, 2002], take the game hide and seek as an example in their research about cognitive modelling techniques, and address some issues that have to be taken in consideration when developing pursuit algorithms. To find their targets they have to be able to identify other objects and agents, to plan their path and move towards their target trying to avoid the obstacles in their way. In other words any information agents gather perceptually will be valuable when seeking and navigating, and the more efficiently they navigate the better they will be at the game. [Staugaard, 2002]

They give an illustrated example of temporal reasoning about the robot rolls behind an occluding screen and then after that a robot that looks the same comes out. Then someone puts a barrier behind the screen. [Staugaard, 2002]

Many issues from the logical study of intelligence arise here: what can be assumed and why; how can an assumption be falsified and when is there more than one explanation for an event, and how do we choose between them. The authors argue these issues of reasoning have to be taken in consideration whenever we try to build an effective hide-and-seek or a pursuit system. [Staugaard, 2002]

The Authors argue that any system uses hide and seek has to revise provisional beliefs and inferences that followed from it. So seekers must engage in probabilistic interferences in order to keep track of the hiders efficiently. [Staugaard, 2002]

Other work has done with one robot that seeking robot and there are several hiding robots in an arena. The seeking robot will initially wait a certain time until the hiding robots find their places to hide on the arena (the arena contains walls and obstacles that make it difficult for the seeker robot to find the other robots). These hiding robots will have the capability to move as soon as the seeking robot detects them. [Kranz, et al 2006]. The difference in this game is that when the robot detects other robots it begins shooting them with laser not actually chasing the robots. The hiding robots will try to move away to prevent themselves from being hit by the laser. If the hiding robot finds the seeking robot before the seeker finds it, it can attempt to evade the seeker.

The other hide and seek algorithm. A human hides a vehicle anywhere in the hiding area. The seeker will search the hiding area for the opponent and attempt to touch it. A human will try to drive the vehicle away once it is spotted by the seeker. If the robot loses the vehicle during the chase, he should look for it behind nearby objects to see if it is hiding there. The game will be over when the robot touches its opponent [Coulter, 1992]

One of the pursuit algorithms is a tracking algorithm that is based on calculating the curvature that will move a robot from its position to another goal position. First a goal position has to be chosen some distance ahead of the robot on the path. This algorithm is derived from the way humans drive. A human tend to look some distance in front of the car and drive towards that spot. The look ahead distance changes as a human being drive to reflect the twist of the windings of the road. [Staugaard, 2002]

4.2 Formation for pursuit problem

Formation is very important in the field of multi-agents robotics, in order for the robots to roam together to complete a certain tasks they need to form a chain and maintain this formation. This research only concentrates on how to achieve formation coordination without providing the robots with global knowledge of other robots' positions or headings.

Formations can be done in different strategies ranging from simple, behaviors-based, ones to more involved ones relying, on global knowledge of the environment, typically a global coordinate system or maybe knowledge of other robots' positions in the environment and their headings. The first category is characterized by robustness but there is a lack of guarantees that the wanted formation will actually emerge; the second category is characterized by reliability and efficiency but there is a need for the global knowledge of the environment and sometimes complex computation. [Riley et al. 2005]. In [Watson, 1925] three principles of formation control are identified and categorized depend on references.

In [Riley et al. 2005] an algorithm was presented using IDs assigned to the robots, in this algorithm, the robots can change in group size and also switch between formations and avoid obstacles.

Analyzing of the different types of geometric formations that multi-agents can do and they also agree that a geometric formation consists of three main parts; Conductor: which is the robot at the head of a group in a formation, Leader: the leader is the agent who guides the one that follows it, and Follower: All the agents in a formation are followers except the conductor [Fredslund & Mataric', 2002] and [Balch & Arkin, 1998].

The main objective of this paper is to design, implement and test algorithms that allow many agents to cooperate together in order to catch a hider using agent of sub-team. The

sub-teams are multi-agents that communicate and collaborate together in order to reach their goal. Each sub-team of robots has its own pursuit algorithm that is behavior-based and uses some mathematical manipulations to make it efficient. The robots also make use of certain formations to accomplish their task.

4.3 Algorithms-behaviors design

Our systems are dealing with four sub-teams of agents that have their own way to chase the hiding agent. These sub-teams will collaborate together to make their search efficient. Our work with assumptions that the Environment is unknown and none of the robots have any previous information about the environment, No localization devices are used, agents are identical, and the work is only concerned about pursuit algorithms, no algorithms were implemented to the hiding agent.

All the agents have obstacles avoiding behaviors which make each agent capable to detect the obstacles using the sonar. This algorithm will be executed whenever an obstacle is detected with less than 0.4 meters in front of the robots. The sonar beams is covered area by the ranges up to 6 m.

4.3.1 Attacking the target methods

These behaviors and it own algorithms have been design to give the agents capability to deal with target.

4.3.2 Roams the environment

This algorithm is done with one agent that roams the environment and avoiding obstacle. When it sees the hiding agent it starts attacking it. The hiding agent is recognized by its color. This algorithm is derived from the way humans think about attending a desired objective. A human tend to look at the spot where he wants to go and then walk straight towards that spot if there are no obstacles in the way.

Whenever the pursuer in this algorithm sees an agent with the green color it will know that this is the opponent and it will start attacking. First it will speed up from the normal speed to the attacking speed (from 0.4 meters/s to 0.8 meters/s) and it uses the information from the camera to adjust the heading to the target by calculating angle difference between the target and the heading of the robot The information about the angle returned from the camera is between 0 and 160 degrees as illustrated in the figure (5) (the actual heading of the robot is at 80 degrees). At turning speed and the linear speed of 0.8 meters per second the agent will keep on chasing the target until it is caught.

4.3.4 Heading with considering the speed of the hider

In this algorithm the pursuer tries to detect the speed and the heading of the hider relative to its own position and heading without using any localization device. After it detects the speed and the heading of the hider it tries to predict where it will be after a certain time and then it will go straight to that meeting point to catch the hiding agent.

In order to calculate the heading and the speed of the opponent, the pursuer will have to take the distance and the angle between him and the agent targeted. Then the pursuer will have to stop for exactly one second before taking another reading of the distance and the angle between its heading and the target. This can happen in two situations, the first is when the hider heading away from the pursuer and the second is when the hider is heading

towards the pursuer. These two situations can also happen in two different ways: when the hider is heading to the left of the pursuer and when the hider is heading to the right of the pursuer. The first situation are illustrated in the figure (6).



Fig. 5. Camera heading robot-target angle. Fig. 6. Speed detection.

In figure (6) we have:

P: position of the pursuer relative to the other agent.	R ₁ : the range between the pursuer and the hider before the 1 second interval.
R ₂ : the range between the pursuer and the hider after the 1 second interval.	H ₁ : Position of the other agent as soon as it is spotted by the pursuer (H stands for hider).
H ₂ : position of the other agent after 1 second from when it was first sighted. (Note that the pursuer during the 1 second does not move)	

Let X₁ be the angle returned by the camera just before the 1 second stop and X₂ be the same angle just after the 1 second stop

Let P₁ be the projection of H₁ on R₂.

The angle alpha in the figure above is the absolute value of X₁ - X₂:

$\alpha = X_1 - X_2 $	$PP_1 = R_1 \times \cos\alpha$	$H_1P_1 = R_1 \times \sin\alpha$
$H_2P_1 = R_2 - PP_1$	$H_1H_2 = \sqrt{H_1P_1^2 + H_2P_1^2}$	

Having the length of H₁H₂ which is the length of the track of the hider calculated over 1 second we can conclude that the other robot has a speed of H₁H₂ meters/s. We also know that the hiding agent will go 3 x H₁H₂ meters in 3 second, or 5 x H₁H₂ in 5 seconds. Assuming that the hider has a constant speed and it only goes straight until it faces an obstacle the pursuer is now able to predict how far the hider will go in a certain amount of time. Now to meet the hider at the defined point (on figure (7) is H₃) after 3 or 5 seconds the pursuer will still have to know the angle that it has to turn from its original position as well as the distance to that meeting point. This is illustrated in figure (7) which is an extension to figure (6) with the addition of H₂H₃ which is the calculated distance that the robot will go in 3 seconds after the 1 second stop interval.

The purpose of this diagram is to calculate PH₃ which is the distance that the pursuer will go to get to the meeting point with the hider. It will also calculate the angle ∠H₁PH₃ which will help determining the turning angle of the pursuer however this angle is not the angle that the pursuer has to turn in order to get to his destination, because with absence of a localization device the original heading of the pursuer is unknown.

Let P_2 be the projection of P on H_1H_3 . From this graph we can conclude

$\gamma = \tan^{-1} H_2P_1/H_1P_1.$	$\beta = \tan^{-1} PP_1/H_1P_1.$
$P_2P = \sin(P_2H_1P) \times R_1$	$P_2H_1 = \cos(P_2H_1P) \times R_1$
$H_1H_3 = 4 \times H_1H_2 ((1\text{second} + \beta\text{seconds}) \times H_1H_2 \text{ meters/second})$	$P_2H_3 = P_2H_1 + H_1H_3 = \cos(P_2H_1P) \times R_1 + 4 \times H_1H_2.$
Angle $\angle P_2H_1P = 180 - (\gamma + \beta).$	Angle $H_1PP_2 = \sin^{-1} (P_2H_1/PH_1)$
Angle $H_3PP_2 = \tan^{-1} (P_2H_3/P_2P)$	
Angle $H_3PH_1 = H_3PP_2 - H_1PP_2$	$H_3P = P_2H_3 / \sin H_3SP_2$

Having determined H_3P (which is the distance that the robot have to go to get to its objective after 3 seconds we can calculate the linear speed of the pursuer that will make him get there after 3 seconds: $\text{speed} = H_3P/3 \text{ m/s}.$

As mentioned above the camera gives the angle of the line of sight of the hider, this angle is between 0 and 160 degrees with 80 being the heading of the pursuer
 With the absence of a localization device the turning angle of the robot has to be calculated relatively to its original heading when it stopped and waited for 1 second. This can be done with the use of the value of the angle H_3PH_1 calculated earlier and the value of X_1 and X_2 (with X_1 being the angle returned by the camera just before the 1 second stop and X_2 be the same angle just after the 1 second stop).
 To calculate the angle that the robot has to turn.

In the case where the hider is going right (that is $X_2 > X_1$):	In the case where the hider is going left (that is $X_2 < X_1$):
If $X_1 < 80$ then the hider's initial position before the 1 second stop is $(80 - X_1)$ degrees to the left of the pursuer. In this case the turning angle would be $80 - (H_3PH_1 + X_1).$	If $X_1 < 80$ then the hider's initial position before the 1 second stop is $(80 - X_1)$ degrees to the left of the pursuer. In this case the turning angle would be $(80 - X_1) + H_3PH_1.$
If $X_1 > 80$ then the hider's initial position before the 1 second stop is $(X_1 - 80)$ degrees to the right of the pursuer. In this case the turning angle would be $- ((X_1 - 80) + H_3PH_1)$ (note that the - is for the anti-clockwise turning)	If $X_1 > 80$ then the hider's initial position before the 1 second stop is $(X_1 - 80)$ degrees to the right of the pursuer. In this case the turning angle would be $X_1 - 80 - H_3PH_1$

To get the turning rate this angle will be divided by 3 seconds.
 The second situation is when the hider is heading towards the pursuer: With this situations the graphs and the calculations are different, however the purpose of these graphs is the same as in the previous situation, which is to find the distance that the pursuer have to go to meet the hider on his trajectory, the linear speed and the turning speed. The calculations for the second situation are illustrated in the figure (8)

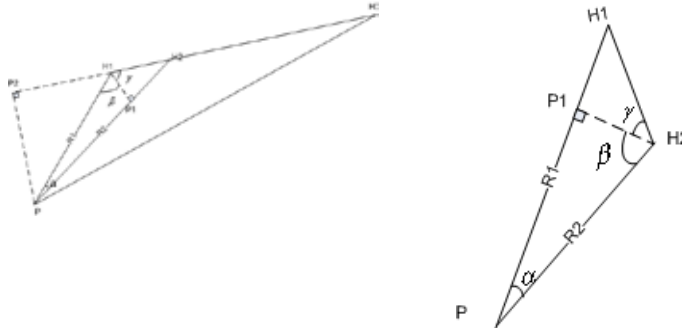


Fig. 7. Linear speed and turning speed calculations. Fig. 8. Speed detection 2.

Same concept as before, that the hiding agent will go $3 \times H_1H_2$ meters in 3 second, or $5 \times H_1H_2$ in 5 seconds.

4.4 Cooperate agents and formation

In this algorithm a group of two robots will cooperate together in order to catch the hider. As soon as the robots see each other they will roam the environment together forming a linear formation: a leader and a follower. Then when the hider is sighted the formation will break and the robots will attack the hider one from the left and one from the right. The main benefits of this algorithm is that the attacking robots do not have to worry about the direction of the hider since they attack from both sides.

The formation coordination between the two robots is achieved without providing the robots with global knowledge of other robots' positions or headings. The leader will not communicate to the follower and the follower is the only one responsible of maintaining the formation.

There are two phases in the formation, first starting the formation and then maintaining the formation. The following is done by recognizing the leader's color and then keeping the right angle with the leader. The follower keeps in line with the leader he by adjusting its heading to match the leader's heading. At the same time the follower is responsible of keeping a certain distance to the leader and changing its speed accordingly as in figure (9). The formation algorithm is done in a way that makes it very easy for the group to avoid obstacles together. Once the target is sighted after the formation is done the leader will follow the target straight until it is 5 meters from it, then it will speed up to 0.800 meters/second and it will check if the follower is aligned with it (within 10 degrees difference) then it will attack from the right hand side (this is done with the use of the information about the hider's line of sight's angle returned by the camera). If the follower is not aligned with the leader (this can happen if the leader has recently faced an obstacle and the follower is still recovering from that) then the leader will wait until the follower fixes its heading before attacking. Then the leader will attack from the right by keeping an angle of 40 degrees to the leader, it will also signal the follower to attack as well. At this time the follower will attack from the left by keeping an angle of 40 degrees from his target to the left. When any of the attacker is at a distance of less than 1.5 meters to the hider it will fix its angle and attack straight until the hider is caught. This is illustrated in figure (10).

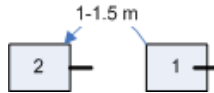


Fig. 9. Following algorithm.



Fig. 10. Formation of reaching the target.

The other type of formation are three robots forming a linear formation. As soon as the target is sighted, the team will chase the hider and then change the formation to a circle formation and surround the target inside the circle.

In this algorithm we have three robots doing the linear formation as shown in figure (11). In this formation the conductor, number 1, which is at the same time a leader to agent number 2. Agent number 2 is a follower to number 1 and a leader to number 3. The conductor is responsible of leading the group and the other two will follow in the same way described in the previous algorithm.

The conductor will decide when to switch from the linear formation to the circle formation. When this is done the robots will have to follow the conductor respectively by maintaining the angle, the speed and the distances calculated depending on figure (12)

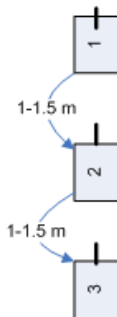


Fig. 11. Linear formation.

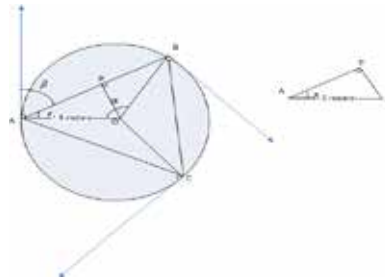


Fig. 12. Circle formation.

4.5 Anti-Flocking

The anti-flocking algorithm used in this work is based on the color detection. Every subteam in this design has its distinctive color, so basically every time the agent in the first and the second attacking methods sees a color other than its color and the target's color a random number between -20 and 20 will be generated and the agent will make this number as its turning speed until the other team or robot is out of sight. This concept is also the same for the leaders in the formations in attacking. In this way the robots will try to stay out of sight of each other and thus searching different areas in the environment. The Anti flocking behavior will only be effective whenever the target is out of sight of the team that performs the anti-flocking behavior. However this antiflocking algorithm has other benefits as well: It prevents the teams from coming in the way of each other; it also prevents blocking the view and preventing other teams from seeing the target as well as helps clearing the way of a team attacking the target.

4.6 Negotiations

The negotiations will take place to decide who will chase the target as only one team can chase the target at one time.

If a team (e.g. Team1) sees the target for the first time it will signal the other teams that the target is being chased and it will also keep on sending its distance to target 10 times a second. Then if another team (e.g. Team2) sees the target while being chased it checks its own distance to target. If that distance is less than the distance of team1 to the target it will signal the other teams that the chase is taken over by team2 this way team1 will know that the target is closer to another team and it will stop chasing leaving the chase to the other team. So there are two messages to be sent between the subteams, the first is the distance to target when target is sighted, and the second is the signal to tell the other teams that the chase is done by this team.

4.7 Implementation and simulation results

This paper presents the acquiring of a dynamic target by multi-agents collaborating. The environment is considered to be unknown; no localization system is used. The heading and the speed of the target are considered unknown. Other problems are the environment is quite small and have obstacles, the robots will have to search different parts of the environment and avoid flocking together, and the communication between the robots should be minimized. The simulation of the design is done using a simulation Stage and player program and using Pioneer Robots. The final implementation including 8 agents.

Most of the simulation results show the screen capture of two windows Stage/ player and G2 taken at the same time when the robots are running inside the environment. This way the full track of the robots is shown and the robot's location at the time when the capture is taken can be seen as well.

4.7.1 Exploration and obstacle avoidance

The first feature tested in the system is the basic roaming of the robots inside the environment, and the obstacle avoidance whenever an obstacle comes up. To test both static and dynamic obstacle avoidance, two robots were used and located inside facing each other figure (13) shows the robots just after meet each other:

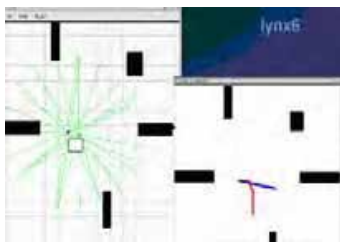


Fig. 13. Obstacle avoidance Phase 1.

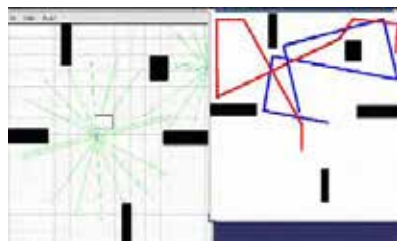


Fig. 14. Obstacle avoidance Phase 2.

It can be seen how the robot with the red path avoided the other robot when it came within its sonar range of avoidance (0.4 meters). The robots run further in figure (14) : and how they avoid different static obstacles.

The degree of the robot to turn left or right depending on the range calculations explained in section, then as soon as the obstacle is avoided the robot goes back to the normal exploration of the environment. The problem of the robot trapped in corner problems or the obstacles interact to block the robot had been eliminated, as shown in the in figure (14) by introducing a second turn of 30 degrees/second as soon as the obstacle is detected without checking the ranges within that second. So the robot turns for about 1 second then it checks on the obstacle to see if it s still there. As it can be seen from these simulation captures, the behaviors of obstacle avoidance works well in this environment and the robots are avoiding the dynamic and static obstacles successfully.

4.7.2 Attacking Behaviours

The test of the attacking behaviors, quit few simulation test have done, one is when the target is sighted the agent, it will raise its speed and follow the target straight as shown in figure (15) , the target starting from the left side and attacker starting from the right side of the screen and heading towards an obstacle. As it tries to avoid the obstacle to the right it sees the target then it raises the speed and it starts heading towards the target. In the a of the figure (15) the red line and the green line being about the same length, however in phase two it can be seen that the red line is longer than the green line even though they started at the same time. This is due to the increase in speed of the attacker after it spots the target. The curve in the red line of figure (15) illustrates the path of the attacker when adjusting its heading towards the target. In other attacking scenario, the features of formation starting, formation maintaining, and attacking were all tested. In this algorithm the team of two robots start the linear formation as soon as they spot each other, it starts the formation by adjust its speed accordingly until the formation is achieved and then roam the environment maintaining the formation when avoiding obstacles

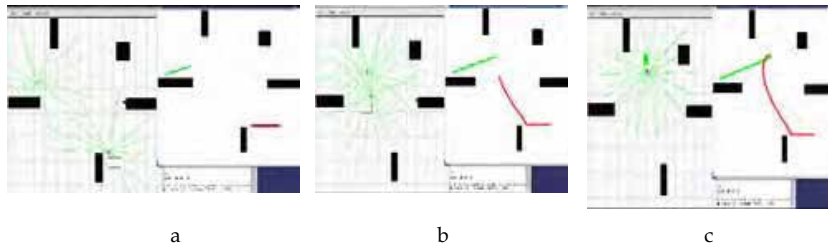


Fig. 15. Attacking behavior for one agent(robot).

As soon as the group spots the target it is advance towards until the leader is 5 meters away and signal the follower to break the formation, both robots attack the target one from the left and the other from the right as illustrate in figure (15).

The robots are placed initially in the environment where the one in green is the target and it starts from the top. The two attackers with the blue and black colors start from the bottom of the screen initially 1 meter apart so the formation can be seen before the attack starts.

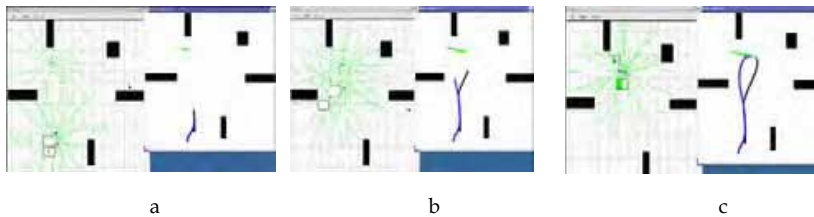


Fig. 16. Attacking behavior for one agent(robot) with formation.

In a of the figure (16), the robot is black starts the formation by following the robot in blue and then the formation is kept. At the same time the robot in blue spots the target and starts heading towards that target, while the robot in black maintains the formation. when the leader gets to 5 meters from the green target that is heading towards the bottom, It signals the other robot to break the formation and at the same time it starts attacking from the right, while the follower starts attacking from the left as shown in b of the figure (16), and the formation is broken. The final phase of the attack is illustrated in c after the formation is broken the robots increase their speed and attack from both sides until they are 1.5 meters away from the target, the robots attack straight and this way the target is acquired from both sides.

The other test have done for a team of three robots start the linear formation as soon as they spot each other, and then roam the environment maintaining the formation when avoiding obstacles.

As soon as the group spots the target it is supposed to advance towards until the leader is 1.5 meters away and then the leader should signal the other 2 followers to break the initial formation and start another formation that surrounds the target, then the three robots will circle the target and trap him inside the circle by maintaining the exact distances between each other and changing their speed accordingly as figure (17) illustrated.

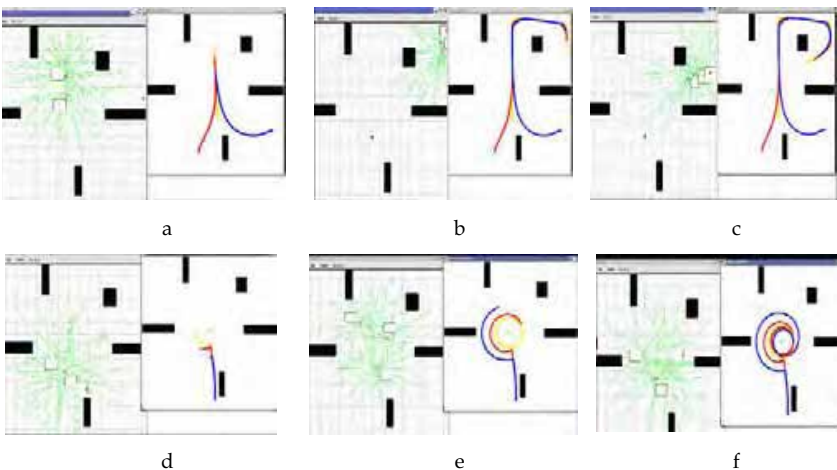


Fig. 17. Demonstration of pursuit problem.

The three robots with the current speeds in the circle formation can lock up a dynamic target of up to a speed of 0.300 meters/second. With that speed whenever the target tries to get out of the circle, it is blocked by one of the attackers and it has to avoid them and go back inside the circle. At worst case if the target's speed is slightly more than 0.300 the target is caught by one of the attackers doing the circle formation, but it can never escape.

5. References

- Allsopp, D. N.; Beautement, P.; Kirton, M.; Bradshaw, J.M.; Suri, N.; Durfee, E. H.; Knoblock, C. A.; Tate, A. & Thompson, C. W. (2002). Coalition Agents Experiment: Multiagent Cooperation in International Coalitions. *IEEE Intelligent Systems*, Vol., 17, (May/June 2002) page numbers (26 - 35)
- Arenas, A. E. & Sanabria, G. B.(2003), Modelling Intelligent Agents for Organisational Memories, In: *Lecture Notes in Computer Science*, Vol., 2773, page numbers (430 - 437), Springer, 0302-9743, Berlin, Germany.
- Arkin, R. C.(1998). *Behavior-based robotics*, MIT Press, 0262011654, London, England Chapters 1,2,3 and 9.
- Bainbridge, L. (1993). The change in concepts needed to account for human behavior in complex dynamic tasks, *Proceedings of IEEE Systems Man and Cybernetics Conference*, Vol. 1, pp. 126 - 131, Westminster hotel, October 1993, IEEE, Le Touquet - France.
- Balch, T. & Arkin, R. C. (1998). Behavior-based Formation Control for Multi-robot Teams. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, (December 1998) pp. (926-939), 1042-296
- Beer, M.; D'Inverno, M.; Luck, M.; Jennings, N.; Preist, C. & Schroeder, M. (1999). Negotiation in Multi-Agent Systems, *The Knowledge Engineering Review*, Vol. 14, Issue 3, (Sep 1999) page numbers (285-289), ISSN: 0269-8889.
- Cassimatis, N. L.; J. Trafton, G.; Schultz, A. C.; Bugajska, M. D. & Adams W. (2002), A Task Domain for Combining and Evaluating Robotics and Cognitive Modeling Techniques, Measuring the Performance and Intelligence of Systems: Proceedings of the 2002 Performance Metrics for Intelligent Systems (PerMIS) Workshop, Washington DC, 2002
- Changhong, L.; Minqiang, L.; & Jisong, K.; (2002). Cooperation Structure of Multi-agent and Algorithms, *Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems*, pp. 303-307, ISBN:0-7695-1733-1, September 2002, IEEE, Divnomorskoe, Russia.
- Coulter, R. C., (1992). *Implementation of the pure pursuit path tracking algorithm*, The Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania.
- D'Inverno, M.; Luck, M. & Wooldridge, M. (1997). Cooperation Structures, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 600-605, August 1997, Nagoya, Japan.
- Floreano, D.; Nolfi, S. & Mondada, F. (1998). Competitive co-evolutionary robotics: from theory to practice, *Proceedings of the Fifth International Conference on Simulation of Adaptive Behaviour on From animals to animates 5*, pp. 515 - 524, ISBN:0-262-66144-6, University of Zurich, Zurich, Switzerland, 1998, Mit Press, Cambridge.
- Fredslund, J. & Mataric', M. J. (2002). A General, Local Algorithm for Robot Formations. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, Oct 2002. pp. (837-846), 1042-296

- Gustafson, D. A.; & Matson, E. (2003). Taxonomy of Cooperative Robotic Systems, *Proceeding of 2003 IEEE International Conference on Systems, Man and Cybernetics*, pp.1141-1146, ISBN: 0-7803-7952-7, Crystal City Hyatt Regency, October 2003, IEEE, Washington DC.
- Jones, P. M.; & Mitchell, C. M. (1991). Human-machine cooperative interaction in the supervisory control of a complex dynamic system, *Proceedings of the 1991 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 2, pp.1301-1306, Omni Charlottesville hotel and the University of Virginia, Charlottesville, Virginia, October 1991, IEEE.
- Jones, P. M.; & Mitchell, C. M. (1995). Human-computer cooperative problem solving: Theory, design, and evaluation of an intelligent associate system for supervisory control, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No.7, (May 1995) page numbers (1039-1053), ISSN: 1094-6977.
- Jones, P.M.; & Jacobs, J. L. (2000). Cooperative Problem Solving in Human-Machine Systems: Theory, Models and Intelligent Associate Systems, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 30, No.4, (November 2000) page numbers (397-407), ISSN: 1094-6977.
- Kranz, M.; Rusu, R. B.; Maldonado, A.; Beetz, M. & Schmidt A. (2006). A Player/Stage System for Context-Aware Intelligent Environments. To appear in *Proceedings of the System Support for Ubiquitous Computing Workshop (UbiSys 2006)*, at the 8th Annual Conference on Ubiquitous Computing (UbiComp 2006), Orange County, California, September 2006.
- Li, C. S.; Zhang, C. & Zhang, Z. L. (2002). An Agent-Based Intelligent System for Information Gathering from World Wide Web Environment, *Proceedings of the 2002 International Conference on Machine Learning and Cybernetics*, Volume 4, pp.1852-1857, ISBN: 0-7803-7508-4, Prime Hotel, Beijing, China, November 2002, IEEE.
- Litt, J.; Wong, E.; Krasowski M. & Greer L., (2003). Cooperative Multi-Agent Mobile Sensor Platforms for Jet Engine Inspection--- Concept and Implementation, *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems. KIMAS'03: Modeling, Exploration, and Engineering*, October 2003 pp. 716-721, ISBN: 0-7803-7958-6, Royal Sonesta Hotel, Cambridge, MA, USA. Minyi,
- Shaowen, W.; Y. & Mingtain, Z. (2000). Cooperative Mobile Agents in Dynamic Network Environment, *Proceedings of the 36th International Conference on Technology of Object-Oriented Languages and Systems*, pp.162-167, China, November 2000, IEEE
- Mitchell, C. M. (1999). Model-based design of human interaction with complex systems In: *Handbook of Systems Engineering and Management*, Sage, A. P. & Rouse, W. B., page numbers (745-810), John Wiley & Sons, ISBN: 0-471-15405-9, Canada.
- Nitschke, G. (2003). Co-evolution of cooperation in a Pursuit Evasion Game, *Proceedings of the IEEE/RSJ International Conference on intelligent Robots and Systems*, Volume 2, pp. 2037-2042, ISBN 0-7803-7860-1, Las Vegas, Nevada, October 2003, IEEE.
- Onishi, R.; Yamaguchi, S.; Morino, H.; Aida, H. & Saito, T. (2001). The Multi-agent System for Dynamic Network Routing, *Proceedings of the 5th International Symposium on Autonomous Decentralized Systems*, pp. 375-382, ISBN 0-7695-1065-5, Dallas, Texas, March 2001, IEEE.
- Papp, Z.; Labibes, K.; Thean, A. H. C. & Eikm, G. V. (2003). Multi-Agent Based HIL Simulator with High Fidelity Virtual Sensors, *Proceedings of the IEEE IV2003 Intelligent Vehicles Symposium*, pp. 213 - 218, ISBN: 0-7803-7848-2, Hilton hotel Colummbus, Ohio, USA, June 2003, IEEE.

- Parunak, H. V. D. (1999). Characterizing Multi-Agent Negotiation, *Proceeding of the Workshop on Multi-agent Systems, Negotiation Science and Technology: Opportunities and pitfalls*, Centre for Electronic Commerce, ERIM.
- Staugaard, A. C. (2002) *Structured Object-Oriented Problem Solving Using C++*, Prentice-Hall, 0130284513, New Jersey.
- Watson, J.B. (1924/1925). *Behaviorism*. New York: People's Institute Publishing Company; and Watson, J.B. & McDougall, W. (1928). *The Battle of Behaviorism*. London: Kegan Paul, Trench, Trubner & Co.
- Yen, J.; Yan, Y. H.; Wang, B. J. & Sin, P. K. (1998). Multi-agent Coalition Formation in Power Transmission Planning, *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, Volume 4, pp. 433 - 443, ISBN: 0-8186-8255-8, The University of Hawaii, January 1998, IEEE.

Grid technologies for Intelligent Autonomous Robot Swarms

Fabio P. Bonsignorio
Heron srl
Italy

1. Introduction

This chapter shows how grid computing characteristics enable the deploying of huge networks of comparatively intelligent robotic agents.

It describes and discusses the concept of a grid based distributed multiagent control architecture for intelligent autonomous robot swarms, in the perspective of network robotics state of the art and the coming pervasive computing and communication scenario.

It underlines the issues and opportunities created by the general objective of harnessing the huge on demand computing power provided by grid technologies to expand the capabilities of networked robotic systems.

As an example it is shown how they are embedded into RobotGrid middleware system.

It implements a planning, control and cooperation architecture which is grid based and exploits the n-tiered java and wireless technologies of common use in the world wide web, see Fig. 1.

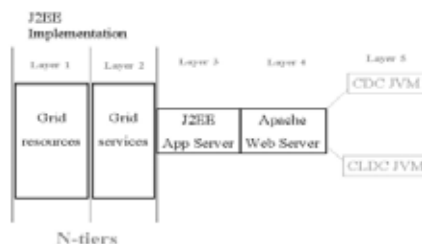


Fig. 1. The n-tiers.

Moreover from a computer science standpoint it is interesting to blend grid and multiagent system technology paradigms.

RobotGrid system architecture is a natural application of common web application design concepts to the task we have identified. The most common application environments which enable this kind of architectures are Java 2 Enterprise Edition (J2EE) and .Net framework.

A few hints about the 'enabling' technologies are recalled.

Those technologies are the 'grid' infrastructure itself, a - comparatively - widespread development toolkit for interfacing to grids (the 'Globus' toolkit), the grid service implementation.

According to various sources (e.g. Gartner Research, the technical report on Pervasive Computing and Communications Thematic Group Report of the IST Beyond the Horizon project), we are moving, from 2006 and beyond, towards an era of pervasive computing and communication devices.

In the meantime the “moore’s law” is continuing to work (although in ten/twenty years we could approach the physical limits of VLSI technology), see Fig 2.

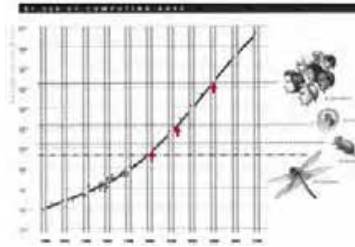


Fig. 2. 1000 dollars buy (from ‘The Age of Spiritual Machines’, R.Kurzweil).

According to Gartner the next years will see the spreading of:

- pervasive wireless
- real-time infrastructure (grids)
- service-oriented architecture
- low power-consumption mobile devices

There are some visionary statements, like L.V.Gertsner (former IBM’s CEO) talking about “...a billion people interacting with a million e-businesses with a trillion intelligent devices interconnected ...” .

There are some important international initiative like the South Korean Ministry of Information and Communication UR (Ubiquitous Robotics Companion) project.

Grid computing characteristics enable the deploying of huge networks of comparatively intelligent robotic agents. We describe here a middleware architecture which is grid-based and exploits the n-tiered java and wireless technologies of common use in the world wide web in the first stage of development and a new concept highly distributed swarming autonomous multi robot system as a further step.

As a typical real time multiagent application it constitutes an interesting challenge for the grid paradigm: as a consequence the possible fields of application of grid technologies are expanded making possible new applications in new domains like, for example, interactive multimedia user interface applications.

A grid infrastructure can be regarded as a distributed computing infrastructure which enables the supply of computational capacity on demand.

This kind of infrastructure can deploy very high calculation performance to a great number of points of service with a great variation as a function of time, position, serviced unit.

Actually, this fits the elaboration needs of each individual robotic agent in the considered environments, which can lead to very different computational loads from robot to robot in a given time frame. This allows to optimize the exploitation of the available overall capacity. Self adaptation capabilities are enhanced: for instance, the destruction of a robotic agent can be overcome moving another one in the same place and substituting the ‘mind’ of the out of order robot to the one substituting it.

1.1 Network robotics

Networked robotics is a new and growing area of research integrating robotics, networking, multimedia and component-based software technologies in support of local, remote, distributed, cooperative and multi-robot system architectures and operations.

Networked robotics constitutes a new conceptual framework in which to explore and extend traditional problems in robotics, while creating important new robotics applications.

A paradigm shift (in the T. Kuhn sense) is emerging: from designing an architecture that integrates sensors and actuators within a single physical platform under centralized control to one in which robotic sensors, actuators, computing, and human interfaces are distributed across multiple physical robot platforms, possibly in time-delayed and/or asynchronous communications.

There are several important research project in this field, some of the most important are the already quoted Korean KIST URC project, US DARPA MARS Vision 2020, Japan Network Robot Forum.

Close in concept are the US ARMY FCS initiative and the DARPA Coabs project.

The URC and the Network Robot Forum motivation are clearly showing, on top the scientific challenges the economical relevance of the network robotics field. This project in particular looks like an extension to robotics of the thin-client concept of internet computing: 'the network is the robot'.

Network Robot Forum estimates the networked robot industry to grow to 19.8 trillion yen (>130 G€) by 2013.

KIST hope to put a 'network robot' in any Korean household (100% penetration!) by year 2010 (actually S. Korea has a strong high bandwidth internet penetration.



Fig. 3. Examples of Network Robot usage (courtesy KAIST).

There is a clear need for research and ongoing research activities in many fields, spanning from research on sensor, actuator, and processor resources to the development of models for collective sensing and decentralized control in distributed robotic systems and of modular, extensible architectures for multi-robot interaction and real-time coordination

Moreover we can see extensions of the Networked Robotics paradigm and architectures to related problems: smart structures, smart home, pervasive computing and ambient intelligence and to the problem of cognition itself: (multiagent) cognition

could/should be seen as a cooperative effort raising question spanning from computer science to epistemology, could we talk of a 'network embodied cognition'?

1.2 Multi Agent Systems and Multi Robot Systems

The emergence of autonomous agents and multi-agent technology is one of the most exciting and important events to occur in computer science during the 1990s. It is widely believed that this technology will play a central role in the development of complex distributed systems, networked information systems, and computer interfaces during the next decades.

Multi Robot Systems (MRS) are a particular form of multi agent system.

There is a natural exchange of concepts and methods between the two areas.

The advantage of a MRS are in term of the effectiveness and performance and in term of robustness and reliability (Dudek, 1996; Parker, 1998). MRS research and development is affected by recent research on biological systems and by new models in cognitive science and economics (Cao, 1997), sometimes influenced by complex system theory. According to the number and kind of mutual interaction between the robotic agent of the multi robot system can be regarded as a swarm (Cao, 1997; Parker, 1998), or a colony (Arkin, 1997) or, more generally, as a robot collective (Dudek et al., 1996).

A special kind of MRS are the RoboCup teams (Asada, 1998; Kitano et al., 1999). An interesting feature of RoboCup and similar environments, is that the RoboCup environment is not only highly dynamic but also involve an opponent robotic network.

1.3 Grid and Multi Agent systems

There is a common underlying thread between multiagent and grid technologies, the creation of networks of entities (call them communities or VO) with a common goal.

In the case of grids the focus is on how the communities form and operate, on the other hand, the understanding of how this can be used in order to obtain large scale systems with stable collective behaviour is less mature.

Commonly available Grid tools provide mechanisms for accessing data on different storage system but not for the semantic integration of that data.

Grids need flexible decentralised decision capabilities agents need a robust distributed computing platform allowing them to discover acquire federate and manage the capabilities necessary to execute their decisions.

One approach is layering the two technologies i.e. put a multiagent system on top of a grid infrastructure.

It seems likely that a grid/agent approach will give better results with a fine grain intertwining of the two technologies.

1.4 Pervasive computation scenario

The large scale computing, grid, paradigm shift is enabled by recent advances in computer science, with parallel and distributed computing playing a key role.

The grid paradigm offers an important abstraction for combining national and continent wide computational resources..

Driven by the increase in internet usage grid computing will provide the next phase in the use and dissemination of knowledge.

The grid can be seen as a way for integrating resources which can span from dedicated high performance arrays of computers to low end clusters of workstation.

To manage this complexity a kind of AI behaviour is needed.

2. RobotGrid Architecture

2.1 Grid Infrastructure

Grid technologies support the sharing and coordinated use of heterogeneous resources (memory, cpus, devices of any kind) in dynamic computing network environments. This networks of resources can be distributed on a local area network, or geographically on an intranet but also over several (virtual) organisations or the WWW.

The key distinctive point of this kind of infrastructures is that resources are not statically allocated to a specific service /task (or pool of services)but they are assigned dynamically on the basis of current needs.

Thus allowing an optimal exploitation of available resources.

An other remarkable advantage, which could be of great help in deploying large scale or intelligence intensive application of the kind we are considering here, connecting to a public large scale grid could provide unprecedented high performance computing capabilities enabling a whole array of advanced highly effective applications.

Grid technologies were conceived in order to support scientific cooperation on various issues from (Catlett, 1992; Catlett et al., 1992; Foster, 2002; Foster & Kesselman, 1999a; Johnston et al., 1999; Stevens et al., 1997), graphical rendering of large databases, high computation needs for data analysis, integration of scientific instruments with remote data and computing resources (Johnston, 1999).

It is reasonable to envision applications of this kind of technologies to the robotic field where the management of dynamic sets of resources and services can be of great help.

In practice it is necessary a set of 'API' (Application Programming Interface' in order to be able to really deploy a given software system on a given platform

The Globus toolkit is one example of this set of API.

2.2 The Globus Toolkit

The Globus Toolkit (Foster & Kesselman, 1999b; Foster et al., 2001) is a community-based, open-architecture, open-source set of services and software libraries that support Grids and Grid applications. The Globus Toolkit has been adopted by hundreds of sites and by many of the chief Grid projects around the world.

Globus includes a set of basic utilities:

- Grid Resource Allocation and Management (GRAM) protocol which supplies service creation and management
- Meta Directory Service (MDS-2) , which supplies information discovery
- Grid Security Infrastructure (GSI), which supports single sign on, delegation and other security features

From our standpoint this kind of toolkits are the ground base for implementing web services which make available to the planning and control system grid functions in order to exploit the expected computation power advantages.

2.3 'Grid services'

A 'grid service' is a web service that provides a set of grid functions.

A key point are transient service instances. Serviced units typically do not require a set of

persistent services for handling their needs, but, instead, they need 'transient' service instances to manage a particular time limited state of an ongoing activity, when the activity finishes the service is no longer needed and the instance can be destroyed.

In our case service instances could be instantiated dynamically to supply dynamically supplemental computing, memory and bandwidth resources in order to cope with a transient workload.

2.4. RobotGrid tiered structure

As told above, the middleware architecture described here is a natural application of common web application design concepts to the task of enabling innovative multi robot system applications.

RobotGrid middleware architecture is based on J2EE.

Within the research community is emerging a set of 'new generation' web technologies which are collectively known as 'grid' technologies. A bunch of products exploiting various aspects of this methods are already available on the market (e.g. Oracle 10g Application Server).

We will see below how this methods can be blended in order to achieve an environment suitable for advanced robotic applications.

And how the overall capabilities of a 'robot grid' can be augmented interfacing a one or more wireless sensor networks. In order to better explain the suggested solution, a few hints about the 'enabling' technologies are recalled below.

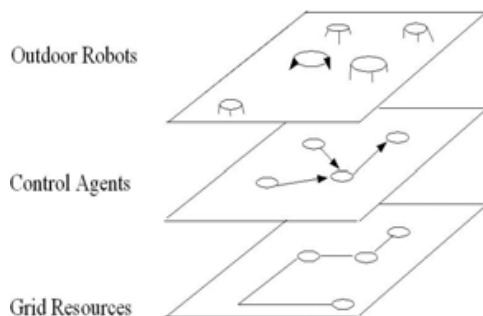


Fig. 4. RobotGrid layered architecture.

Those technologies are the 'grid' infrastructure itself, a - comparatively - widespread development toolkit for interfacing to grids (the 'Globus' toolkit), the grid service implementation, the WSN technologies.

The n-tiered structure of the RobotGrid middleware is depicted in Fig. 5. At the lower layer we have the infrastructure resources: computers with their operating systems, san (storage area network), nas (network area system), (remote)sensors,(remote) robots. Immediately above we have the grid services layer, which implements the virtualization of the resource of the bottom level, according to the grid service paradigm on demand. Then we have the RTRDBMS (Real Time Relational Data Base System), RTKBMS (Real Time Knowledge Base Management System) and other data and rule application management system, simulation tools. Above we have the application server level, where the (e.g. enterprise java) beans implementing the

planning and control logic reside. The DB and Application are here together regarded as the 'Layer 3'.

On this layer there are beans calling from their code for instance the Globus APIs, other interfacing for instance the Java Virtual Machine embedded in the autonomous robot, other implementing the Planning and Control Logic.

The web serving layer, which make available through the http protocol the complex services made possible by the overall architecture.

The onboard virtual machine, which is the part of the system that necessarily resides on the robot and manages by means of 'applets' the physical interfaces of the robot and the ones with the control algorithms, rules and methods. A description of an application deployment environment fit for this use (Java 2 Micro Edition) in described in the next paragraph.

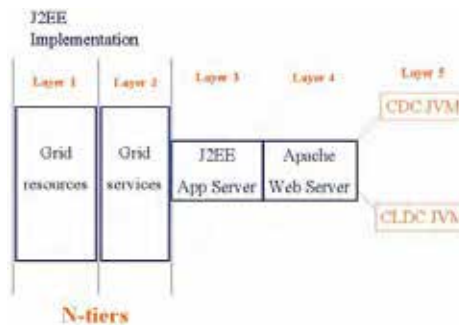


Fig. 5. The n-tiers.

We show in figure 6 the flow of the sense/reaction scheme of a single robot.

The 'applet' running on the robot onboard virtual machine sense e.g. a movement of a 'focus object' in the 'environment'.

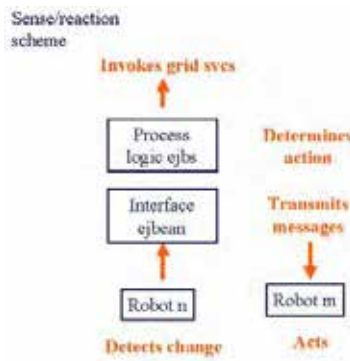


Fig. 6. The sense/reaction scheme.

A front end unit of service (an interface 'enterprise java bean') is invoked with a string of parameters , the bean activate the necessary actions instantiating the necessary transient grid services (adding resources where necessary) and some where another bean dispatch a message that trough an on board applet act in the environment (e.g. moving one or more robot on the field from one position to another) .

From a simplified standpoint, we can say each robot in the 'swarm' is represented by a software agent within RobotGrid.

Immediately above we have the grid services layer, which implements the virtualization of the resource of the bottom level, according to the grid service paradigm on demand.

This layer is running on a soap server and wraps the underlying grid services implemented encapsulating of one of the grid service sets under development by various organization or, directly, the java cog apis.

The exposed web services are invoked by the set of ejbeans running within a J2EE application server container mapping the swarm as a collection of RobotJbeans and Stateful Session Beans representing the individual robot of the physical swarm.

A very simplified schema is depicted in fig. 7.

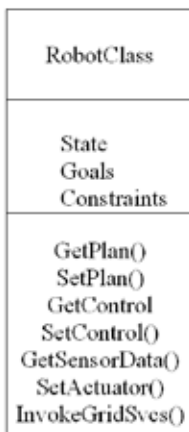


Fig. 7. The Robot Class

Within the same (logical) application server run the wsn set of wsn ejbeans mapping the sensor network wrapping the call to the wsn subnetworks gateways, for instance Intel/Crossbow Stargate devices.

A schematic is shown in fig. 8.

3. Issues and opportunities

The architecture described here is largely based on 'commodity' software environments .

There are, although, limits that prevent from some extreme application.

For example the smaller robotic unit should - without workarounds and with the constraint of using commodity system environment like those quoted above - be in the 10 cm scale.



Fig. 8. The Wsn Class.

The environment is complex and although capable of supplying high computing power it requires some resources.

The biggest issues regard Quality of Service, this kind of application requires:

- Execution time predictability
- Predictable scheduling algorithms;
- Predictable cost models for parallel algorithms;
- Failure handling and graceful degradation on the Grid side

This can be achieved by means of scheduling algorithms, exploiting:

- Staging of executable files and data;
- Application start-up and overhead reduction and predictability;
- Advanced allocation policies;
- Resource Agreement reliability;

On the other end, the real strength of the proposed architecture is that an inherently on demand flexible and distributed grid based computing environment is coupled with a distributed sensing/actuating environment.

Another advantage is the capability to allocate the computing resources to the robot units which really need them when and how they need them,

Given that and considering that the physical implementation and 'low' level control algorithms perform in parallel as well as the robot tasks and goals and the most common computing tasks (planning for all) are of a kind that can greatly benefit from parallel execution with are best condition to fully exploit advantages of the underlying architecture.

Last but not least the computing grid can be a disconnected local one, but also a big Virtual Organisation grid or even 'the Grid', as a consequence the deployed computing power can be quite noticeable.

Today, applications of networked automation are already pervasive in most fields of human endeavor. The networked automation tools and technologies have a wide range of applications. New questions, that could not have been formulated a few years ago, are being posed, mainly due to the technological advancements in robotics, control, computing and communication. Again, these technological developments made

possible to envision the implementation of systems which could have not been imagined before.

Network and networked automation are already ubiquitous and pervasive in most areas of daily experience of citizens in the old and new industrialised countries.

The new advances in robotics (in particular service robotics) , control (in particular distributed control), computing and communications (in particular the web, the semantic web, the grid, object of this proposal), create the opportunities for new product, services, new 'service-products'.

As a consequence of this technological driving force, the transversal field on network robotics is gaining momentum enabling a broad range of new solutions for old problems and new opportunities.

- multi-vehicle systems, robotized highways, waterways
- domotics
- Industrial automation: wireless automation and wireless factory; new generation robots, multi agent systems, holonics systems
- Ambient intelligence and intelligent spaces
- Health care: sensor networks for monitoring and intervention; integrated robotic help;
- remote interventions; automated drug delivery, etc.
- Earth sciences: networks of sensors and robotic devices for detailed scientific field
- studies in remote and hostile environments inaccessible to humans; studies of global warming, etc.
- Disaster management: ; USAR, intervention in disaster
- scenarios; fire detection, management and fire fighting, etc.
- Security, surveillance and civil defense: deployment of interoperated sensor and communication
- networks in disaster scenarios; surveillance and security
- Agriculture: crop monitoring and interventions; optimization of irrigation systems; harvesting
- Forest management
- Constructions
- Urban maintenance
- humanitarian demining
- mining operations
- Management of natural resources: networks of sensors and robotic vehicles for water management; air quality; etc.
- Military: new strategic concepts and tactical deployments rely heavily on network centric concepts, which involve not only manned systems but with an increasing rate on sensor and robotic vehicle networks (US Army Future Combat System)
- An important aspect of networked robotics system is their 'multi role' characteristics a middleware for network robotics can in principle be used for USAR or for urban cleaning or for forest management.

Making the grid technologies available for network robotics means to set up an enabling technological platform which could prepare the advent of a (huge) number of

new service products ready for the upcoming pervasive computing and communication scenario.

4. Reliability and Performance

The RobotGrid grid environment is a quite complex one, as a consequence giving quantitative prediction of a network robot system based on this middleware can be difficult, nevertheless it is possible to draw some conclusions on the basis of simple assumptions, on the basis of system reliability theory and of multi server system performance evaluation techniques.

Since it is not possible to define an explicit closed form model of the system, better predictions can be obtained by simulation and this is part of ongoing research.

4.1. Modeling Reliability

The reliability of a system is the probability that it is working properly in a given time period.

The reliability of a system made of n parallel components is given by:

$$R_p = 1 - P_r \quad (1)$$

where, r_i reliability of the i component:

$$P_r = \prod_{i=0}^{n-1} (1 - r_i) \quad (2)$$

As a consequence:

$$R_p = 1 - \prod_{i=0}^{n-1} (1 - r_i) \quad (3)$$

if $r_i = r = \text{const}$, we have:

$$R_p = 1 - (1 - r)^n \quad (4)$$

In Fig. 9 we see how with several values of r we get a very high level of reliability with a small number of parallel components: in our case both the robots in the physical world and the computing resources supplied by the grid are sets of parallel resources.

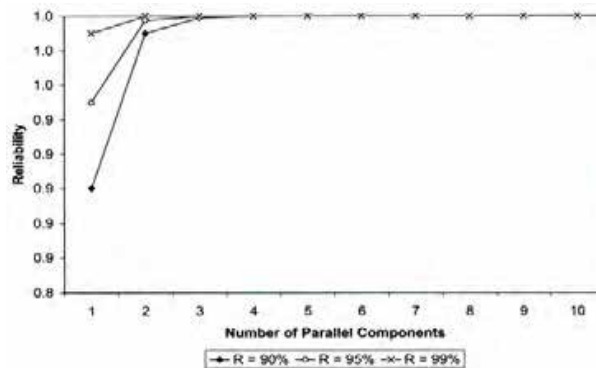


Fig. 9. Reliability of parallel components.

4.3 Modeling Performance

Let λ (request/s) the average arrival rate of rate of requests, μ (request/s) the average request throughput of the servers.

Assuming infinite population and finite queue, we get,
 P_k the fraction of time the server has k request

$$P_k = \begin{cases} \frac{1 - \lambda/\mu}{1 - (\lambda/\mu)^{W+1}} \left(\frac{\lambda}{\mu}\right)^k & k = 0, \dots, W \quad \lambda \neq \mu \\ 1/(W+1) & k = 0, \dots, W \quad \lambda = \mu \end{cases} \quad (5)$$

U , server utilization:

$$U = \begin{cases} \frac{\left(\frac{\lambda}{\mu}\right) \left[1 - \left(\frac{\lambda}{\mu}\right)^W\right]}{1 - \left(\frac{\lambda}{\mu}\right)^{W+1}} & \lambda \neq \mu \\ W/(W+1) & \lambda = \mu \end{cases} \quad (6)$$

X , average server throughput:

$$X = U \times \mu \quad (7)$$

\bar{N} , average number of requests in the server:

$$\bar{N} = \begin{cases} \frac{\left(\frac{\lambda}{\mu}\right) \left[W \left(\frac{\lambda}{\mu}\right)^{W+1} - (W+1) \left(\frac{\lambda}{\mu}\right)^W + 1 \right]}{\left[1 - \left(\frac{\lambda}{\mu}\right)^{W+1}\right] \left(1 - \frac{\lambda}{\mu}\right)} & \lambda \neq \mu \\ W/2 & \lambda = \mu \end{cases} \quad (8)$$

The previous equations lead to:

$$R = \bar{N} / X \quad (9)$$

In Fig. 10 is plotted the average response time (R) of an infinite population/finite queue multiserver system, according to equation (9).

It shows how there is a threshold number of servers (in the ideal model) above which the response time can be made very low.

This suggests the usefulness of adding computing capacity on demand.

The plot assumes 500 requests/s from the 'swarm', a limit of 200 simultaneous connection and of 20 request/s for each identical server.

5. System Simulation

As told above, since it is not possible to define an explicit closed form model of the system, when designing a solution for a specific application the behaviour of the system should be supported by a simulation tool.

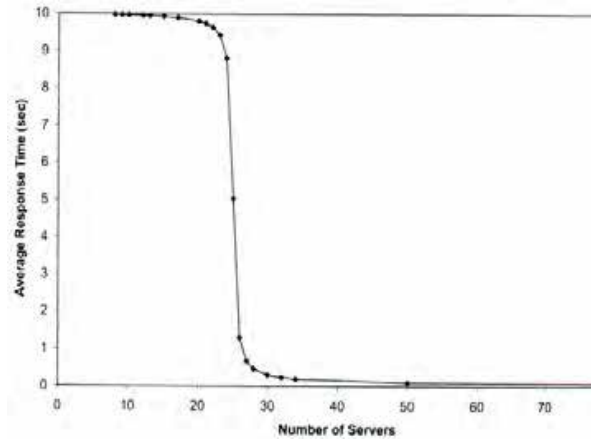


Fig. 10. Response time vs. number of servers.

The simulation of the system is performed in Webots (Webots, 2006).

The multirobot system is constituted by a variable number of Khepera robots (see Fig. 11).

Khepera is a mini mobile robot produced by K-team SA, Switzerland.

It is a caster-like robot with two differential wheels, which can be equipped with various kind of sensors and actuators.

In our first simulations the Khepera are identical and are equipped with 7 distance sensor and 7 light sensors.

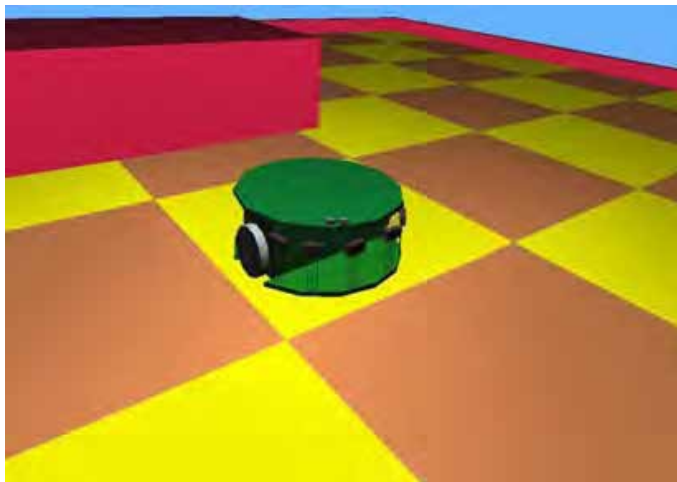


Fig. 11. The Khepera robot.

The 'environment' where they are tested is very simple, see Fig. 12.



Fig. 12. The simulated environment.

In the future we will test more diversified 'swarms' in environments made progressively more complex.

The Khepera robots can be controlled remotely through a tcp/ip interface. They are suitable to be interfaced to the architecture we have described.

The grid infrastructure is simulated as a variable set of computing and communication resources.

6. Further Research

We have described at a conceptual level the architecture of a middleware system for intelligent robot swarms integrated with one or more wireless sensor networks.

It is ongoing the simulation and implementation of the software architecture in specific cases and the physical implementation in some relevant case among those quoted above.

Possible application examples could be humanitarian demining , robotic cleaning 'agency' and intelligent vehicles for urban and suburban mobility.

7. Conclusions

From a certain respect what is described above can be seen as a generalized grid where some of the resource managed are various kind of intelligent autonomous robotic agents.

This seems to show some inherent advantages.

If the computing grid, is a VO grid, or 'the grid' the high performance computing capabilities added to the multiplication effect in the 'real world' of deploying a swarm of intelligent autonomous robots could apparently open a wide range of new advanced applications.

Moreover, from a computer science standpoint it is interesting to blend a multiagent system with a grid infrastructure, while the Quality of Service requirements of a real time system expand the reach of application of grid technologies.

As we move towards a pervasive computation and communication scenario, where grid computing facilities will (or could) become a commodity service, the kind of architecture we described here, could become quite common.

8. References

- Arkin, R.C. & Bekey, G.A. *Robot Colonies* (1997) Kluwer Academic Publishers, 1997.
- Asada, M. (1998) The RoboCup physical agent challenge: Goals and protocols for Phase-I. In Kitano, H. editor, *RoboCup-97: Robot Soccer World Cup I*, 1998.
- Cao, Y. ; Fukunaga, A. & Kahng, A. (1997) Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:1-23, 1997.
- Catlett, C. (1992) In Search of Gigabit Applications, *IEEE Communications Magazine* (April).42-51. 1992.
- Catlett, C. & Smarr, L. (1992) Metacomputing, *Communications of the ACM*, 35 (6). 44--52. 1992.
- Dudek, D. ; Jenkin, M. ; Milios, E. & Wilkes, D. (1996) A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375-397, 1996.
- Foster, I. (2002) The Grid: A New Infrastructure for 21st Century Science, *Physics Today*, 55 (2). 42-47. 2002.
- Foster, I. & Kesselman, C. (1999) *Globus: A Toolkit-Based Grid Architecture*, In Foster, I. And Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 259-278.
- Foster, I. & Kesselman C. (eds.). (1999) *The Grid: Blueprint for a New Computing Infrastructure* Morgan Kaufmann, 1999.
- Foster, I.; Kesselman, C. & Tuecke, S. (2001) *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, *International Journal of High Performance Computing Applications*, 15 (3). 200-222, 2001.
- Johnston, W. (1999) "Realtime Widely Distributed Instrumentation Systems" In Foster, I. and
- Kesselman, C. (1999) In Foster, I. And Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 75-103.
- Johnston, W.E.; Gannon, D. & Nitzberg, B. (1999) "Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid." In *Proc.8th IEEE Symposium on High Performance Distributed Computing*, (1999), IEEE Press
- Kitano, H. , Pagello, E. & Veloso, M. (1999) editors *RoboCup-99: Robot Soccer World Cup III*, Springer-Verlag, 1999.
- Parker, L.E. (1998) ALLIANCE: An architecture for fault tolerant multirobot cooperation, *IEEE Transactions on Robotics and Automation*, 14(2):220-240, 1998.

- Stevens, R.; Woodward, P.; DeFanti, T. & Catlett, C. (1997) From the I-WAY to the National Technology Grid, *Communications of the ACM*, 40 (11). 50-61. 1997.
- Webots (2006) <http://www.cyberbotics.com>. Commercial Mobile Robot Simulation Software

Acromovi Architecture: A Framework for the Development of Multirobot Applications

Patricio Nebot & Enric Cervera
*Robotic Intelligence Lab, Jaume-I University
Castellón de la Plana, Spain*

1. Introduction

The presented agent-based framework (Acromovi - an acronym in Spanish which stands for Cooperative Architecture for Intelligent Mobile Robots) was born from the idea that teamworking is an essential capability for a group of multiple mobile robots (Jung & Zelinsky, 1999; Mataric, 1998).

In the last years, there is an increasing interest in the development of systems of multiple autonomous robots, so that they exhibit collective behaviour. This interest is due to the fact that having one single robot with multiple capabilities may waste resources. Different robots, each one with its own configuration, are more flexible, robust and cost-effective. Moreover, the tasks to achieve may be too complex for one single robot, whereas they can be effectively done by multiple robots (Arai et al., 2002).

In this work is described the design and implementation of a distributed architecture for the programming and control of a team of coordinated heterogeneous mobile robots, which are able to collaborate among them and with people in the accomplishment of tasks of services in daily environments, the Acromovi architecture.

Acromovi architecture is a framework for application development by means of embedding agents and interfacing agent code with native low-level code. It addresses the implementation of resource sharing among all the group of robots. Cooperation among the robots is also made easier in order to achieve complex tasks in a coordinated way.

Moreover, in this work is emphasized the reusability of software, allowing the programmer to seamlessly integrate native software components (vision libraries, navigation and localization modules), by means of agent wrappers, providing the programmer with a set of tools for mobile robots. Also, it allows sharing the robots' resources among the team and an easy access to the robots' elements by the applications. Other important characteristics of the Acromovi architecture are the scalability and ease-of-use.

Though robot programming has been extensively done in C or C++ languages, a Java-based multiagent development system was chosen to develop the architecture of our team of robots (Nebot & Cervera, 2005a). Among Java strengths, those particularly pursued were the high-level communication capabilities, and the native interface to existing C/C++ code.

On the other hand, Acromovi architecture is a distributed architecture that works as a middleware of another global architecture for programming robots. Though any common agent architecture can be used, it has been implemented by means of the JADE (Java Agent

Development Framework), a tool for the development of multiagent systems, implemented in JAVA, that fulfils with the FIPA specifications.

The embedded agents that constitute the Acromovi architecture work as interfaces between the applications and the physical elements of the robots. Some agents also make easier the handle of these elements and provide higher-level services. The embedding agents are the key for powerful distributed applications being rapidly developed.

Multiagent systems are an important tool for the development of new robotic applications. They are the natural environment for the development of applications which consist of more than one robot. And they make possible the fast implementation of powerful architectures for the specification and execution of tasks for those teams of robots.

In Section 2, some related works in the literature are compared. Next, Section 3 describes the different robots that make up the team. In Section 4, a description of the Acromovi architecture is given, from the design to the implementation. Some examples of applications implemented with the Acromovi architecture are presented in Sections 5. These applications demonstrate the rapid development capabilities and ease-of-use of the agents implemented with this framework. Finally, Section 6 concludes and presents some final remarks.

2. State of the art

The amount of research in the field of cooperative mobile robotics has grown steadily in recent years (Parker, 2000; Cao et al., 1997). Some examples of these works are presented below, emphasizing their potentials, and drawbacks, and whether they are related (or not) with embedded agents in mobile robotics.

CEBOT (Cellular roBOTics System) is a hierarchical decentralized architecture inspired by the cellular organization of biological entities. It is capable of dynamically reconfiguring itself to adapt to environment variations (Fukuda & Iritani, 1995). It is composed by "cells", in the hierarchy there are "master cells" that coordinate subtasks and communicate among them.

On the other hand, *SWARM* is a distributed system made up of a great number of autonomous robots. It has been called the term "SWARM intelligence" to define the property of systems of multiple non-intelligent robots to exhibit a collective intelligent behaviour. Generally, it is a homogeneous architecture, where the interaction is limited to nearest neighbours (Johnson & Sugisaka, 2000).

Other interesting project, *MICRobES*, is an experiment of collective robotics that tries to study the long time adaptation of a micro-society of autonomous robots in an environment with humans. Robots must survive in these environments as well as cohabit with his people (Picault & Drogoul, 2000).

In an attempt to use traditional IA techniques, the *GOPHER* architecture was thought to resolve problems in a distributed manner by multirobots in internal environments (Caloud et al., 1990). A central tasks planning system (CTPS) communicates with all the robots and disposes a global vision of the tasks that has been done and of the availability of the robots to perform the rest of tasks.

An example of behaviour-based architecture is *ALLIANCE*, which is oriented to the cooperation of a small-medium team of heterogeneous robots, with little communication among them (Parker, 1998). It assumes that robots are relatively able to discern the effects of their actions and those of the rest of agents, either through its perception or through communication. Individual robots act based on behaviours or sets of behaviours to make their tasks.

ABBA (Architecture for Behaviour Based Agents) (Jung & Zelinsky, 1999) is another behaviour-based architecture for mobile robots, whose purpose is to design an architecture

to model the robot's behaviour so that it can select reactive behaviours of low level for his survival, to plan high level objectives oriented to sequences of behaviours, to carry out spatial and topological navigation, and to plan cooperative behaviours with other agents.

Dynamic physical agents are considered in an architecture called *DPA*, being well suited for robotics (Oller et al., 1999). A physical agent is the result of integrate a software agent in an autonomous hardware. This hardware is frequently a mobile robot. *DPA* architecture shows the agent like a modular entity with three levels of abstraction: control module, supervisor module and agent module. The architecture's operation is based on two basic processes for each module: the process of introspection and the one of control, and in a protocol of intermodular dialogue which allows the exchange of information between modules.

In order to reuse native software components, agents can be embedded in an interface level or middleware. *ThinkingCap-II* (Cáceres et al., 2003) is an architecture developed, in a project of distributed platform based on agents for mobile robots. It includes intelligent hybrid agents, a planning system based on a visual tracking, vision components integration, and various navigation techniques. Furthermore, it has been developed over a real-time virtual machine (RT-Java), implementing a set of reactive behaviours.

Miro is a middleware for create autonomous mobile robot applications from the University of Ulm. It is based on the construction and use of object-oriented robot middleware to make the development of mobile robot applications easier and faster, and to foster portability and maintainability of robot software (Enderle et al., 2001). This middleware also provides generic abstract services like localization or behaviour engines, which can be applied on different robot platforms with virtually no modifications.

The previous works implement architectures and systems so that teams of mobile robots can make cooperative tasks. Our work consists of the implementation of an architecture of such type, emphasizing software reuse, and adding the versatility and power of the multiagent systems, for the resolution of cooperative tasks for a group of heterogeneous robots.

3. Description of the team of robots

In order to understand the purpose and working environment of the presented agent architecture, their physically embedding instances are described in this section: a team of mobile robots. For the development of this project, a team consisting of six Pioneer robots and one Powerbot robot is used, all of them manufactured by ActivMedia Robotics, as can be seen in Fig. 1.



Fig. 1. Team of robots used in the project.

Though sharing the same platform, the robots exhibit different features, constituting therefore a heterogeneous group. Within the group, dimensions of robots may vary due to the different accessories added to robots (cameras, laser, grippers), or its own size (Pioneer vs. Powerbot).

Particularly, only one Pioneer has a laser rangefinder system and two Pioneers carry camera systems; moreover, the Powerbot has front and rear bumpers and front IR sensors. All the Pioneer robots have a gripper and a front ring with 8 sonar discs, except the robot with laser that also includes a rear sonar ring. The Powerbot has 28 sonar discs distributed 14 at front and 14 at rear, but it does not include a gripper. Instead, a 7 d.o.f. arm is mounted on its top, as can be seen in the last section of this work.

All robots of the team maintain the same logical design based on a microcontroller which transfers the readings of the standard elements of the robot (ultrasound sensors, wheel encoders) via RS232, by cable or radio, to a computer client where the applications are executed. This computer can be located on board of the robot or not, in which case the communication is made by radio-modem. In our team, onboard computers are only available on three Pioneer robots (those that also carry either a camera or the rangefinder) and the Powerbot.

Finally, the robots with onboard computing power are connected by means of a Wireless Ethernet network, which, through several base stations, can access to the local network of the university, and to the Internet.

4. Acromovi architecture

To establish mechanisms of cooperation between robots implies to consider a problem of design of cooperative behaviour: given a group of robots, an environment and a task, how the cooperation must be carried out. Such problem implies several challenges, emphasizing among them the definition of the architecture of the group.

The multiagent systems are the natural environment for such groups of robots, making possible the fast implementation of powerful architectures for the specification and execution of tasks.

4.1 Design of the architecture

The robots that constitute the team have already a programming architecture with native (C/C++) libraries for all their accessories. In the Pioneer robots, this architecture is splitted in two layers. The lower layer is called ARIA, and it takes care of the requests of the programs to the robot components. The upper layer -called Saphira- provides services for range sensor interpretation, map building, and navigation.

However, the existing native code is oriented to both local and remote processing from only one controller, without defining collaboration mechanisms between robots or applications. Acromovi architecture tries to overcome this problem by the introduction of a new level over this architecture that allows an easy collaboration and cooperation.

Also, the Acromovi architecture is able to subsume any other extra software, like the ACTS (a colour tracking library) and the Vislib (a framegrabbing library). ACTS is a native (C/C++) module which, in combination with a colour camera and a framegrabber, allows the users to select regions of colour in an image. ACTS is an

external program to the architecture that works as a server, and the applications work as clients that communicate by means of an ARIA library. Vislib is a library that provides a fast and flexible image processing and single-camera machine vision. The image processing functions that Vislib offers are generic convolution (2D kernel), filtering and morphological tools. Vislib also is written in the C language.

Subsuming existing C/C++ libraries is a powerful and quick method for rapidly developing Java code and thus embedding agents with all the functionality of native systems. Furthermore, the new code adds seamless integration with other distributed agents, either running on remote applications, or even as applets in web pages.

Acromovi, in Fig. 2, is a middleware between the native robot architecture and the applications, which allows sharing the resources of each robot among all the team of robots. This middleware is based on an agent-embedding approach.

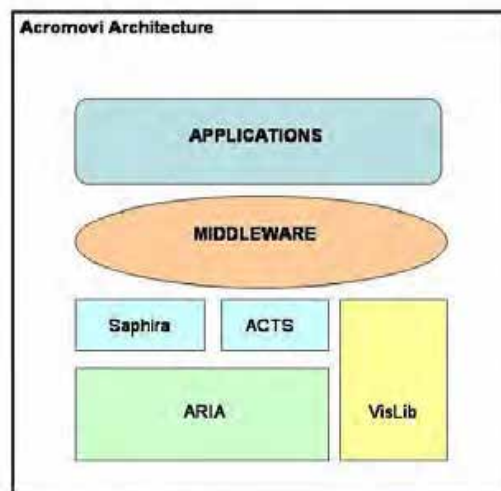


Fig. 2. General diagram of the Acromovi architecture.

This middleware level has also been divided into two layers. In the lower layer, there are a set of components, such as the sonar, the laser, the base, and so on. The others are special components that offer services to the upper layer, as the vision, the navigation or the localization.

These components only perform two kinds of functions: requests processing and results sending. Thus, they could be implemented like software components. But because of reasons of efficiency, implementation facility and integration with the whole structure, they have been developed as agents that encapsulate the required functionality. But one should take into account that this implementation may vary in the future if needed.

The upper layer of the middleware comprises a great variety of embedded and supervising agents, e.g. agents that monitor the state of the agents/components of the lower layer, agents that provide services of subscription to a certain component, agents that arbitrate or block the access to a component, and any other type of agent which might

offer a service that can be of interest for the whole architecture or for a particular application. These agents act as links between the applications and the agents that access the components of the robot.

The structure of the entire middleware layer can be seen in the Fig. 3.

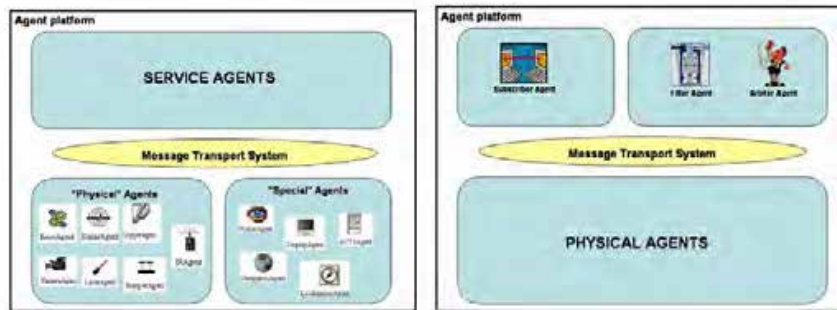


Fig. 3. The middleware layer: lower & upper layers.

The presented middleware has made feasible the change from an abstraction based on library functions for the handling of the robot -that are available in compilation and not consider multiplicity of systems- to an abstraction based on embedded agents -that are available in execution and natively distributed. One major design guideline was the distribution requirements and the communication between various platforms.

Because of the heterogeneous character of the team, there are different middleware layers for the robots, depending on the configuration of each robot of the team. These middleware layers are generated in execution time according to the elements that each robot has active at the moment of its execution. Therefore, each middleware layer will have active those agents that permit the access to the own components of the robot. In the Fig. 4, several examples of different configurations of the middleware layer are depicted. Only the agents of the lower layer are presented in each configuration, due to the agents of the upper layer depends on these agents.

The applications layer is on top of the middleware layer. These applications can be implemented as agents too. The applications, in order to access to the components of the robots, must communicate with the agents of the middleware, which then access to the bottom layer that controls the robot.

Once an application has been tested, and if it is useful for the whole architecture, it can be converted in a new agent of the upper layer of the middleware. Thus, each application done can increase our system to make applications more difficult and more interesting, following a bottom-up design.

4.2 Implementation of the architecture

Due to the fact that the middleware is just a set of embedded agents, a multiagent systems programming tool has been selected for its implementation.

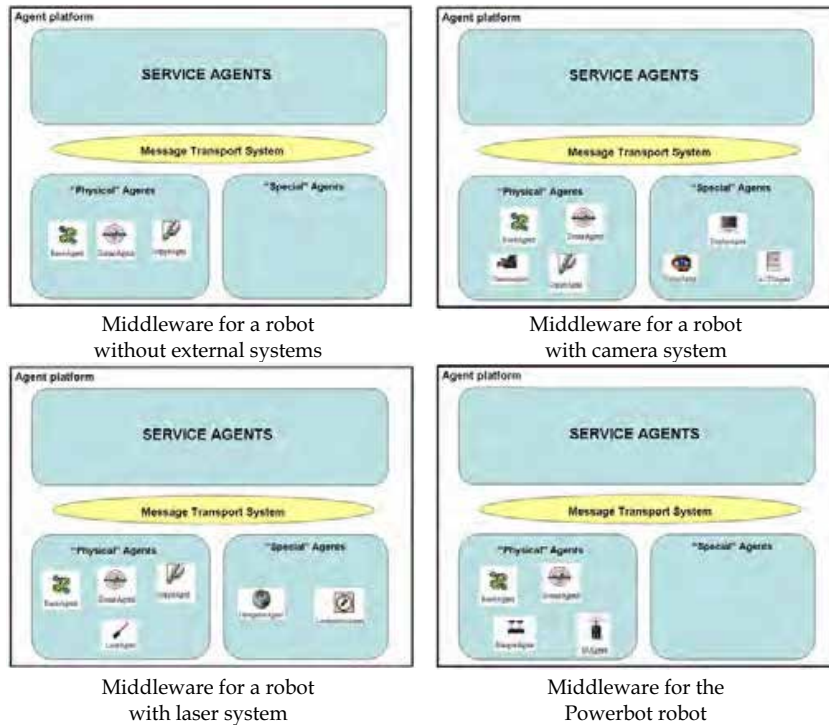


Fig. 4. The different middleware layers of the Acromovi architecture.

The multiagent tool chosen was JADE (Java Agent DEvelopment Framework). It is a software framework to develop agent-based applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. It is fully implemented in Java language and simplifies the implementation of multiagent systems through a middle-ware and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS). The JADE middleware implements an agent platform for execution and a development framework. It provides some agent facilities as lifecycle management, naming service and yellow-page service, message transport and parsing service, and a library of FIPA interaction protocols ready to be used.

The agent platform can be distributed on several hosts. Only one Java application, and therefore only one Java Virtual Machine (JVM), is executed on each host. Each JVM is basically a container of agents that provides a complete runtime environment for agent execution and allows several agents to concurrently execute on the same host.

The communication architecture offers flexible and efficient messaging, where JADE creates and manages a queue of incoming ACL messages, private to each agent. The transport

mechanism is like a chameleon because it adapts to each situation, by transparently choosing the best available protocol.

JADE supports also scheduling of cooperative behaviours, where JADE schedules these tasks in a light and effective way. The runtime includes also some ready to use behaviours for the most common tasks in agent programming, such as FIPA interaction protocols, waking under a certain condition, and structuring complex tasks as aggregations of simpler ones

Having all in mind, the Acromovi architecture is implemented. Each robot of the team is a main container, thus, there are seven main containers, one in each robot that works as host of the distributed network. In each container, there is a group of agents. These agents are created in execution time depending on the robot configuration. Each of these agents represents one of the elements that in that moment has active the robot. So, in that way, the heterogeneity mentioned in the generation of the middleware layer of Acromovi is implemented. All this, can be seen in the Fig.

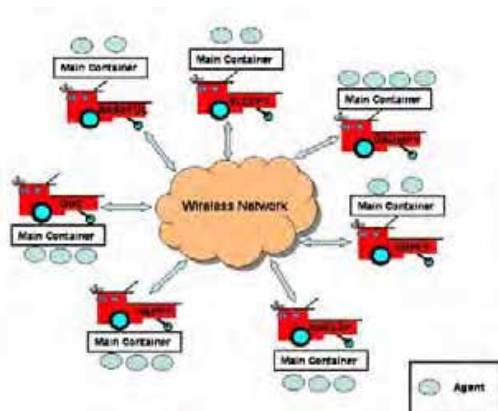


Fig. 5. Structure of the architecture implementation.

Because of the fact that ARIA and Saphira are implemented in C++ and JADE in Java, it has been necessary to use a mechanism to integrate such different programming languages. For this reason, JNI (Java Native Interface) has been used, thus allowing managing C++ code in Java programs.

The robot elements are represented and managed by agents. These agents communicate with the native layer of the global architecture by means of JNI.

5. Examples of applications developed with the Acromovi architecture

In this section, some applications implemented and tested with the Acromovi architecture are explained. These applications demonstrate that the Acromovi architecture is suitable for such tasks for it was developed. Also, they demonstrate the rapid development capabilities and ease-of-use of the agents implemented with this framework. Only the main idea and the operation of each application are shown here. For more details, consult the bibliography annexed.

5.1 Remote vision

This application consists of a graphical user interface (GUI), in Fig. 6, which allows the user to send commands to the vision and camera agents and displays the results. Such commands include the capture and display of the current image grabbed by the robot camera, the pan/tilt/zoom motions of the camera, and the image processing operations provided by the Vislib and Java 2D libraries. Such libraries provide low-level operations like thresholding, blurring, edge detection, as well as high-level ones as colour or motion tracking.



Fig. 6. Graphical user interface of the remote vision application.

The graphical interface is built around pure Java Swing components, thus resulting in a cross-platform application, capable of running in any operating system running the Java virtual machine.

The remote vision GUI just sends messages to the agents, which take care of the operations in the own robot. The agents return the appropriate result in another message to the GUI agent, which then displays the image (in grabbing and image processing operations) or just indicates whether the operation has been made correctly or not, in the case, e.g., of camera motions.

The main drawback of this application is responsiveness. Agent communications of images is quite surely not the best efficient way of sending data through a network. However, flexibility and ease of use are its main advantages.

5.2 Robot following

In this application one robot equipped with local vision pursues another one. The leader robot is assumed to navigate through an indoor environment, controlled either by a user or by its own navigation algorithms. The follower robot uses its PTZ vision system to determine the position and orientation of the leader robot, by means of a colour pattern target, in Fig. 7, consisting of three rectangles (Renaud et al., 2004).

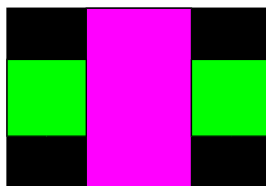


Fig. 7. Target used in the robot following application.

Such target, together with the local visual parameters, allows to easily computing the distance and relative orientation of the leader robot with regard to the follower-observer one.

An example of execution is depicted in Fig. 8. The application may be extended to more robots in a line formation, provided that each robot follows its predecessor, and it is equipped with a vision system.



Fig. 8. Some images from a robot following experiment.

Colour tracking is provided by the ACTS server, which needs to be trained with the colours of the target. The ACTS Agent works as link between the ACTS server and the Robot Following Application.

The application always tries to have centred the target in the centre of the image (and thus, always visible for the follower robot). To do so, it determines the blob centroid, computes the motion needed by the camera and compensates the camera and robot motions.

When the target is centred, the application calculates the leader robot orientation and position. With these information is easy to calculate the path to the target of the leader robot by means of a Bézier curve, with two intermediate Bézier's curve control points and the initial and final points (the positions of the follower and leader robots).

When the curve is defined, motion commands are sent to the robot to perform the real motion.

In Fig. 9, the trajectory of two robots in an indoor environment is shown. The follower robot (red trajectory, named Happy) follows a smoother trajectory than that of the leader robot (blue, named Sneazy), as a result of the algorithm based on Bézier curves.

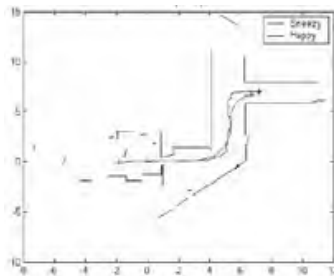


Fig. 9. Trajectory followed by two robots in a real robot following run.

It is important to remark that the whole process does add a very small processing overhead, thus demonstrating the capabilities of Java-based agent architectures for fast real-time vision-based tasks, as long as no image transmission is needed.

Also, it is important to point out that the system is robust enough to respond to sudden changes in the position or orientation of the leader robot, thanks to the real-time tracking system, and the small processing overhead.

5.3 Optical flow navigation

The purpose of this application is to use the optical flow technique to allow a mobile robot equipped with a colour camera navigates in a secure way through an indoor environment without collide with any obstacle. It reacts to the environment by itself. The objective is that the robot be able to avoid obstacles in real-time (Nebot & Cervera, 2005b).

The optical flow is the distribution of the apparent velocities of movement of the brightness pattern in an image, and arises from the relative movement of some objects and a viewer. As the robot is moving around, although the environment is static, there is a relative movement between the objects and the camera onboard the robot.

The visual information coming from the camera of the robot is processed through the optical flow technique, which provides the information the robot needs to safely navigate in an unknown working-environment.

From the optical flow generated in this way the robot can get an inference on how far an object present in a scene is. This information is embedded in the time to crash (or time to collision) calculated from the optical flow field. Using only optical measurements, and without knowing one's own velocity or distance from a surface, it is possible to determine when contact with a visible surface will be made.

The application, first of all, must to capture the input image frames. The video from the camera on board the robot is captured using the Java Media Framework (JMF). Next, the optical flow derived from the new image is calculated, and from it, the time-to-contact vector is calculated. In Fig. 10, there is an example of the optical-flow field and the time-to-contact graphic.

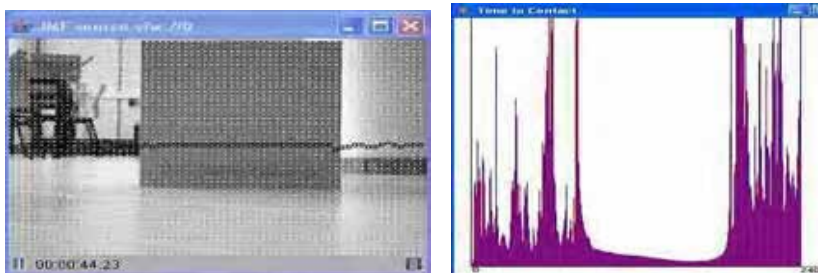


Fig. 10. Frame with the optical flow and the time-to-contact graphic associated.

Once the time-to-contact vector is calculated, an agent decides which situation is done in the environment, and depending on the situation decides the movements associated to perform by the robot.

In this case, the experimental results have shown that the robot is effectively able to avoid any obstacle, in real-time, based only on the information from the optical-flow and the time-to-contact agents.

6. Conclusion

In this work, the Acromovi architecture has been explained. This architecture is the support for all the programming and interaction of the agents that manages our team of robots.

The Acromovi architecture is basically an agent-based distributed architecture for the programming and controlling of teams of mobile robots. This architecture allows the use of native components, by means of agent wrappers. The agents add the distributed capabilities, with message interchange for cooperation between robots. This makes possible an important reutilization of code, this being a basic principle in software engineering.

Also, it allows the scalability of the system. That is, if one tested application is of interest for the whole system, it can easily be converted into a new agent of the architecture, to serve as basis for new applications more complex.

Moreover, it implements another concept, the resources sharing. This means that all the elements of one of the robots of the team can be easily accessed by all the other robots of the teams by means of the agents that control each of the other robots.

Finally, Acromovi architecture allows the quick development of multirobot applications with the advantages of the multiagent systems. The users need very little time to develop new applications for the team of robots. In just a few weeks, a student with very low Java skills is able to program agents and develop the necessary testing procedures. In this aspect also is important the first concept of the framework, the reusability of the agents previously implemented.

For these reasons, this architecture is very appropriate for the implementation and execution of tasks that require collaboration or coordination by part of the robots of a team.

As future work, it has been now implemented a new application for the team of robots. This application tries to form multirobot formations. Taking the example of the robot following, robots in line formation, it is possible that each follower robot could internally add a fixed displacement to its leader position. As can be in Fig. 11, each robot computes a virtual point instead of the leader position. These virtual points can be calculated simply displacing the leader position in the correct form. Now, the followers do not follow the leader, but these virtual points.

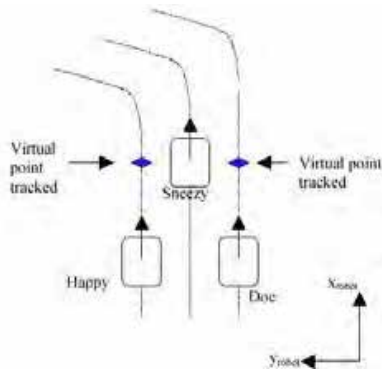


Fig. 11. Triangular multirobot formation using virtual points.

At the end, an interesting application domain would be the use of a team of robots to perform automatic, distributed surveillance tasks in indoor environments, or tasks of vigilance in buildings like factories.

7. Acknowledgements

Support for this research is provided in part by “Ministerio de Educación y Ciencia”, grant DPI2005-08203-C02-01, and by “Generalitat Valenciana”, grant GV05/137.

8. References

- Arai, T.; Pagello, E. & Parker, L. E. (2002). Guest Editorial: Advances in MultiRobot Systems. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, October 2002, 655-661, 1042-296X.
- Cáceres, D.A.; Martínez, H.; Zamora, M.A. & Balibrea, L.M.T. (2003). A Real-Time Framework for Robotics Software, *Proceedings of Int. Conf. on Computer Integrated Manufacturing (CIM-03)*, Wisla (Poland), 2003.
- Caloud, P., et al. (1990). Indoor Automation with many Mobile Robots, *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'90)*, pp. 67-72, Ibaraki (Japan), July 1990.
- Cao, Y.U.; Fukunaga, A.S. & Kahng, A.B. (1997). Cooperative Mobile Robotics: Antecedents and Directions. *Autonomous Robots*, Vol. 4, No. 1, March 1997, 7-23, 0929-5593.
- Enderle, S.; Utz, H.; Sablatnög, S.; Simon, S.; Kraetzschmar, G.K. & Palm, G. (2002). Miro: Middleware for Mobile Robot Applications. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 4, August 2002, 493-497, 1042-296X.
- Fukuda, T. & Iritani, G. (1995). Construction Mechanism of Group Behavior with Cooperation, *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'95)*, pp. 535-542, 0-8186-7108-4, Pittsburgh (USA), August 1995.
- Johnson, J. & Sugisaka, M. (2000). Complexity Science for the Design of Swarm Robot Control Systems, *Proceedings of 26th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pp. 695-700, 0-7803-6456-2, Nagoya (Japan), October 2000.
- Jung, D. & Zelinsky, A. (1999). An Architecture for Distributed Cooperative Planning in a Behaviour-based Multirobot System. *Journal of Robotics and Autonomous Systems (RA&S)*, Vol. 26, No. 2/3, February 1999, 149-174, 0921-8890.
- Mataric, M. J. (1998). New Directions: Robotics: Coordination and Learning in Multirobot Systems. *IEEE Intelligent Systems*, Vol. 13, No. 2, April 1998, 6-8, 1094-7167.
- Nebot, P. & Cervera, E. (2005a). A Framework for the Development of Cooperative Robotic Applications, *Proceedings of 12th International Conference on Advanced Robotics (ICAR'05)*, pp. 901-906, 0-7803-9178-0, Seattle (USA), July 2005.
- Nebot, P. & Cervera, E. (2005b). Optical Flow Navigation over Acromovi Architecture, *Proceedings of 15th International Symposium on Measurement and Control in Robotics (ISMCR 2005)*, Brussels (Belgium), November 2005.
- Oller, A.; del Acebo, E. & de la Rosa, J.L.L. (1999). Architecture for Co-operative Dynamical Physical Systems, *Proceedings of 9th European Workshop on Multi-Agent Systems (MAAMAW'99)*, Valencia (Spain), July 1999.
- Parker, L. E. (1998). ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 2, April 1998, 220-240, 0882-4967.
- Parker, L.E. (2000). Current State of the Art in Distributed Autonomous Mobile Robotics, *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS2000)*, pp. 3-14, 4-431-70295-4, Knoxville (USA), October 2000.

- Picault, S. & Drogoul, A. (2000). The Microbes Project, an Experimental Approach towards Open Collective Robotics, *Proceedings of 5th International Symposium on Distributed Autonomous Robotic Systems (DARS'2000)*, pp. 481-482, 4-431-70295-4, Knoxville (USA), October 2000.
- Renaud, P.; Cervera, E. & Martinet, P. (2004). Towards a Reliable Vision-based Mobile Robot Formation Control. *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS04)*, pp. 3176-3181, 0-7803-8464-4, Sendai (Japan), September 2004.

Multi-Robot Systems and Distributed Intelligence: The ETHNOS Approach to Heterogeneity

Antonio Sgorbissa
University of Genova
Italy

1. Introduction

In the summer of 1999 the Azzurra Robot Team wins the second prize in the RoboCup competition in Stockholm. In the semi-finals, ART defeats the illustrious CS Freiburg (world champion in Paris 1998, Melbourne 2000, and Seattle 2001) in an unforgettable match that almost causes a heart attack in the supporters of either team. However, in the finals, ART is in difficulty against the very strong and quick team from Sharif CE University, Iran. Sharif players, which have been defeated by ART in the eliminatory rounds, seem to have learnt the lesson. Taking advantage of their undoubted superiority in mechanics, and the robustness of their vision and control algorithms (which, incredibly, run in MSDOS, the OS image being loaded from a bootable disquette), Sharif players are able to easily dribble ART players and to score three times before the end of the match, thus winning the first prize with a final score of 3-1.

In contrast with this preamble, RoboCup is not the subject of the discussion here: ART is rather taken as a case study of a "successful" Multi Robot system with very unique characteristics. This is perfectly in accordance with the declared purposes of RoboCup; that is, favouring scientific and technological advancements in robotics and embedded AI with the final aim of transferring the achieved results to real-world applications (Kitano et al., 1998). The choice of robotic soccer as a test field for the design of "autonomous agents" is justified by the intuition that - in most real-world scenarios - a step ahead towards Multi Robot systems is fundamental, if one wants to address issues such as efficiency, fault tolerance, quick response to user requests, or simply consider tasks which cannot be achieved by a single robot. Some examples are autonomous surveillance, rescue operations in large-scale disasters such as earthquakes or fires, humanitarian demining, cooperative manipulation or transportations of objects and furniture, autonomous explorations on desert areas, abandoned mines, or remote planets. To fully take benefit of the larger number of available resources, coordination is important if not essential; the underlying assumption in RoboCup is that these issues can be in part investigated by trying to coordinate a team of soccer-playing robots.

In this context, ART uniqueness is not due to the robots' perceptual skills, the mechanisms adopted for behaviour selection and execution, their coordination, localization, or planning capabilities. Its peculiarity is rather the extreme heterogeneity of the team itself: differently from the usual approach, in which a team is designed and built within a single University or

research laboratory, ART has been the only “National” team in the history of RoboCup, being composed of different robotic players developed in six Italian universities autonomously from each other (Rome – team leader, Genova, Milano, Padova, Palermo, Parma). This gave birth to players which were very different both in the hardware and software architecture (see Fig. 1), and was further accentuated by the fact that research groups involved in ART agreed only partially (and sometimes had very passionate debates) on the philosophy according to which autonomous agents should be designed and built. Heterogeneity in Multi Robot systems have been widely investigated in literature, and journal special issues have been specifically dedicated to the subject (an attempt to formalize behavioural heterogeneity can be found in Balch, 2000); the main difference is that, in ART, heterogeneity was not a choice to develop autonomous agents with specialized functionalities, but rather a very strong constraint to deal with. In spite of this, after training together for less than one week before RoboCup, and some minor changes in the onboard software and hardware, the robots managed to successfully cooperate and achieve a significant result.



Fig. 1. From the left: TinoZoff (the goalie), Rullit, Homer.

In the chapter, I argue that one of the main responsible of ART success in 1999 was the middleware adopted by the research units as a starting platform to develop robotic players: ETHNOS, a Real Time extension to the standard Linux kernel which provides a general architectural framework for the design of distributed robotic applications. When comparing it with other architectural approaches, the most tangible difference consists in the fact that ETHNOS does not make any cognitive assumption (e.g., behaviour based, hybrid deliberative-reactive, etc.). In fact, it should be more likely considered as an “Operating System” for mobile robotics; even if, literally, it is not an OS since it is implemented on top of the standard Posix API, the founding idea is to identify common needs and characteristics in mobile robotic systems, and to provide “ETHNOS system calls” to decompose their inherent complexity without annoying the developer with a pre-established cognitive architecture. As a side effect of its versatility, students from different Universities involved in ART, even when writing their own algorithms in complete autarchy and in total contrast with the healthy principle of code reuse, started nevertheless to talk a common language thanks to ETHNOS; an important achievement, if one considers the educational objectives of RoboCup. Even more important, after 1999 many Italian research groups continued to use the system, both for RoboCup and in different scenarios: among the others the Golem team, which won the second prize in Melbourne 2000, and was completely designed by undergraduate students autonomously from the University (De Pascalis et al., 2001). Nowadays many programming frameworks for robotics share a similar philosophy. In spite of this, ETHNOS still contains original elements: in the next Sections, these issues will be addressed in depth. First, Multi Robot systems in literature will be reviewed under different perspectives;

out of the hundreds of articles on the subject, we select a representative subset for sake of brevity. Second, ETHNOS will be described, by outlining its relevance in determining the success of the Azzurra Robot Team in RoboCup99; design choices will be justified through theoretical investigation and experimental validation. The last part of the chapter will address our recent research, where the ETHNOS multi-robot paradigm is extended in order to include a broader conception of Multi Robot and Multi Agent coordination: the deployment of robot teams in intelligent environments, where they cooperate with a network of fixed intelligent devices for the achievement of common objectives.

2. Related Work

In the last years, Multi-Robot systems are receiving an increased attention in the scientific community. In part, this is due to the fact that many problems related to the control of a single robot have been – at least partially – solved, and researchers start to look at the massive introduction of mobile systems in real-world domains. In this perspective, Multi Robot systems are an obvious choice for all those applications which implicitly take benefit of redundancy; i.e., applications in which, even in absence of a coordination strategy, having more robots working in parallel improves the system's fault tolerance, reduces the time required to execute tasks, guarantees a higher service availability and a quicker response to user requests. There are many cases for which this holds: autonomous cleaning (Jung & Zelinski, 2000; Kurazume et al., 2000); tour guiding in museums, art-galleries, or exhibitions (Siegwart et al., 2003); surveillance and patrolling (Rybski et al., 2002; Vidal et al., 2002), rescue in large scale disaster such as fires, floods, earthquakes (Bahadori et al., 2005); landmine detection (Acar et al., 2003); autonomous exploration and mapping, possibly with the additional constraint of retrieving samples or objects of interests (Rekleitis et al., 2000; Burgard et al., 2005). All the previous applications share some common characteristics: they require the execution of a set of loosely coupled tasks more or less comparable to "coverage", i.e., the problem of visiting the largest number of locations in a given area (Fontan & Mataric, 1996; Choset, 2001), and "foraging", i.e., the problem of finding objects or persons of interest ("food", in the metaphor) and returning them to a specified location (Arkin, 1992; Drgoul & Ferber, 1993). If we ignore issues related to the interference between robots operating in the same area (Fontan & Mataric, 1996; Gustafson & Gustafson, 2006), coverage and foraging obviously improve, when more robots operate in parallel, even in absence of coordination; however, coordination through implicit or explicit communication can provide a significant aid (Balch & Arkin, 1994).

When taking into account more complex scenarios, Multi Robot systems are no more an option; consider, for example, a team of robots trying to achieve an objective which cannot be attained by a single robot, or maintaining a constant spatial relationship between each other in order to execute a task more effectively. Examples of the first type are cooperative manipulation and transportation (Berman et al., 2003; Yamashita et al., 2003), pushing (Mataric et al., 1995; Parker, 1998), or other tightly coupled tasks; examples of the second type are moving in formation or in a convoy for patrolling or transportation, a strategy which allow team members to improve security by protecting each other against attacks, or supporting each other in a different way (Balch & Arkin, 1998; Beard et al., 2001; Barfoot & Clark, 2004). In these and similar scenarios, coordination becomes fundamental, introducing new challenges and opening new possibilities (Balch & Parker, 2002). One of the first issues to be faced is whether coordination should be formalized and solved through a centralized

control process, which computes outputs to be fed to actuators to perform a coordinated motion of all the components of the system, or distributed, with each member of the team taking decisions in autonomy on the basis of its own sensorial inputs, its internal state and – if available – information exchanged with other robots (for sake of brevity, we purposely include in the same class “decentralized” and “distributed” approaches). Whereas central approaches often lead to an optimal solution, at least in a statistical sense, distributed approaches usually lack the necessary information to optimally solve planning & control problems. They are nevertheless more efficient to deal with real-world, non structured scenarios: since decisions are taken locally, the system is still able to work when communication is not available, and – in general – allow a quicker, real-time response and a higher fault-tolerance in a dynamically changing environment. Most teams in RoboCup share these considerations, even if counterexamples exist (Weigel et al., 2002). Notice also that, a part from optimality, univocally accepted metrics or criteria to compare the performance of centralized and distributed Multi Robot systems are missing; the problem is raised, for example, in (Schneider, 2005).

The dual problem is cooperative sensing, in which robots share their perceptual information in order to build a more complete and reliable representation of the environment and the system state. The most notable example is collaborative localization and mapping (CLAM, see Fox et al., 2000; Schmitt et al., 2002; Madhavan et al., 2004), which has been recently formalized in the same statistical framework as simultaneous localization and mapping (SLAM), as an extension of the single robot problem. However, the general idea is older and different approaches exist in literature, often relying on a sharp division of roles within the team (Kurazume et al., 2000). Cooperative sensing can include tracking targets with multiple observers (Parker, 2002; Werger & Mataric, 2000), helping each other in reaching target locations (Sgorbissa & Arkin, 2003), or the pursuit of an evader with multiple pursuers (Vidal et al., 2002). Notice also that cooperative sensing often relies on cooperative motion, and therefore they cannot be considered as totally disjoint classes of problems.

When considering Multi Robot coordination at a higher level of abstraction, architectural and organizational issues become of primary importance. In (Brotan et al., 2006), software architectures for robotics are classified as “reference” or “emergent” architectures. Reference architectures usually implement a cognitive architecture, and therefore pose very tight constraints on how a complex system should be decomposed into interacting components; depending on the cognitive assumptions made, these can be – from time to time – behaviours, motor schema, functional blocks, agents of different nature, etc. Reference architectures can be more targeted to single robot systems (Brooks, 1986; Bonasso et al., 1997; Konolige & Myers, 1998), Multi Robot (Parker, 1998; Pirjanian et al., 2000; Gerkey & Mataric, 2002) or both (Arkin, 1990; Simmons et al., 2002). When specifically designed for Multi Robot systems, reference architectures address such issues as the allocation of tasks to multiple robots, each possibly having different hardware and software capabilities, dynamic role assignment to perform a joint task (Parker, 1998; Gerkey & Mataric, 2002; Stone & Veloso, 1999), route planning with multiple vehicles and goals (Brumitt et al. 2001), etc. Indexes considered to evaluate an approach are near-optimality of the assignment, stability of the solution in presence of fluctuations in the environmental conditions, smooth performance degradation when resources are temporarily unavailable, robustness in presence of uncertain and incomplete information. Behaviour-based, auction based, or similar distributed approaches (with or without explicit communication) are very common; this happens for the reasons already

discussed, not last the fact that optimality in task and role assignment is computationally very expensive to achieve, and consequently inappropriate for real-time operation. Emergent architectures, on the opposite, do not make strong assumptions on cognitive aspects. They rather specify the internal organization and the interactions between software processes, providing a framework to decompose complexity into simpler software components; for this reason, they are also referred to as “Middleware” or “Software Frameworks” for robotics. Services provided range from inter-process communications, support for code reusability and fast-prototyping, hardware simulation. Examples are Player/Stage (Gerkey et al., 2003), MIRO (Utz et al., 2002), Marie (Coté et al., 2006), CLARATy (Nesnas et al. 2006), SmartSoft (Schlegel 2006). A joint effort in this sense is the OROCOS project: www.oroocos.org), whereas code reusability and portability in robotics is especially addressed in (Alami et al., 2000).

RoboCup offers many opportunities to investigate all the issues discussed so far. Synchronised actions, such as passing the ball to each other until a goal is finally attempted, require the robots to execute tightly coupled tasks; Multi Robot formations can be considered - as in human play - as a mean to improve the effectiveness of game strategies, enabling robots to support each other when attacking or defending their own goal; the new trend in collaborative localization and sensing has been significantly speeded-up by research in RoboCup, as well as task/role allocation strategies and algorithms according to different architectural approaches. In this general context, we mostly address architectural aspects, i.e. we investigate a software architecture for:

- Decomposing the complexity of a single robot player into basic software components, providing services for real-time scheduling and analysis, real-time inter-process communications, code reusability.
- Decomposing the complexity of a Multi Robot system into a distributed team of autonomous robots, providing support for inter-robot communication in a real-time scheduling context, and addressing specific issues such as real-time task allocation and role assignment in a distributed fashion.

Under the following constraints:

- The software architecture must adapt to a team of heterogeneous robots which differ significantly both in the hardware and in the software, allowing to implement different cognitive architectures according to the approaches adopted by the various research units (hence it is an “emergent” architecture).
- Inter-robot communication and coordination protocols must be robust to changing environmental conditions in a very dynamic and “hostile” environment, allowing smooth performance degradation in the case that inter-robot communication or robots themselves are temporarily unavailable.

Cooperative motion and sensing are not explicitly addressed; however, some architectural choices determine the appearance of unforeseen cooperative behaviours, even if in a rudimentary form. On one side, this is nothing more but another point in favour of the claim that intelligent behaviour is an emergent property; the interesting result is that, in the case of ETHNOS, intelligence seems to emerge from OS design choices, even in absence of cognitive assumptions on the nature of ongoing processes, their interaction with the environment, and their mutual interactions. Whether this means that every architecture is inherently “cognitive”, is a question that would deserve a deeper discussion. In the following, we limit ourselves to discuss ETHNOS in details, by leaving the reader to draw her own conclusions.

3. ETHNOS

ETHNOS (Expert Tribe in a Hybrid Network Operating System¹) is a framework for the development of Multi Robot systems. It is composed of:

- A dedicated distributed real-time operating system, supporting different communication, execution, and representation requirements; these functionalities are built on top of a Posix compliant Linux kernel.
- A dedicated network communication protocol designed both for single robot and Multi Robot systems, taking noisy wireless communication into account.
- An object oriented Application Programming Interface (API) based on the C++ language (and a subset based on Java).
- Additional development tools (a simulator, a Java-applet template, etc.).

The reference architecture of a single robot, and consequently of the ETHNOS operating system, is entirely based on the concept of “experts”, concurrent agents responsible for specific activities. Experts in ETHNOS can be organised in groups, possibly distributed in a computer network, depending on their computational characteristics: the type of cognitive activity carried out, duration, type of data managed, etc. However these groups do not have a hierarchical organisation but, on the contrary, are executed in a flat model in such a way that the real-time constraints are not violated. This generality in the expert specification allows the use of ETHNOS with different architectural choices without loosing real-time analysis, execution and inter-robot communication support. This aspect has been very important in ART, in which:

- Bart and Homer robots (Padova) relied on a behaviour-based model with arbitration.
- Relè (Genova) adopted a hybrid deliberative/reactive model.
- Rullit robot (Milan) was based on fuzzy logic control.
- TinoZoff goalkeeper (Parma) was based on a central controller.
- TotTino and RonalTino (Rome) were developed in the Saphira architecture, and rely on ETHNOS only for inter-robot communication and coordination.

The next paragraphs will deal with these properties in greater detail.

3.1 Inter-Expert and Inter-Robot Communication

ETHNOS allows the robots to be programmed in such a way that the different experts can be integrated, during development but also at run time, with little or no intervention in the software of the expert itself, thus facilitating both rapid prototyping and dynamic architectural reconfiguration. The first property allows the development of a robotic application, from a set of basic components or behaviours, even by non-highly specialised programmers (for industrial purposes but also for didactic activity in student projects, particularly relevant in RoboCup). The second property allows the system to switch at run-time from a configuration in which it is performing a certain task (for example in which the robot is attacking and thus the active experts are responsible of ball pushing, path planning, and obstacle avoidance) to a different configuration (for example in which the robot is defending and the active experts are responsible of opponent-tracking, ball interception, etc.). These properties are achieved by exploiting a suitable inter-expert communication protocol which comprises of three different

¹In ETHNOS the expression “expert tribe” draws an analogy between a robotic system and a tribe composed of many experts (concurrent agents) in different fields (perception, actuation, planning, etc.). The term “hybrid” indicates the support to the integration of cognitive agents of different nature: deliberative, reactive, subsymbolic, etc; the term “network” refers to the possibility of distributing components on a network. For documentation refer to <http://www.ethnos.laboratorium.dist.unige.it>.

communication methods (a comparison between communication patterns and their relevance in component-based robotics can be found in Schlegel, 2006):

- Asynchronous message-based communication.
- Synchronous access to an expert (similar to function calling).
- Access to a shared representation.

The first method is the most general of the three and it is at the base of ETHNOS applications, hence it is the only one discussed here. It is a message-based communication protocol (EIEP – Expert Information Exchange Protocol) fully integrated with the expert scheduler. EIEP encourages the system developer to de-couple the different experts in execution to reach the limit situation in which the single expert is not aware of what and how many other experts are running; in this way, an expert can be added or removed at run-time without altering the other components of the system. Expert de-coupling is achieved by eliminating any static communication link: EIEP is an asynchronous message-based method in which the single expert “subscribes” to the desired message types, known a priori. When another expert “publishes” any message of the subscribed types, the subscriber receives it transparently. In this way, an expert does not know, explicitly, who the sender of a message is or to which other experts, if any, its message is being distributed. Moreover, the same principles apply also if planning & control activities are distributed in a computer network (for example, to deal separately with on-board reactive components and remote high-level components responsible of computationally intensive tasks: Fig. 2 on the left).

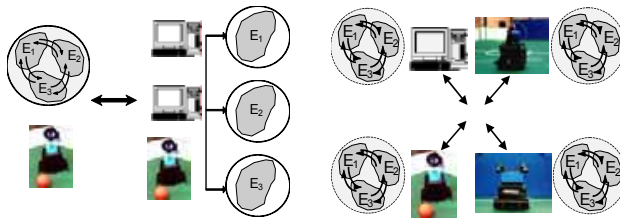


Fig. 2. Left: Experts E_1 , E_2 , E_3 are distributed in a LAN. Right: multi-robot configuration.

EIEP provides also transparent inter-robot communication support. Using the same publish/subscribe principle, messages can be exchanged on the basis of their content not only within the same robot but also among different robots (e.g., players within the same team). However, in inter-robot communication, it is often important to distinguish between internal messages (messages to be distributed within the experts implementing a single player) and external messages (messages to be sent from a player to another) to avoid the explosion of unnecessary network data communication (e.g., commands to be issued to actuators or sonar readings are meaningful only locally). In ETHNOS the different experts are allowed to subscribe to “communication clubs”; we may envisage a single club to which the different players belong or even different clubs, to which only players which are performing a cooperative activity belong, for example the one which carries the ball toward the goal and the one which is supporting it. Message subscription and publication can thus be distinguished in internal (possibly distributed), internal and external in a specific club, or internal and external in all clubs. Again, it is the responsibility of the system to transparently distribute the messages to the appropriate receivers; Fig. 2 on the right shows robots that communicate in a single club, to which all of them have subscribed together with an external supervisor. Because of EIEP, whenever we want to add or remove a player, it is not necessary to explicitly inform other

players about the modifications in the team's composition. This has been very important in ART in which there were more than four types of robots available, with different characteristics of play, and thus different solutions were selected for the single matches and also modified during the matches to better contrast the opponent.

Finally, in wireless communication, transmission packets are sometimes lost due to noise or interference. In this context, both TCP/IP and UDP/IP cannot be used: the former because it is intrinsically not efficient in a noisy environment; the latter because it does not guarantee the arrival of a message, nor provides any information on whether it has arrived or not (for efficiency reasons, ETHNOS communication is built on top of the standard Unix Socket interface). To deal with this, ETHNOS comprises a protocol called EEUDP (Ethnos Extended UDP). EEUDP allows the transmission of messages with different priorities. The minimum priority corresponds to the basic UDP (there is no guarantee on the message arrival) and should be used for data of little importance or data that is frequently updated (for example the robot position in the environment, that is periodically published). The maximum priority is similar to TCP because the message is sent until its reception is acknowledged. However, it differs because it does not guarantee that the order of arrival of the different messages is identical to the order in which they have been sent (irrelevant in ETHNOS applications because every message is independent of the others), which is the cause of the major TCP overhead. Different in-between priorities allow the re-transmission of a message until its reception is acknowledged for different time periods (i.e. 5 ms, 10 ms, etc.). Notice that, when sending messages with the minimum priority, EEUDP is an efficient implementation of MailBoxes in a distributed scenario, an asynchronous communication method particularly suited to real-time applications (Buttazzo, 1997) since it does not delay the execution of hard-real tasks (even if Ethernet-based communication – in a strict sense – is not real-time, since the time required for packet-delivering is not predictable).

3.2. Real-Time Expert Scheduling

ETHNOS supports real-time analysis and execution of experts. Throughout the chapter, the term real-time is used in its formal sense, by referring to the mathematical theory of real-time scheduling and communication as depicted, for example, in (Buttazzo, 1997). This is not a common approach in mobile robotic architectures (with some exception, e.g. Musliner et al., 1993; Rybsky et al., 2003); however, we argue that schedulability analyses assume primary importance whenever computational resources are limited, which is exactly the case for RoboCup (and all other systems which have constraints on their dimensions, weight, an on-board power supply). Suppose that one needs to understand the cause of an observed anomalous behaviour: e.g., the robot stands still whereas we would expect it to move towards the ball. Where does this behaviour come from? Is it a bug in the code, or a temporary CPU overload that delays time critical tasks? Verifying that each task, when taken singularly, meets deadlines is not sufficient; real-time scheduling analyses allow to consider the set of tasks as a whole, and to know in advance if what we are asking the CPU to do is feasible at all.

Before discussing ETHNOS scheduling policies, we qualitatively analyse the timing requirements of RoboCup players. Results can be summarized as follows:

- Sensor acquisition is handled by periodic tasks. They are executed at a high frequency, their computational time is usually low and predictable: e.g., reading encoders; reading sonar or laser rangefinders; acquiring video, etc.
- Reactive planning & control activities are periodic as well. Frequency is high, computational time is low and predictable: e.g., computing speed & jog values for reaching or pushing the ball; avoiding obstacles; updating odometry, etc.

- Data filtering, segmentation, and aggregation activities are periodic or sporadic (their period depends on sensor acquisition) with a computational time at least one order of magnitude greater, but still predictable: e.g., detection of color blobs; building maps; localization of important features, etc.
- Inter-robot communication is aperiodic, with a non-predictable computational time; in a distributed context, a robot does not know in advance when and for how long it will receive messages from teammates.
- Aperiodic, computationally intensive activities can be optionally considered, whose time cannot be easily predicted but - usually - is orders of magnitude greater: collaborative self-localization, symbolic planning, etc.

Notice that only the former four activities are critical for the system, whereas the latter class includes activities that increase performance, but are not critical. An overall architecture for the development of RoboCup players must take these aspects into account and thus permit the integration of less computationally intensive activities (whose execution is critical for the system, and therefore have hard real-time requirements) with more computational intensive ones (whose execution is not critical for the system, and therefore have not real-time requirements).

Periodic, time bounded, critical activities (Sensor acquisition, Reactive planning & control, Data filtering, segmentation, and aggregation) are executed in high priority experts according to two possible algorithms: Rate Monotonic or the non-preemptive Earliest Deadline First algorithm (Jeffay et al., 1991). EIEP allows exchanging information between real-time experts asynchronously, thus avoiding unpredictable delays or priority inversion phenomena. Aperiodic, not time-bounded, non-critical activities are executed in low priority experts; they run in the background and communicate their results to real-time experts whenever their outputs can improve the behaviour of the system. Asynchronous communication through EIEP guarantees that background experts do not interfere with the real-time scheduling policy of the system. Inter-robot communication deserves a special attention. In fact, if we want the robot to coordinate through explicit communication, this should be considered a critical activity; however, it cannot be scheduled as the other real-time tasks since we are not allowed to make reasonable assumptions about when and for how long messages are going to be exchanged with other robots. The solution adopted is common in literature for aperiodic real-time tasks: we insert in the system a couple of Periodic Servers (Lehoczy et al., 1992), i.e. real-time periodic tasks which are given the lowest period (hence the minimum latency) and a predefined "capacity" to serve requests from aperiodic tasks. In our case, the capacity of RxServer and TxServer is spent, respectively, for receiving and sending messages. The mechanism guarantees that a predefined amount of CPU time is always devoted to inter-robot communication, thus avoiding starvation. Taking all these considerations into account, ETHNOS implements different possible scheduling policies ranging from two extremes (Fig. 3) in which:

- Real time Experts (periodic, sporadic, and Periodic Servers) are mapped, one to one, into different Posix threads and scheduled as stated by the Rate Monotonic algorithm, whereas background experts are assigned a lower priority and scheduled in time-sharing.
- All real time Experts are mapped into a single high priority thread, which non-preemptively schedules the real-time as stated by the non-preemptive Earliest Deadline First algorithm, whereas background experts are assigned a lower priority and scheduled in time-sharing.

The two extreme solutions (as well as intermediate ones) can be adopted depending on the application considered. In the former, the Rate Monotonic algorithm (chosen because of its ease of implementation in every Posix compliant OS) is exploited, allowing low latencies in expert execution (depending on the granularity of the Posix scheduler). In the latter, low latency is traded off with an increased processor utilisation, since the presence of only one thread for high frequency tasks guarantees a lower context switching rate with respect to a fully pre-emptive scheduler; moreover, it allows to access critical regions without synchronization primitives and related problems.

As well as providing transparent real-time execution (automatic computation of the required priorities) ETHNOS also includes tools for analysing the feasibility of the scheduling. To allow this, in a preliminary phase, the worst execution time for each expert is computed, and continuously updated when the system is running and also across different runs. Whenever the user asks the system to schedule a set of experts with the selected RM or EDF algorithms, ETHNOS acts as follows (Fig. 3 on the right):

- For each expert (periodic or sporadic) the scheduling conditions (for RM or EDF) are tested referring to its worst execution time and the period (for sporadic experts, the Minimum Interrival Time between two consequent runs). The “capacity” of RxServer and TxServer is automatically computed.
- If the scheduling is possible, experts are scheduled according to the selected policy. Periodic experts are scheduled according to their period, whereas sporadic and aperiodic ones are activated whenever they receive a EIEP message to which they have subscribed.
- Should an expert miss its deadline (this is possible because of the approximate execution time analysis) a very simple recovery policy is adopted: the next deadline of the expert is obtained by adding the expert’s period to the end of the last run (not to the end of the previous period). This is equivalent to changing the phase of the expert, saving the processor from computational overload and from the consequent deadline missing by other experts.
- Background experts are scheduled when no other expert is ready for execution. Should we need a temporary intensive activity in background it is a matter of the specific application to reduce real-time tasks activities.

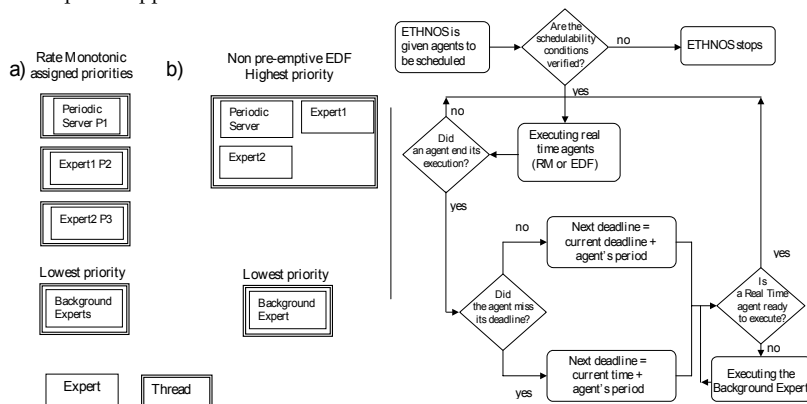


Fig. 3. Left: Mapping experts onto threads with RM and EDF. Right: ETHNOS scheduler.

Some technical considerations: recently, programming frameworks have been proposed relying on Linux RT extensions (e.g., RTAI - www.rtai.org) to exhibit hard real-time characteristics. On the opposite, we have chosen the Linux standard distribution, which follows Posix specifications for real-time (thus being very portable), but have many limitations: in particular, does not allow scheduling tasks at very high frequency (>100Hz) and has significant jitters in task execution. Our choice is justified by the following two considerations. First, most hardware platforms for mobile robotics have dedicated processors or microcontrollers for low-level I/O functions that are executed at a high frequency: the tasks running on the on-board computer have looser requirements (i.e., frequency is \ll 100Hz), and are therefore compatible with the standard Linux kernel capabilities. In our view, it is most important to deal with limitations in computational resources, than guaranteeing latencies below 1ms. Second, Posix compliance guarantees portability, and allows us to easily recompile a new version of ETHNOS as soon as a new version of the underlying Linux kernel is available. ETHNOS currently takes benefit of the 2.6 series of the Linux kernel, which is considerably improved and therefore more adequate to real-time applications: among the others, it is now fully re-entrant, the priority-based internal scheduler works in constant time, and its granularity is one order of magnitude smaller (1ms). An RTAI-enhanced version of ETHNOS designed for the control of hybrid systems, composed of mobile robots and manipulators, has been recently made available.

3.3. Coordination: Explicit Role allocation

Efficient inter-robot communication and real-time scheduling are the building blocks of Multi-Robot coordination in ART (Iocchi et al, 2003), which has the main purpose of

- Selecting the specific activity that each robot has to execute (e.g., attacking, defending, etc.), according to the current situation (i.e., location of the ball and of the opponents) and the currently selected game strategy.
- Adequately distributing players and on the field.

The previous two concepts are synthesized in the concept of "role": roles imply both a subset of activities that the robot is asked to execute, and a territorial division of the playing field. Game strategy is achieved through the concept of "formation"; formations determine which roles, among a set of available ones, must be active in the current situation, and the percentage of robots to be assigned to each active role. Coordination in ART is not oriented to tightly coupled tasks, such as passing the ball to each other, or similar; more similarly to "coverage" and "foraging", the underlying principle is that of taking benefit of redundancy, by reducing interference on the basis of an efficient division of tasks and competencies. Consider the role "striker": it includes a subset of activities aimed toward reaching the ball, dribbling the opponents and the goalie, and finally score (details on how this is done depend on each player's hardware and software). The role "striker", as well as other roles (e.g., "defender", "supporting striker", etc.) must be assigned to the robot which, given the present situation and its intrinsic capabilities, is most suited to it. In this general scenario, and by assuming that inter-robot communication is available, we deal with the following additional requirements and assumptions:

- Smooth degradation: in temporary absence of communication, the robots must be able to perform sub-optimally, on the basis of the information available.
- Heterogeneity: the robots significantly differ in their hardware and software, i.e. they can usually perform the same activities, but in a different way and with different performance.
- Stability: since robots operate in a highly dynamic and hostile environment, roles must be periodically re-assigned; however, role-assignments must be stable in

presence of small fluctuations in the input parameters, to avoid frequent and unproductive changes in robots behaviour.

The strategy adopted in ART is based on the concept of an utility function $f_{i,r}(\dots)$ which returns, for every robot i , a subjective evaluation about how much the robot “feels confident” in assuming role r . If the robot is able to compute $f_{i,r}(\dots)$ for every role which is active in the current formation, and to broadcast the value to other players, a very efficient algorithm for distributed role assignment can be implemented. That is, given:

- A set T composed of n robots t_i .
- A set R composed of m roles r_j which are active in the current formation and ordered according to their priority, i.e., assigning r_j is more important than assigning r_{j+1} .
- A ratio p_j of “required covering” for each role ($0 \leq p_j \leq 1$), which determines the number $n_j = \text{ceil}(p_j \cdot n)$ of robots in the team which must assume role r_j .

Each robot T_i in T acts as follows:

- For each role r_j in R : compute and broadcast $f_{i,j}$.
- For each robot $T_k \neq i$ in T , and for each role r_j : collect the broadcasted values $f_{k,j}$.

At this point, each robot t_i knows how much each robot in the team (included itself) feels confident in assuming role r_j . Next, t_i computes role-robot assignments:

- For each role r_j in R , and for $c=1$ to n_j : assign the role r_j to the robot t_l in T for which $l = \text{argmax}_k(f_{k,r})$; delete t_l from T (to prevent a robot to take many roles).

This obviously requires each robot to estimate how adequate it is to perform a given activity. This is a very critical point; in (Iocchi et al, 2003) details are provided on how $f_{i,j}(\dots)$ is modelled for each role and each robot, depending on its hardware and software characteristics, through an experimental phase in which all the parameters involved must be carefully tuned. Omitting details, $f_{i,j}(\dots)$ is a weighted linear combination of different components, taking into account the variables of the robot state that are considered relevant for each role. Examples are distance from the ball; maximum speed achievable; presence of opponents or teammates in the neighbourhoods; intrinsic sensing or motion capabilities; external or internal causes that can obstruct motion. Each component heuristically measures how close the robot is to achieving the objectives associated to the role; experimentation provides insight on their relative weights in the utility function.

The algorithm heavily relies on ETHNOS communication protocol: each robot t_i transmits updated utility values for each $f_{i,j}(\dots)$ approximately every 100ms (i.e., the period of the RxServer and TxServer in the current implementation). EIEP do not require any static communication link to be established: in role-assignment, this allows each robot to autonomously adapt its behaviour to the number and the identity of robots currently in the field, even in presence of temporary malfunctioning or sudden substitutions in the team composition. In fact, from ETHNOS point of view:

- Every robot subscribes to the MSG_UTILTY_FUNCTION message type.
- Every 100ms, each robot publishes a message containing its utility functions.
- When a robot receives a message from another robot, it stores utility values. If a new message from the same sender is not received in the meanwhile, the last received utility values are kept alive for 1 second (this is important to avoid tightly synchronized communication and to deal with communication noise).
- Every 100ms each robot compares its utility values with those that it has stored in memory, and assigns roles according to the algorithm.

The system exhibits many good properties in a distributed, dynamic environment in which communication is not guaranteed. First, if all robots receive all the utility values broadcasted

by teammates, the task assignment is optimal and – obviously – identical for all robots. Computational complexity of the assignment algorithms is $O(mn)$, with n the number of robots and m the number of roles. (Parker 2000) demonstrates that task assignment for a generic team of heterogeneous robots is NP-hard. This apparent discrepancy derives from the fact that we are not trying to maximise a global evaluation function computed over the whole team; instead, we simply assign the role r_i with the highest priority (out of an ordered set) to the $n_i = \text{ceil}(p_i \cdot n)$ robots which are best suited to it, and iterate the procedure over all the set of available roles. Transmission complexity, which determines bandwidth requirements, is also $O(mn)$; since the number of roles is usually small and lower than the number of robots, the size of each broadcasted messages is small as well, and the bandwidth required is very low. Second, if some or all the robots are temporarily unable to communicate their values, the algorithm shows a very robust, sub-optimal behaviour: each robot proposes an assignment which takes in account only the robots with which it is in contact, considering the highest priority roles first. This leads to the limit situation in which, if communication is completely unavailable, each robot takes the highest priority role, which is very reasonable in most scenarios. Finally, since inter-robot communication is not perfectly synchronized, it can happen that, at a given instant, not all the robot propose the same role assignment. To limit the consequences of this, and to avoid oscillations whenever different robots return comparable “utility values” for the same role, a simple stabilizing mechanism is introduced; when computing the new utility values, each robot receives an additional score for the role that it is currently playing. This has the effect of eliminating small fluctuations through a sort of “hysteresis” which affects the computation of $f_{ij}(\dots)$: being assigned a given role is much harder than conserving it. A parallel with the influence of hysteresis in the assignment of social roles in human communities would be intriguing, but this is not definitely the place for it.

4. Experimental Results

The reliability of EIEP communication has been experimentally verified both in benchmarks and during RoboCup matches. One of ETHNOS peculiarity is the fact that, for network communication, the system allocates a maximum guaranteed and dedicated time (also referred to as capacity) within a couple of expert which are given the highest priority possible (i.e., RxServer and TxServer). The server capacity value is computed automatically on the basis of the schedulability analysis, in such a way that the real-time execution of other experts is also guaranteed; moreover, since servers has the maximum priority, we are guaranteed that, within a predictable and reasonable amount of time, all messages to and from other robots will be sent and received (without considering problems related to communication noise).

In Fig. 4, the four graphs represent different machines (with different processing power) corresponding to three robots (Relè, Homer and Bart) and a coach (a visual interface), connected using wireless Ethernet. The top line in the graph indicates the calculated time available every 100ms for communication purposes; clearly this value decreases as the number of experts in execution increase (and so the computational load). The bottom line indicates the time spent in communication every 100ms; since, for this experiment, we have assumed that the activity of every expert involves either transmission or reception of messages, this value increases with the number of experts. In this way it is always possible to determine a priori whether the system is capable of both communicating information and executing in real-time. For example, if we

consider Relè, the limit situations in which the two lines converge is also the limit beyond which the schedulability analysis fails. Instead, if we consider Bart, real-time scheduling is possible also beyond the intersecting point, but only if we accept communication degradation. Notice that, in real-time systems, an alternative and very common solution to deal with network communication is to execute it in low-priority, non real-time tasks (e.g., QnX, LinuxRT, RTAI). The problem is that, if real-time tasks heavily utilize the CPU, background communication activities can be delayed for an indefinite time, which is definitely not desirable.

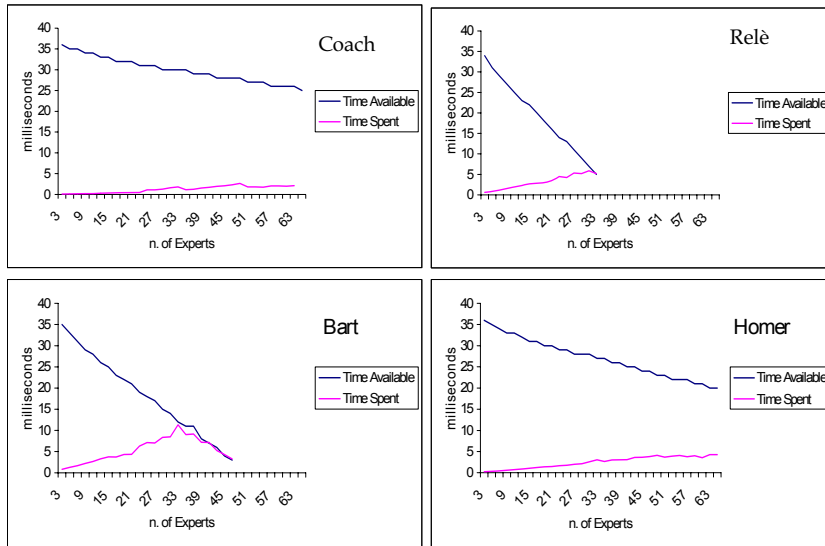


Fig. 4. Network Communication in ETHNOS.

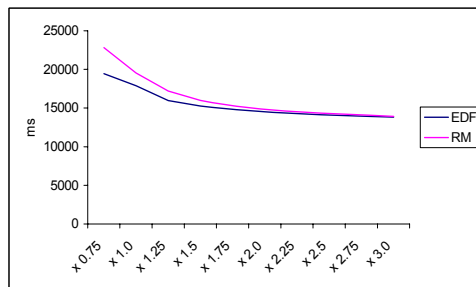


Fig. 5. Time taken to complete operation with different task periods.

To clarify the concept consider Fig. 5, which compares the two scheduling techniques RM and non-preemptive EDF available in ETHNOS. In the experiment, a fixed number of

experts with a pre-defined computational time are scheduled together with a single background expert; in absence of RxServer and Tx Server, the background expert can be devoted to network communication, or other computationally intensive activities (collaborative localization, planning, etc.). The time taken to carry out the background expert is plotted against the factor applied to the base period of the periodic experts; the smaller the factor, the shorter the periods and therefore the higher the processor load. As expected, when the processor load is high, the time required by the background expert to end its computations increases; moreover, the time available significantly depends on the context switches rate, as outlined by the fact that non-preemptive EDF is more efficient than RM in these conditions. As a consequence, communication in background turns out to be very hardly predictable.

Communication in ETHNOS is one the building blocks for distributed role assignment; however, it favours also the emergency of unforeseen coordinated behaviours. EIEP allows to specify whether a message must be delivered only internally, i.e., to experts belonging to the same robot, or externally, i.e., to all robots which belong to the same communication club. An example of the first type is `MSG_SPEEDJOG`, which is published by the expert which generates the trajectory but it is obviously delivered only to local motion experts. Examples of the second type are `MSG_UTILITY_FUNCTION`, or `MSG_STARTSTOP`, the latter used to notify all the robots that the match has started or has been suspended. In experiments performed during the one-week training phase before RoboCup, we programmed robots to share externally also messages of type `MSG_BALL_PERCEPTION`, which contain the perceived position of the ball in a fixed reference frame. Notice that, at that time, the information was meant to be used by a graphical monitor to visualize the current situation in the field for debugging purposes; during RoboCup, there were attempts to use the same mechanism to fuse perceptions on the basis of each robot reliability, but this was only partially accomplished since research units had the unconscious tendency to attribute a very low "reliability" to other robots with respect to their own, therefore invalidating the mechanisms (this should give you the feeling of how "heterogeneous" ART'99 was).

During an experiment, while debugging Bart with the graphical monitor, we observed the following behaviour, which was subsequently repeated and recorded: Bart is provided with a camera, but nobody has pressed the button that enables motors, and consequently the robot cannot move. Homer lies unused in a corner, its camera has been borrowed for some other robot, and somebody forgot its motors switched on. As the start message is published, Bart stands still, and Homer - to our surprise - tries to reach the ball in its place. Obviously, since Homer is not localized with respect to the same reference frame than Bart, Bart's perceptions are not sufficient to correctly reach the ball; we thus repeat the experiment after opportunely positioning Homer in the field (the experiment was evocatively baptized "the limp and the blind"). This time, Homer uses the vision messages sent by Bart to correctly locate the ball, it reaches it, and finally it pushes the ball towards the opponent goal and scores. In Fig. 6, we have emphasised the messages produced by the experts running in Homer and Bart during a 100ms interval in two different situations: Homer going to the ball and Homer carrying the ball. In Fig. 7 the experts in execution and the type of messages distributed are listed. A part of these messages is shared in broadcast (implicitly on the basis of the EIEP) on the local network (lines in bold in Fig. 6) and can therefore be received by the other robots or by a supervision workstation. Notice that only Bart's expert n. 4 (ETVision) produces vision messages (`MSG_BALL_PERCEPTION`). Such messages are received by Homer and, in particular, by the expert n. 3 (ETArbiter). In the absence of analogously locally produced messages,

ETArbiter uses them to activate different behaviours (experts ETGoToBall, ETCarryBall, ETNearWall etc.) depending on the situation in which the robot is in.

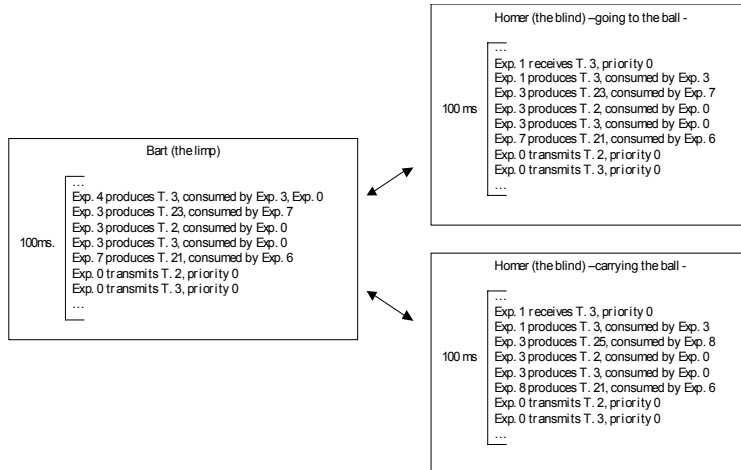


Fig. 6. "The limp and the blind": message exchanged between Bart and Homer.

Experimental data of the role-assignment algorithm have been collected by University of Rome during the European championship in year 2000; log files, compared with the a visual review of the game, show how roles are assigned to robots depending on the number of players currently available in the field. For example, when only two robots are working, the most important roles, i.e., "Striker" and "Defender", are correctly assigned, whereas the role "Supporting" is left unassigned. Fig. 8 on the left shows the percentage of time in which a role is assigned to at least one robot during different matches. Fig. 8 on the right shows the percentage of time in which at least one robot was present in the Forward, Middle and Backward zone. It can be noticed that, even if no explicit coverage algorithm has been implemented, yet the robot are reasonably distributed all over the field, with at least one robot almost constantly defending the goal in the Backward zone. As a result of the stabilizing mechanism, other analyses show that, on average, robots change role every 10 seconds. Some major problems come to the fact that the utility functions are subjectively evaluated, and it is possible that a robot overestimate its ability to play a role without being really suited to it. Consider a robot which incorrectly sees the ball very close, and therefore improperly takes the role "Striker"; the coordinated behaviour of the team turns out to be very sensitive to individual faults in hardware and algorithms. To obviate that, we exploit a mechanism similar to "acquiescence" described in (Parker, 1998); a metric is introduced to measure the individual success of each robot in playing its role, and a factor is consequently introduced in the utility function to take into account the progress made. Once again, the metric to be adopted is task dependent, and it is heuristically chosen through experimental validation. In case of a "Striker", we can take in account the perceived distance from the ball, which must decrease - on average - when the robot is approaching it, or the distance from the goal. Experiments show that this very simple mechanism allows reducing the effect of wrong role-assignments, thus improving the performance of the robot coordination algorithm.

System Experts			
Number	Description	<*>	<^>
0	RxServer	P	100000
1	TxServer	P	100000

User Defined Experts			
Number	Description	<*>	<^>
2	ETPioneer	P	100000
3	ETArbiter	P	100000
4	ETVision	P	100000
5	ETFindBall	SP	100000
6	ETMove	SP	100000
7	ETGoToBall	SP	100000
8	ETCarryBall	SP	100000
9	ETNearWall	SP	100000

* - Periodic (P), Sporadic (SP)
^ - Period or minimum inter-arrival time

Internal Messages	
Description	Type Number
ActivateMove	21
ActivateKick	22
ActivateGoToBall	23
ActivateFindBall	24
ActivateCarryBall	25
ActivateNearWall	26

Internal and External Messages	
Description	Type Number
MSG_STARTSTOP	1
MSG_POSITION	2
MSG_BALL_PERCEPTION	3
MSG_OPPONENTS_PERCEPTION	4
MSG_GOAL_PERCEPTION	5

Fig. 7. Left: active experts on Bart and Homer. Right: message types.

Match	Striker (%)	Supporting (%)	Defender (%)
1	82.9	39.5	98.2
2	84.6	98.0	80.8
3	87.6	38.5	90.0
4	89.5	81.8	96.9
5	93.5	84.1	98.9
Avg	87.6	68.3	93.0

Match	Forward (%)	Middle (%)	Backward (%)
1	78.8	68.7	97.8
2	65.4	68.8	98.6
3	53.2	54.5	99.3
4	23.5	80.4	93.1
5	64.1	72.1	98.9
Avg	57.0	68.9	97.5

Fig. 8. Left: Statistics for role assignment. Right: Statistics for each area coverage.

Finally, ETHNOS has been widely exploited in the ART team from the software engineering perspective, providing support for code reuse, component-based design, run-time debugging at a behavioural level, etc. Despite all the differences in robots hardware and software and the large number of people involved, expert de-coupling and object orientation have allowed to easily put together components developed by different people in the same group and, in a limited number of cases, across different groups. Whenever this happened, expert real-time scheduling and analysis allowed evaluating solutions in different architectural contexts, whereas inter-expert and inter-robot communication protocols allowed programmers to ignore details on how information is exchanged at a lower level. Even if we cannot provide a quantitative evaluation of ETHNOS "usability" in this sense, the importance of these properties was particularly evident in the one-week training phase before RoboCup.

5. Future works

In these years, ETHNOS properties have been exploited in many other robotic systems. Fig. 9 shows the robot Staffetta, a platform which has been especially designed for autonomous transportation within hospitals, but has been also adapted to different indoor scenarios and applications such as tour-guiding in museums and public expositions, autonomous mobile

surveillance, etc. By extending all the concepts that have been introduced so far, we now face all the problems related to autonomy in a fully distributed Multi Agent perspective. Robots are thought of as mobile physical agents within an intelligent environment (we metaphorically call it an "Artificial Ecosystem - AE") where they coexist and cooperate both with other robots and with several fixed physical agents, controlled by hundreds of micro-processors and connected through a building automation network. Fixed agents are intelligent sensing/actuating devices with different roles: they handle automated doors and elevators, provide an aid for autonomous navigation and localization, detect emergency situations such as fires, or liquid leak, etc.; in mobile surveillance applications, they handle passive infrared sensors, RFID readers to distinguish allowed personnel from intruders, cameras, etc.



Fig. 9. Left: Staffetta at Gaslini Hospital. Right: Staffetta at Villanova d'Albenga Airport.

There are both theoretical and practical justifications to this approach; if one consider existing mobile robotic systems in literature, only few of them have proven to be really autonomous in a generic, non structured environment, i.e. able to work continuously, for a long period of time, carrying out its tasks with no performance degradation or human intervention - the "six months between missteps" that (Moravec, 1999) advocates. Following (Brooks, 1990), an explanation can be given: the kind of intelligence we can expect from biological beings heavily relies on the sensors and actuators they are provided with, and the way in which they evolved in time in tight relation with the environment where they live. Given this assumption, there is no wonder that attempts fail, when trying to build an autonomous artificial being which is able to "live" in the same environments where biological beings live: the current technology in sensors and actuators is not adequate to such environments. Thus, we can choose between two different approaches: reproducing biological beings (which implies reproducing also their sensors and actuators) or building robots which have limited sensing and actuating abilities, but which operate within an environment which is well-suited for them. We claim that, whereas part of the current research aims at building very expensive and complex humanoid robots, it is possible - and, in the short time, preferable - to put robots back in their habitat, which allows to build simpler, cheaper, and more reliable autonomous systems. Ambient Intelligence and related disciplines are nowadays receiving a great attention by the scientific

community, and the idea of using cameras or other sensors to track robots in the environment has been proposed; however, the problem is rarely faced in an integrated perspective, by addressing – as we do – the architecture of a system composed of many robots and many fixed sensors distributed in the environment (recent exceptions are Broxvall et. al, 2006; NRF).

In the Artificial Ecosystem, both robots and intelligent devices are handled by ETHNOS experts, which communicate on a common message board through the publish/subscribe EIEP and are given a high level of autonomy; since autonomy and intelligence are not only a characteristic of robots, but are distributed throughout the environment (Fig. 10 on the left), we say that robots are autonomous but not “autarchic”. Next, we extend the concept of intelligent devices to the sensors and actuators on board the robot, allowing to control the robot at two different levels (thus increasing fault-tolerance, Fig. 10 on the right): on the lower level a higher reactivity is achieved through on-board intelligent devices which control sensors and actuators; on a higher level, sensorial inputs are collected by experts running on the onboard computer and performing more sophisticated computations before issuing control to actuators.

For on-board and off-board devices we rely on fieldbus technology and, in particular, Echelon LonWorks, a standard for building automation: experts are executed on LonWorks Neuron Chips, microprocessors with a low computational power and a dedicated OS. ETHNOS publish/subscribe protocol is implemented in LonWorks through the concept of “network variables”, i.e. strongly typified data which are propagated on the bus and delivered to all devices which have subscribed to that kind of information (the “network variables” abstraction is provided by the Application Layer of the LonWorks communication protocol). All the control nodes in the system (i.e. sensors/actuators both on board the robot and distributed in the building) are connected to the same fieldbus, by adopting a low bandwidth infrared channel for the communication between on-board devices and the building network.

The architectural infrastructure of the system has been extensively tested, and some case studies are available for evaluation; our past set-up at the Gaslini Hospital in Genova (years 2000–2002), the mobile surveillance system ANSER at the Villanova d’Albenga airport, and a permanent set-up in our Department at University of Genova (Fig. 9). In these scenarios, we mostly implemented devices for self-localization (“intelligent beacons”) and for the control of automated doors and elevators (“intelligent gatekeepers”). However, to fully take benefit of the approach and draw conclusions about its properties, more work has to be done. Visionary scenarios include: advanced coordinated behaviours, involving both on-board and off-board intelligent devices; multipurpose intelligent devices, able to assume different roles and perform different tasks according to a given allocation strategy; a parasite robot which relies on the sensors of other robots, possibly without them being aware of that; a robot which is “possessed” by another robot or by an intelligent device, which takes control of its actuators and forces it to execute a given task; robots and intelligent devices which ask (or force...) humans to help them to execute a given task; etc. This “ubiquitous distribution” of competencies and capabilities in the environment, possibly with humans being involved, has not been investigated yet; the Artificial Ecosystem, however, is totally compatible and open to this new paradigm, thus definitely encouraging to explore it.

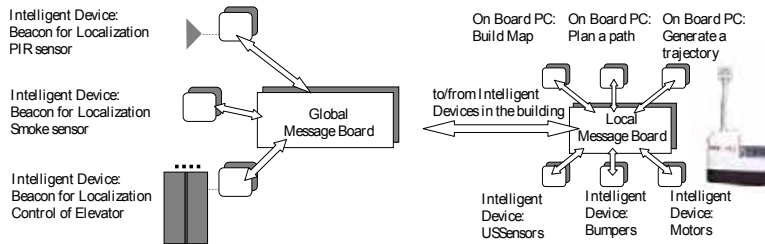


Fig. 10. Left: Intelligent devices in the building; Right: Intelligent devices on the robot.

6. Conclusions

We presented ETHNOS, a software architecture and a programming environment for the real-time control of distributed Multi Robot systems; its properties have been described as well as its influence on the development of a soccer team for RoboCup. Throughout the chapter, we have argued that the ART team in Stockholm 1999 has been an unique experience because of its extreme heterogeneity: players in the team differed both in the hardware and in the software, and were designed in total autonomy by different research units with almost no cooperation; however, after only one week of training before RoboCup, they were able to coordinate themselves on the soccer field and to win the second prize. ETHNOS contribution to achieving this is discussed.

In particular, to meet the expectations of all the research group involved, with many different point of view about robotic intelligence, ETHNOS does not imposes a specific cognitive architecture. In fact, it rather specifies the internal organization of software processes and their interactions, providing a framework to decompose complex Multi Robot systems into simpler software components. For this reasons, it is also referred to as an “Operating System” for robotics. All the design choices made have been discussed in details, providing both theoretical justifications and experimental validations. The system’s versatility is outlined as well, by showing how the traditional Multi-Robot paradigm can be extended to include a broader conception of Multi-Robot and Multi-Agent coordination: the Artificial Ecosystem approach, which takes inspiration from research in Ambient Intelligence to propose a new paradigm for the design of efficient, reliable, and autonomous (but not autarchic) mobile robotic systems.

7. Acknowledgements

I wish to thank Prof. Zaccaria and Dr. Piaggio, with whom I designed ETHNOS; Prof. Nardi, coordinator of ART in 1999 and 2000, Prof. Adorni, Prof. Bonarini, Prof. Chella, Prof. Pagello, Prof. Iocchi and all the students of the Universities of Rome, Parma, Milano, Palermo, and Padova for their significant contribution in the experimentation of ETHNOS, and for some of the data and the photos shown here.

8. References

- Acar, E.U.; Choset, H.; Zhang, Y. & Schervish, M.S. (2003). Path Planning for Robotic Demining: Robust Sensor-based Coverage of Unstructured Environments and Probabilistic Methods. *Int. Journal of Robotic Research*, Vol. 22, No. 7-8, 441-466

- Alami, R.; Chatila, R.; Fleury, S.; Herrb, M.; Lnggrand, F.; Khatib, M.; Morisset, B.; Moutarlier, P. & Simeon, T. Around the lab in 40 days, *Proc. of the IEEE Int. Conference on Robotics Automation*, pp. 88–94, April 2000, San Francisco, CA.
- Arkin, R.C. (1990). Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, Vol. 6, No. 1-2, 105–122.
- Arkin, R.C. (1992). Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, Vol. 9, No. 3, 351–364.
- Bahadori, S.; Calisi, D.; Censi, A.; Farinelli, A.; Grisetti, G.; Iocchi, L. & Nardi, D. (2005). Autonomous systems for search and rescue. *Rescue Robotics, A Birk, S. Carpin, D. Nardi, Jacoff A., and S. Tadokoro (Eds.)*, Springer-Verlag, 2005.
- Balch, T. & Arkin, R.C. (1994). Communication in Reactive Multiagent Robotic Systems, *Autonomous Robots*, Vol. 1, No. 1, 27–52.
- Balch, T. & Arkin, R.C. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, 926–939.
- Balch, T. (2000). Hierarchic Social Entropy: An Information Theoretic Measure of Robot Group Diversity, *Autonomous Robots*, Vol. 8, No. 3, 209–237.
- Balch, T. & Parker, L.E. (2002). *Robot Teams: From Diversity to Polymorphism*, T. Balch and L.E. Parker (Eds.), A K Peters Ltd.
- Barfoot, T. & Clark, C. (2004). Motion Planning for Formations of Mobile Robots. *Journal of Robotics and Autonomous Systems*, Vol. 46, No. 2, 65–78.
- Beard, R. W.; Lawton, J. & Hadaegh, F.Y. (2001). A feedback architecture for formation control. *IEEE Transactions on Control System Technology*, Vol. 9, 777–790.
- Berman, S.; Edan, Y. & Jamshidi, M. (2003). Navigation of decentralized autonomous automatic guided vehicles in material handling. *IEEE Transactions on Robotics and Automation*, Vol. 19, No. 4, 743–749.
- Bonasso, R.; Firby, R.; Gat, E.; Kortenkamp, D.; Miller, D. & Slack, M. (1997). Experiences with and architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9, No. 2, 237–256.
- Brooks, R.A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, 14–23.
- Brooks, R.A. (1990). Elephants Don't Play Chess. *Journal of Robotics and Autonomous Systems*, Vol. 6, No. 1-2.
- Brotten, G.; Monckton, S.; Giesbrecht, J. & Collier, J. (2006). Software Systems for Robotics An Applied Research Perspective. *International Journal of Advanced Robotic Systems*, Vol. 3, No. 1, 11–16.
- Broxvall, M.; Gritti, M.; Saffiotti, A.; Seo, B.S. & Cho, Y.J. PEIS ecology: Integrating robots into smart environments, *Proc. of the 2006 Int. Conf. on Robotics and Automation*, pp 212–218, May 2006, Orlando, FL.
- Brumitt, B.; Stentz, A.; Herbert, M., & the CMU UGV Group (2001). Autonomous Driving with Concurrent Goals and Multiple Vehicles: Mission Planning and Architecture, *Autonomous Robots*, Vol. 11, No. 2, 103–115.
- Burgard, W.; Moors, M.; Stachniss, C. & Schneider, F. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, Vol. 1, No 3, 376–385.
- Buttazzo, G. C. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.
- Choset, H. (2001). Coverage for robotics—A survey on recent results. *Annals of Mathematics and Artificial Intelligence*, Vol. 31, 113–126.

- Côté, C.; Brosseau, Y.; Létourneau, D.; Raïevsky, C. & Michaud, F. (2006). Robotic Software Integration Using MARIE. *Int. Journal of Advanced Robotic Systems*, Vol. 3, No. 1, 55-60
- De Pascalis, P; Ferraresso, M; Lorenzetti, M; Modolo, A; Peluso, M; Polesel, R; Rosati, R.; Scattolin, N.; Speranzon, A. & Zanette, W. (2001). Golem Team in Middle-Sized Robots League, *RoboCup 2000: Robot Soccer World Cup IV, Lecture Notes In Computer Science, Vol. 2019*, Springer-Verlag, London.
- Drgoul, A. & Ferber, J. (1993). From tom thumb to the dockers: Some experiments with foraging robots. *From Animals to Animats 2: Proceedings of the 2nd Int. Conf. on Simulation of Adaptive Behavior*, pp 451-459.
- Fontan, M.S. & Mataric, M.J. (1996). A study of territoriality: The role of critical mass in adaptive task division. *From Animals to Animats 4: Proceedings of the 4th Int. Conf. on Simulation of Adaptive Behavior*, MIT Press, Cambridge, MA.
- Fox, D; Burgard, W.; Kruppa., H. & Thrun., S. (2000). A Probabilistic Approach to Collaborative Multi-Robot Localization. *Autonomous Robots*, Vol. 8, No. 3, 325-344.
- Gerkey, B. & Mataric, M. (2002). Sold! auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, 758-768.
- Gerkey, B.; Vaughan, R. T. & Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. *Proc. of the 11th Int. Conf. on Advanced Robotics*, pp. 317-323.
- Gustafson S. & Gustafson, D.A. (2006). Issues in the scaling of multi-robot systems for general problem Solving, *Autonomous Robots*, Vol. 20, No. 2, 125-136.
- Iocchi, L; Nardi, D.; Piaggio, M. & Sgorbissa, A. (2003). Distributed Coordination in Heterogeneous Multi-Robot Systems. *Autonomous Robots*, Vol. 15, No. 2, 155-168.
- Jeffay, K.; Stanat, D.F. & Martel, C.U. (1991) On Non-preemptive Scheduling of Periodic and Sporadic Tasks, *IEEE Real time Systems Symp.*, pp 121-139, December 1991.
- Jung, D. & Zelinski, A. (2000). Grounded Symbolic Communication between Heterogeneous Cooperating Robots. *Autonomous Robots*, Vol. 8, No. 3, 269-292.
- Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa, E. & Matsubara, H. (1997). Robocup: A challenging problem for AI and robotics, in *RoboCup-97: Robot Soccer World Cup I. Lecture Notes in Artificial Intelligence, Vol. 1395*. Springer-Verlag, New York, 1998, 1-19.
- Konolige, K. & Myers, K. (1998). The saphira architecture for autonomous mobile robots. *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems, D. Kortenkamp, RP Bonasso, and R. Murphy (Eds.)*, MIT Press, March 1998.
- Kurazume, R., & Hirose, S. (2000). Development of a Cleaning Robot System with Cooperative Positioning System, *Autonomous Robots*, Vol. 9, No. 3, 237-246.
- Lehoczky, J. P.; L. Sha; & J. K. Strosnider. (1992). Enhanced Aperiodic Responsiveness in Hard RealTime Environments. *Proceedings of the Real Time Systems Symposium*, pp. 110-123, , December 1992, Phoenix, AZ.
- Madhavan R.; Fregene K. & Parker, L.E. (2004). Distributed Cooperative Outdoor Multirobot Localization and Mapping. *Autonomous Robots*, Vol. 17, No. 1, 23-39.
- Mataric, M.J.; Nilsson, M. & Simsarian, K.T. (1995). Cooperative multi-robot box-pushing, *Proc. of the Int. Conf. on Intelligent Robots and Systems*, pp. 556-561, May 1995, Pittsburgh, PA.

- Moravec, H., Rise of the Robots, *Scientific American*, December 1999, pp. 124-135
- Musliner, D.; Durfee, E. & Shin, K. (1993). Circa: A cooperative, intelligent, real-time control architecture. *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 23, No. 6.
- Nesnas, I. A.D.; Simmons, R.; Gaines, D.; Kunz., C.; Diaz-Calderon, A.; Estlin, T.; Madison, R.; Guineau, J.; McHenry, M.; Shu, I-H. & Apfelbaum, D. (2006) CLARAty: Challenges and Steps Toward Reusable Robotic Software, *International Journal of Advanced Robotic Systems*, Vol. 3, No. 1, 23-30.
- NRF, Network Robot Forum, www.scat.or.jp/nrf/english/.
- Parker, L.E. (1998). Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 2, 220-240.
- Parker, L.E. (2000). Current state of the art in distributed robot systems. *Distributed Autonomous Systems 4*, L.E. Parker, G. Bekey and J. Barhen (Eds.), Springer-Verlag.
- Parker, L.E. (2002). Distributed Algorithms for Multi-Robot Observation of Multiple Moving Targets. *Autonomous Robots*, Vol. 12, No. 3, 231-255.
- Pirjanian, P.; Huntsberger, T.; Trebi-Ollennu, A.; Aghazarian, H.; Das, H.; Joshi, S. & Schenker, P. (2000). Campout: a control architecture for multi-robot planetary outposts. *Proc. of the SPIE Symposium on Sensor Fusion and Decentralized Control in Robotic Systems III*, pp. 221-230, Boston, MA.
- Rekleitis, I.M.; Dudek, G. & Milius, E.E. (2000). Graph-based exploration using multiple robots. *Distributed Autonomous Robotics Systems 4*, L.E. Parker, G.W. Bekey, and J. Barhen (Eds.), Springer-Verlag, Berlin.
- Rybski, P. E.; Stoeter, S. A.; Gini, M.; Hougen, D. F.; & Papanikolopoulos, N. (2002). Performance of a distributed robotic system using shared communications channels. *IEEE Trans. on Robotics and Automation*, Vol. 22, No. 5, 713--727.
- Schlegel, C. (2006). Communication Patterns as Key Towards Component-Based Robotics, *International Journal of Advanced Robotic Systems*, Vol. 3, No. 1, 49-54.
- Schmitt, T.; Hanek, R.; Beetz, M.; Buck, S. & Radig, B. (2002). Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, 670-684.
- Schneider, F.E.; Wildermuth D. & Kräußling, A. (2005). Discussion of Exemplary Metrics for Multi-Robot Systems for Formation Navigation. *International Journal of Advanced Robotic Systems*, Vol.2, No. 4, 345-353.
- Sgorbissa, A. & Arkin, R.C. (2003). Local Navigation Strategies for a Team of Robots, *Robotica*, Vol. 21, No. 5, 461-473.
- Siegwart, R.; Arras, K.O.; Bouabdallah, S.; Burnier, D.; Froidevaux, G.; Greppin, X.; Jensen, B.; Lorotte, A.; Mayor, L.; Meisser, M.; Philippsen, R.; Pignet, R.; Ramel, G.; Terrien, G.; & Tomatis, N. (2003). Robox at Expo.02: A LargeScale Installation of Personal Robots. *Robotics and Autonomous Systems*, Vol. 42, No. 3-4, 203-222.
- Simmons, R.; Smith, T.; Dias, M.B.; Goldberg, D.; Hershberger, D.; Zlot, R. & Stentz, A. (2002) A layered architecture for coordination of mobile robots. *Multi-Robot Systems: From Swarms to Intelligent Automata*, A. Schultz and L. Parker (Eds.), Kluwer Academic Publisher, May 2002.
- Stone, P. & Veloso, M. (1999). Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, Vol. 110, No. 2, 241-273.
- Utz, H.; Sablatnög, S.; Enderle, S. & Kraetzschmar, G. (2002). Miro—Middleware for Mobile Robot Applications, *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 4, 493-497.

- Vidal, R.; Shakernia, O.; Jin, H.; Hyunchul, D.; & Sastry, S. (2002). Probabilistic Pursuit-Evasion Games: Theory, Implementation, and Experimental Evaluation. *IEEE Trans. Robotics and Automation*, Vol. 18, No. 5, 662- 669.
- Weigel, T.; Gutmann, J.-S.; Dietl, M.; Kleiner, A. & Nebel, B. (2002). CS freiburg: Coordinating robots for successful soccer playing. *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 5, 685-699.
- Werger, B.B. & Matarić, M.J. (2000). Broadcast of local eligibility for multitarget observation, *Distributed Autonomous Robotic Systems 4*, L. E. Parker, G. Bekey, and J. Barhen (Eds.), Springer-Verlag.
- Yamashita, A.; Arai, T.; Ota, J. & Asama, H. (2003). Motion planning of multiple mobile robots for Cooperative manipulation and transportation. *IEEE Transactions on Robotics and Automation*, Vol. 19, No. 2, 223-237.

Force Sensing for Multi-legged Walking Robots: Theory and Experiments – Part 1: Overview and Force Sensing

A. Schneider, U. Schmucker
*Fraunhofer Institute for Factory Operation and Automation
Germany,*

1. Introduction

This paper is intended to direct the attention of researchers and designers of multi-legged walking robots to the problem of force sensing for such vehicles. The use of force information enables such systems to achieve new mechanical properties, increases the reliability and functional capabilities of walking robots and simplifies many control algorithms.

Interest in developing walking machines as vehicles with increased passability has picked up in the last sixty years. Walking machines have a number of principle advantages over wheeled and caterpillar machines:

- They can move over terrains with obstacles up to the size of a leg.
- The movement of their legs provides comfortable motion of the body with cargo and passengers over uneven terrain.
- Ground deformation under their support legs creates less of a mechanical load on the ground than a wheel or caterpillar machine's continuous track (especially important for tundra, mountain and hill slopes, forests, etc.).
- They can work in a complexly-structured environment (sloped, confined work and operation, etc.) and move on consolidating ground and unknown terrain with varying load capacity.
- They can use one or more legs as an auxiliary manipulator.

The advantages of walking machines determine their range of potential applications (Song & Waldron, 1989; Okhotsimsky *et al.*, 1992; Berns, 2006).

In order to achieve foot force distribution by any method, the vehicle legs have to be equipped with *force sensors*, as shown in Fig. 1.1, and *force feedback* should be introduced into the control system (Golubev *et al.*, 1979; McGhee *et al.*, 1980; Klein & Briggs, 1980; Devjanin *et al.*, 1982).

Basic areas of research on motion control for walking robots where *the use of force control is necessary* can be specified as follows:

A. Local Regulation and Gait Cycle Correction of Legs

This section describes the organization of an adaptive step cycle for each leg to step over an unknown obstacle by means of touching movements (Gurfinkel *et al.*, 1981; Devjanin *et al.*, 1983; Steuer & Pfeiffer, 1998; Schmucker, 2002).

Most step cycle organization is based on trajectories of movement of the end of a leg and has rigidly programmed kinematics. Modifying a step cycle (support and a transfer phases) on the basis of a leg's contact to an environment (tactile force sensor signals) significantly expands the possibilities to adapt a leg to the roughness of a support surface. An adaptive step cycle has been developed for the legs of a six-legged robot "Masha" (Devjanin *et al.*, 1983).

As the leg touches the ground, the vertical movement is stopped and the trajectory of the foot is modified so that the time of the complete cycle remains constant.

B. Force Motion Control of Legs in Interaction with a Support Surface

One of the central problems of motion control is the distribution of force between legs and the organization of robot motion within margins of static stability. Support reactions need to be controlled during movements on rough terrain and to increase ground passability.

In terms of the foot forces acting on its legs, a multi-legged walking vehicle is a *statically indeterminate* mechanical system. For example, the horizontal components of support reactions cannot be determined in a support phase on three legs. If the number of supporting legs is greater than three, then the distribution of vertical force components cannot be determined.

Actively distributing foot force reactions makes it possible:

- to reduce loads on the vehicle structure and the *energy consumption* of leg drives,
- to increase vehicle passability on a surface with an insufficient load-bearing capability (e.g., sand),
- to generate locomotion over rough terrain and to overcome large obstacles (high ledges, deep holes, etc.),
- to provide control of the horizontal foot force components so that contact forces are within friction cones and
- to minimize the risk of foot slip.

The possibility of force distribution control in a walking vehicle has been in discussion for over twenty-five years. There are a number of approaches to the force distribution problem based on different criteria. At the same time, the problem of optimally distributing support reactions in real time presents significant difficulties. For instance, the movement of the six-legged vehicle consisting of a body and six legs each with two-elements is described by differential equations of the 48th order. Obviously, simplified models need to be developed. Various optimization algorithms for foot force distribution are known. These include: methods based on linear (Orin & Oh, 1981) and square-law programming (Okhot-simsky & Golubev, 1984; Klein & Kittivacharapong, 1990; Cheng & Orin, 1990; Nagy *et al.*, 1992; Marhef & Orin, 1998):

- methods employing additional equations (Kumar & Waldron, 1988),
- methods based on pseudo-inverse approaches (Whitney, 1969; Klein & Wahavisan, 1984; Waldron, 1986; Kumar & Waldron, 1990; Gorinevsky & Shneider, 1990; Gardner, 1992; Lehtinen, 1994; Jiang *et al.*, 2004).

The use of linear or square-law programming to optimize the distribution of support reactions is complicated in practice because of the necessity to calculate the support reactions in real time. Therefore, most publications only contain simulation results.

Pseudo-inverse methods are widely used because they quickly solve foot force distribution in real-time. Most of these methods solve the equilibrium equations in terms of the foot forces, where the pseudo-inverse is used to minimize the sum of squares of the foot forces.

However, the exact condition of energy minimization is more complex (Okhotsimsky & Golubev, 1984).

Pseudo-inverse methods have been presented in connection with the concept of multi-fingered gripping. The *zero interaction force* concept for testing the ASV Hexapod was described by (Waldron, 1986). A more complicated version of *zero interaction force* was presented by (Kumar & Waldron, 1988). The results of computer simulation of vertical foot force distribution for the robot OSU Hexapod while moving by wave gait cycle are presented in e.g. (Klein & Wahavisan, 1984; Schmucker, 2002).

This method has been used to compute the programmed support reactions in (Gorinevsky & Shneider, 1990) and to experimentally demonstrate the distribution of vertical and horizontal foot force reactions for stiff and compliant grounds in real time (see section 4 for more detail).

In (Gardner, 1992), non-linear optimization techniques are used to calculate force distribution that minimizes the maximum ratio of tangential to normal forces at the points of foot contact for the ASV Hexapod. Force distribution is investigated for three tasks (standing on flat terrain, crossing a crest and moving along crest). The force distribution is optimized in such a way that the sum of the squares of the force values is minimized.

The pseudo-inverse method for distributing vertical foot force components in the machine MEGAT1 with fluid drive is described in (Lehtinen, 1994). The results of three pseudo-inverse formulations for real-time optimal foot force distribution (vertical reactions) and the simulation results for a hexapod machine with a tripod gait are presented in (Jiang *et al.*, 2004). The authors developed a method for minimizing the risk of foot slip when there are more than three supporting legs.

C. Force control of robot body maneuvering for service operations

Most research work on the development and application of walking robots deals with their transportation capabilities. However, the use of such robots is not merely limited to the transportation of technological equipment. Another field of application for walking robots is their use as a multi-purpose chassis to perform civil engineering tasks like assembly, repair, emergency and rescue work, etc. They can be used to perform many tasks such as moving the robot in a narrow labyrinth or a pipeline or operation with process equipment, on the robot's body. Operations such as assembly, handling, drilling, repair and machining can be carried out by controlling contact forces caused by interaction with objects. These tasks may be also treated as *constrained motion problems*.

D. Force control of locomotion in problems of climbing and overcoming large obstacles

One of walking robots' basic advantages over wheeled and caterpillar machines is their ability to move over difficult terrain, to overcome larger obstacles by climbing over them and to move over "hummocks", over stones, in a pipe, etc. (Song & Waldron, 1989; Galvez *et al.*, 2001; Kaneko *et al.*, 2002; Golubev & Korianov, 2003; Bretl *et al.*, 2004).

E. Problems of oscillation damping and movement stability

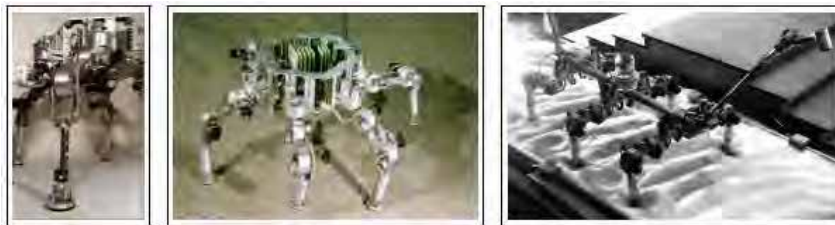
When moving, the walking robot's state may become unstable and nonattenuating oscillations may develop in the system. The most widespread approach to describing walking robots' dynamics and control involves treating the motions of the legs and body separately (Okhotsimsky & Golubev, 1984; Gorinevsky & Shneider, 1987).

However, when the support reactions provide feedback, the control signals in each leg depend on the motion of the robot body and thus on the motions of the other legs.

Other sources of in-system oscillations can be the coefficient of gains, force feedback time delay, force sensor stiffness and body mass (Gorinevsky & Shneider, 1987, 1997; Schneider *et al.*, 2004).

This brief review of the major tasks and problems of motion control, maneuvering, service and other operations reveals that *force sensing* is one of the central problems in engineering highly functional walking robots. Nevertheless and despite the wealth of literature, very few studies have been published on experimental implementation of force control in legged locomotion and other operation.

Unfortunately, many applied problems of multi-legged robot force sensing and force feedback control and problems of force distribution of support reactions in real-time in particular have been insufficiently investigated. At present, it is clear that vehicle legs should be equipped with *force sensors* and that *force feedback* should be introduced into the control circuit.



(a) Three
Component
force sensor

b) Six-legged robot "Katharina"

" (c) Six-legged robot "Masha"

Fig. 1.1. Illustration of three walking machines.

The results presented in this paper are based on the Fraunhofer Institute for Factory Operation and Automation's (IFF) development of the six-legged walking robot

"Katharina" (weight about 25 kg) with a hexagonal body (Fig. 1.1b), which was equipped with an additional manipulator device with two degrees of freedom (Schmucker *et al.*, 1997). Some results were obtained earlier during the development of the "Katharina" prototype, "Masha", a six-legged walking robot (weight about 22 kg) with a rectangular body (Fig. 1.1c) (Gurfinkel *et al.*, 1982). Both vehicles have six legs each with three powered degrees of freedom and equipped with three-component force sensors mounted in the shanks (s. Fig. 1.1a). The manipulator has three-component force sensors as well.

In Section 3 of this paper, we present approaches to the development of force sensors for multi-legged machines. In Section 4, we describe the basic structure of the control system for the robot "Katharina". In Section 5, we consider fundamental force control laws and adduce our own experimental analyses of force control distribution. In Section 6, we report the results of some experiments on locomotion over soft soil. In Section 7, we describe the service operations that have been performed using body displacement and force control.

2. Survey of statically stable walking machines

In the history of scientific research there are many examples of designing robotic devices like mechanical toys and anthropoid robots. For example, in the 4th century in China was

built a mechanical crow. Design studies of Leonardo da Vinci and the development of mechanical dolls with human-like behaviour in the middle age may be regarded as groundwork for legged locomotion. For more information, see Thring, 1983; Todd, 1985; Marsh, 1985; Raibert, 1986; Song & Waldron, 1989; Vukobratovic *et al.*, 1990; Okhotsimsky *et al.*, 1992; Rosheim, 1994; Berns, 2006, and other.

One of first walking mechanisms appeared about 1870. The famous Russian mathematician Chebyshev has developed a *four-legs plantigrade mechanism*. His “foot-walking” machine is a combination of several four-link mechanisms and is the prototype of walking mechanisms constructed on the basis of the so-called trajectory synthesis, where the coordination of extremities movement “is fulfilled in a purely mechanical way” (Chebyshev,1945).

In 1893 George Moore constructed the first biped machine, the so-called *Stearn Man* (s. Rosheim,1994).

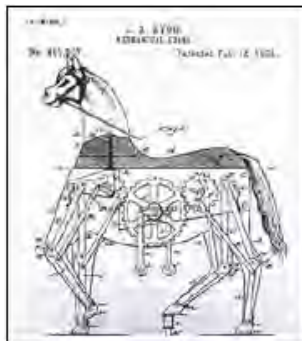


Fig. 2.1. The mechanical Horse.

In the same year, Lewis A. Rygg filed a patent for the construction plans of his *Mechanical Horse* (Fig. 2.1), a four-legged machine with a steering concept similarly to a bike: the driver should move the machine through pedals (Rygg, 1893). However, this machine was never built. The same happened to the *Schreitwagen* of Freiherr von Bechtolsheim designed in 1913. In the following years different walking machines have been constructed, which all were based on the simple mechanical hardware, where the locomotion was reached through a cyclic up and down movement of a mechanism with only one degree of freedom.

Other examples are a mechanism constructed during the First World War (Shigley, 1961), a centipede and a leg driven tractor designed by Thring (Thring, 1983).

2.1 The first steps of development of multi-legged walking machines (from 50ies till 70ies)

The first mechanical constructions were based on a predetermined movement so that an adaptation to the ground was not possible. To obtain more flexibility several researches in fifties started to assign the motion control of the walking machine completely to a human operator by controlling the different degrees of freedom manually.

A lot of research concerning legged locomotion was done from Bekker and his colleagues at the University of Michigan and at the US Army Tank Automotive Center at Warren, Michigan. Bekker has studied land locomotion for many wheels and walking modes (Bekker, 1969).

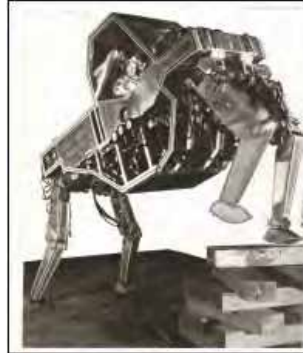


Fig. 2.2. GE Quadruped of Mosher.

One of the most advanced machines was the four legged machine *G.E Quadruped*, the General Electric Walking Truck of R. Mosher (Fig. 2.2). This machine was more than 3 meters high and 3 meter long, with a weight of more than 1.400 kg, and was controlled by operator's hands and feet through a bi-lateral force reflecting hydraulic servo system (Mosher, 1968).

Due to the rapid development in space exploration at the end of the 60ies, the NASA ordered new studies concerning the development of machines for planetary exploration. At this period, six- and eight-legged vehicles moving with tripod- and tetrapod-gait were constructed, like for example the eight-legged *Iron Mule Train* (Todd, 1985; Morrison, 1968). In 1960-1961, Shigley used a hydraulically driven pantographe mechanism to transfer the motion of a horizontal and a vertical actuator. The design had 16 legs, a block of four at each corner (Shigley, 1961). The 70ies are characterized by intensive theoretical research of locomotion of human and animals, mathematical modeling of walking devices on a complex surface, development of a big number of physical models. During this period, the most important results have been received in the research centers of the USA, the USSR and Japan.

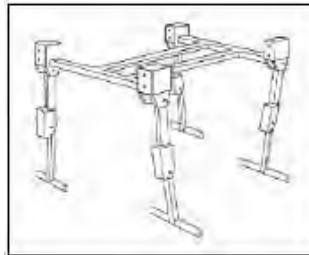


Fig. 2.3. Phony Pony.

In 1966, a four-legged machine named "*Phony Pony*" (Fig. 2.3) was built by Frank and McGhee at the University of Southern California (McGhee, 1976). Each leg had two degrees of freedom. Both joints were actuated by electric motors with gears. Using the method of finite-state control, two types of walking were investigated: the quadruped walk and the quadruped trot. It was only a walking in a straight line. This was the *first legged vehicle which walked autonomously under full computer control*.

Tomovic and McGhee suggested to use the method of finite-state control for a description of a four-legged vehicle movement. In terms of this approach McGhee has developed a four-legged device with pneumatic drives (Tomovic & R, 1966).

In 1968, Morrison constructed a number of six- and eight legged vehicles variants with execution of alternating tripod and tetrapod gaits (Morrison, 1968). Mossi and Peternella built in 1969 at the University of Rome an electrical hexapod. Its legs were unusual in employing telescopic joints for the vertical motion; they were pivoted at the hip to provide the propulsion stroke (Mocci *et al.*, 1973; Peternella & Salinari, 1973).

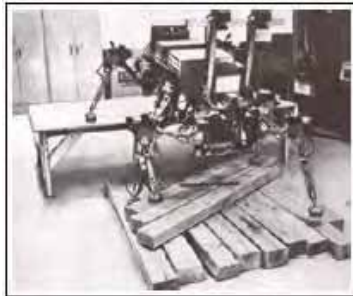


Fig. 2.4. Hexapod of McGhee.

In 1977, McGhee and his group at Ohio State University developed the *OSU-Hexapod*, a six-legged insect-like machine (Fig. 2.4). This vehicle was equipped with force sensors, gyroscopes, proximity sensors and a camera system and was fully controlled by a PDP 11/70. Active force sensing has been used in the control system.

The parameters of the machine are as follows: body length 1,3 m body width 1,4 m, weight about 100 kg, velocity about 0,1 m/s (McGhee, 1977; McGhee & Iswandhi, 1979; Klein & Briggs, 1980).

It has been used for a variety of experiments, like walking with different gaits on a flat surface, climbing up shallow stairs and walking over obstacles (Ozguner *et al.*, 1984). These results in building optimal leg construction and the developed theory of walking layed the basis for the biggest walking machine project at that time. In 1982, Waldron and his group at Ohio State University started with realization of the *Adaptive Suspension Vehicle (ASV)* (Fig. 2.5). The aim of the project was an easily steerable, big and efficient machine that could be used at natural ground conditions (Song & Waldron, 1989).



Fig. 2.5. ASV – The Adaptive Suspension Vehicle.

The ASV is 5,2 m long and 3 m high. It reaches a stride length of 1,8 m and a maximum velocity of 3 m per second. However, the mobility of this machine was considerable. The ASV is able to surmount obstacles up to 1 m and to walk on slopes up to 37%.

The project of development of *Ambler* (Fig. 2.6) started in the late 80th aiming at planetary exploration (Bares *et al.*, 1989; Simmons & Krotkov, 1991).

The researchers intended to construct a machine which is fault tolerant to a break-down of mechanical devices, which possesses an autonomous energy supply and particularly, they wanted to have a machine that could work in a completely autonomous way. The operator of the machine should only prescribe more complicated tasks, but the machine should perform the tasks completely on its own.



Fig. 2.6. Walking machine Ambler.

In 1969, the largest off-road vehicle in the world, a coal-mining dragline called “*Big Muskie*” was built and applied in coal mining.

An eight-legged machine named *ReCUS* was constructed in 1979 and used for underwater measurements of ground profiles for the construction of harbours (Ishino *et al.*, 1983).

For similar applications, the six-legged walking machine *Aquarobots* was developed later (Akozono *et al.*, 1989).

The series of six-legged machines *Odex* with their six legs arranged in a circle are similar to the *Aquarobot*. The first model of the *Odex* was presented in 1983, it was determined for maintenance work in nuclear power plants (Fig. 2.12). The robot was steered in a teleoperation modus by an operator who determined the tasks of the robot (Byrd & DeVries, 1990).



Fig. 2.7. Hexapod Odex.

During this period, a number of industrial and research companies in Japan were developing walking robots for application in various areas of the national economy. During the 80ies several four-legged machines have been developed which were controlled by a simple steering of the machines body, e.g., *Titan 1-1V* (Fig. 2.8) (Hirose, 1984), *PV 11* (Hirose & Umetani, 1980), and *Collie* (Miura *et al.*, 1985; Emura & Arakawa, 1991). The *Titan* series and the *PV-11* are four-legged machines with an insect-like leg construction.

Their automated steering devices make the machines easier to handle rather than a manual steering. However, they showed big disadvantages in terms of moving velocity and the energy efficiency. The robot *Collie 1* had a mammal-like structure, but its moving possibilities have been that restricted, that a dynamically stable walking was possible only in one direction.

The electromechanical four-legged machine *PV-11* has been designed at the Tokyo Institute of Technology. Its weight was 10 kg. The legs are special pantograph mechanisms. A larger model (weight about 150 kg) with similar kinematics has been elaborated and tested (Hirose & Umetani, 1980; Hirose, 1984; Umetani *et al.*, 1985). In 1985, the Laboratory of mechanical Engineering (Japan) designed an electromechanical walking robot *MELWALK3*. Its parameters are as follows: weight 35 kg, body length 0,5 m (Kaneko *et al.*, 1988).



Fig. 2.8. TITAN-2.

Intensive researches have begun at the end of the 60ies in Russia in the field of the theory and development of walking robots of various applications (Artobolevsky & Umnov, 1969). There was published a series of papers about various periodic gaits and optimal gaits for a hexapod (Bessonov & Umnov, 1973).

In 1972, the first Russian walking devices *Ricksha* with two legs has been developed and different gaits have been studied, as well as the interaction between legs (Devjanin *et al.*, 1973; Schneider *et al.*, 1974).

In 1974, a hexapod with legs arranged radially about a central vertical axis was made in the Aviation Instrument Institute in Saint-Petersburg. No information on its tests is available (Ignatyev *et al.*, 1974).

In the early 70s, at the Institute of Applied mathematics (IPM) of the Russian Academy of Sciences in Moscow, a six-legged walking machine controlled on a base of a *mathematical model* of motion control was developed. The machine motion and the terrain were rendered on a display. Motion control algorithms were developed for a walking machine on rugged terrain in both automated and operator control modes. The problems of control were also considered for a dynamic model of a statically stable walking machine (Okhotsimski & Platonov, 1973, 1976).

Two laboratory models of electromechanical walking machines and control systems (1977, 1979) were designed at the Institute of Applied Mathematics (IPM) in collaboration with the

St. Petersburg Mechanical Institute. The machines were tested in supervisory and automated modes.

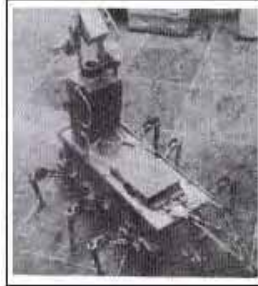


Fig. 2.9. Hexapod IPM.

The *first* six-legged machine was equipped with a laser scanning range finder and is connected with a two-computer system. The walker could move around isolated obstacles which were detected remotely by a scanning distance-measuring system, and could climb over obstacles (see Fig. 2.9).

The parameters of the machine are the follows: body length 0,6 m, body width 0,25 m, weight 56 kg, length of leg 0,4 m, velocity 0,2 m/s (Okhotsimski & Platonov, 1976; Okhotsimski *et al.*, 1978).

A second mechanical model of this robot was developed at the Institute VNIITRASMASH (St. Petersburg). The parameters of the machine are the follows: body length 0,6 m and body width 0,25 m, weight 40 kg, length of leg 0,4 m (Efimov *et al.*, 1982).

The joint researches resulted in the development of the full-scale six-legged *walking rover* controlled by the human-operator (Fig. 2.10) at VNIITRASMASH Institute (St. Petersburg) in 1985. Its weight is 450 kg, the machine is equipped with a generator, storage batteries, a control system, and an armchair with joystick for the operator (Kemurdjian,1995).

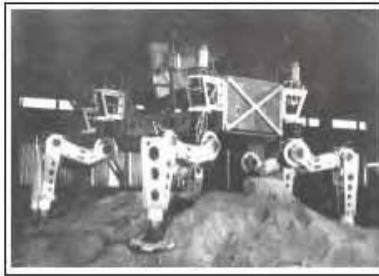


Fig. 2.10. Walking Rover (St. Petersburg).

In 1978, the laboratory prototype of an electro-mechanical walking machine at the Institute of Machine Engineering, Moscow, was able to move at a fixed gait with a given step length and clearance. Its simple control system was based on a PLC-like device. The legs are constructed in an orthogonal kinematics scheme with two degrees of freedom (Umnov, 1981).

At end of the 60ies and at the beginning of the 70ies in Moscow there was constructed the six-legged insect-like walking robot "Masha" (Fig. 2.11).

Robot Masha has six legs, with three powered degrees of freedom each. The control system was based first on a general-purpose analog computer and later on an onboard special-purpose computer. The machine was equipped with a triangular range finder, a vertical gyro and force sensors in each leg which measured the three components of the support reaction and ground contact sensors.



Fig. 2.11. Robot "Masha".

The legs are powered by electric drives with gears and are equipped with joint angle potentiometer sensors.

The triangular optical range finder mounted on the front of the body determined the coordinates of obstacles. A scanning was performed by rotating mirrors. Active force sensing has been used at the control system. The control problems of vehicles moving on deformable grounds were analysed and studied experimentally.

The machine was tested in supervisory (a human operator) and automated (with a range finder) control modes. The locomotion, including by-passing and climbing over obstacles (including the stair-type ones) was controlled automatically. The main geometric and mass parameters of the machine are as follows: the body length 0,7 m and width 0,21 m correspondingly, length of leg 0,28 m, mass is 22 kg, maximal speed 0,25 m/s (Gurfinkel *et al.*, 1981; Gurfinkel *et al.*, 1982; Devjanin *et al.*, 1983).

Present research of multi-legged robots

The 90ies are characterized by a rapid development of theory of multi-legged walking machines, control systems based on classical and on neural networks theory, particularly in the USA, Europe and in Japan. Walking machines are equipped with various sensing systems. Furthermore, AI systems were widely applied to the analysis of an environment and motion of robots on a complex surface.

In 1993, the NASA financed the *Daedalus-project*, which had a similar objective like the *Ambler-project* (Roston, 1994).

The *Dante-project* is one of the few big projects of the 90ies. Under the leadership of Prof. Wittaker two eight-legged robots (*Dante 1 and Dante 2*) for the locomotion on very rough terrain on earth or any other planet have been developed at CMU (Field Robotics Center,

US). *Dante 2* had the task to climb in the crater of the Mound Spurr Vulcano (Alaska, USA) to take measurements of the consistency of gas near the crater bottom.



Fig. 2.12. Dante 2; CMU, USA.

A quadruped machine, intended for the investigation of different gaits of mammal like machines was constructed by Inagaki and Kobayashi (Inagaki & Kobayashi, 1993). Prof. Bekey and his group at University of Southern California, LA developed the quadruped *Meno 2* (Sukhatme & Bekey, 1995).

In the 80-90ies, a big variety of two-, -four- and six-legged walking machines was developed, mostly as research robots for basic investigation of locomotion and behavior, for the construction of walking machines, control systems, and for use in service operations. Features of the developed machines can be found in numerous publications, e.g., in transactions works of annual conferences CLAWAR.

The work of Prof. Cruse and his research group on the walking behavior of the stick insect as well as on the role of neural synapses in control of movements inspired several projects to transfer the results of those studies to walking machines (Cruse *et al.*, 1998).

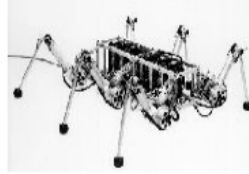
The six-legged walking machine of the Technical University of Munich (TUM), for example, is a result of this idea (Pfeiffer, *et al.*, 1994). The machine is designed and steered similar to a stick insect (Pfeiffer *et al.*, 1995). The control system was realized as a neural structure.

In the 90ies, research and development of multi-legged walking machines were intensively performed in European countries. Some examples of the most known walking machines are shown in Fig. 2.13. Their description and characteristics can be found in the catalogue of walking machines (Berns, 2006) and, e.g., in literature (Pfeiffer *et al.*, 1994; Schmucker *et al.*, 1996; Berns *et al.*, 1998; Roßman & Pfeiffer, 1998; De Santos *et al.*, 2003; Halme *et al.*, 1994; Kirschner & Spennberg, 2001); Arikawa & Hirose, 1996; Hirose *et al.*, 1991; Galvez *et al.*, 2001).

When the speed of walking machines increases, it is more advantageous from energy point of view to pass from statically stable to dynamic modes of motion, as animals do. This transition to dynamic modes makes it possible to increase relative duration of the leg transfer phase, to decrease the frequency of leg oscillations with respect to the body, and to reduce power consumption on the leg motion relative to the body.



(a) Robot *Katharina* (25 kg) with hexagonal body. IFF Magdeburg.



(b) *TUM Walking Machine* (23 kg) with rectangular body. TU, Munich.



(c) *Lauron* (12 kg) with rectangular body. FZI, Karlsruhe.



(d) Four-legged walking machine *SILO4* (67 kg). CSIC, Madrid.



(e) Eight-legged robot *Skorpion* (9,5 kg). Inst. for Autonomous Intelligent Systems (AiS), Sankt Augustin.



(f) Six-legged robot *Silex* (15 kg). Free University, Brussels.



(g) *TUM Pipe Crawling Robot* (20 kg) with 8 legs.



(h) Four-legged robot *TITAN-6* (160 kg). Tokyo Inst. Technology, Japan.



(i) Walking Machine *Fa*. Plustech Oy, Finland

Fig. 2.13. Multi-legged walking machines developed in Europe.



Fig. 2.14. Big Dog of Robotics Boston Dynamics.

This determines the importance of research in mechanics and motion control and in design of legged vehicles with dynamically stable locomotion. A considerable part of the studies dealing with dynamic stable motion of legged vehicles was purely theoretical or was done by methods of mathematical modeling and simulation.

One of the first who constructed walking machines exclusively for investigation of dynamic walking behavior was Reibert. He first started with experiments with one-legged machines in 1983. His *2D*- and *3D-Hopping Machines* are very robust to external disturbances (Raibert, 1986).

As an example, the research company Robotics Boston Dynamics developed a dynamically stable four-legged walking machine *BigDog*, with a size of a greater dog which can bear a cargo in weight up to 55 kg (Fig. 2.14), (<http://www.bostondynamics.com>).

3. Force sensors

Forces and torques are measured by *deformation or displacement* in an elastic mechanical element of a sensor. Various physical principles are employed to convert these quantities into sensor output (Baumann, 1976). The theory of force sensors, design specification, their parameters and engineering methods of calculation are presented for example in (Green & Zerna, 1968; Bray *et al.*, 1990; Gorinevsky *et al.*, 1997). The majority of force sensors established in robotics use strain gauge transducers.

3.1 Fundamentals of force sensors design

At present, only a relatively small number of multi-legged walking robots have foot force sensors and force control systems. Possible reasons might be the great variety of designs of walking vehicles, their different leg and foot structures and differing loads on supporting legs that do not allow the use of commercial force sensors in many cases.

In many cases, gimbals connect the feet to the shin and thus force sensors measure only three components of the support reaction. There are also force sensors that measure five and six components of the support reaction. These are necessary for example for robot climbing of (Galvez *et al.*, 2001).

The basic types of force sensors and their elastic elements are the *bending element* (Fig. 3.1a), *double bending beam element* (Fig. 3.1b), *parallelogram elastic element* (Fig. 3.1c), *Maltese cross*

element with elastic beam support (Fig. 3.1d) and their modifications. Some typical force sensors with one, three and six components were designed for application in legged robots for various carrying capacities. More detailed information on force sensor designs and parameters can be found in (Gorinevsky *et al.*, 1986, 1997). The basic characteristics: *sensitivity, stiffness eigenfrequency, decoupling, linearity, and hysteresis* were used to evaluate and compare the designs developed for force sensors. Strain gauge transducers were used to measure deformations of elastic elements (Fig. 3.1). Many multi-component force sensors are composed of one or two-component modular elements. Such modules should only be sensitive to the component desired from among the multiple components of force and torque acting on it. The modular elements are usually designed to be *mechanically decoupled*, i.e. to have a selective mechanical response to the measured components.

3.2 Design computations for multi-component sensors

3.2.1 Six-component sensors with bending elastic elements

Elastic elements consisting of bending and torsion deformations are typically used to measure moderate loads (less than 1000 N).

We consider the main parameters of the sensors: their sensitivity and stiffness *Sensitivity*. The vector of average strains in strain gauge bridges is linearly dependent on the components of the attached load provided that bending or torsion strain is measured. We can then write $\varepsilon = \frac{L}{Eh^3} \lambda \left[\frac{\bar{F}}{\bar{M}/L} \right]$ where λ is the nondimensional sensitivity matrix of the sensor, L the characteristic beam length, h the beam cross-sectional dimension (thickness) and $\zeta = L/D$ the nondimensional ratio of the characteristic beam length to the end-effector size.

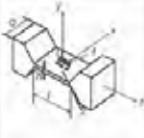
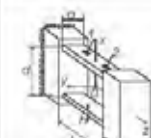
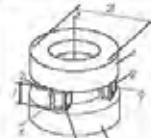
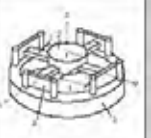
				
	(a) Bending elastic element	(b) Parallelogram module	(c) Compressive-tensile and shear elastic module	(d) Matress cross element
Sensitivity (ε)	$\varepsilon = 6FL / (Eah^2)$	$\varepsilon = Flh_d / (8EI_y)$	$\varepsilon_{\text{tens}}^{(1)} = \bar{r}(1 + \mu)l_{\text{eff}} / (4lsE)$ $\varepsilon_{\text{shear}}^{(1)} = F_z / (4Es)$	$\varepsilon_i = \beta_i \frac{FL}{Eh^3} \geq \varepsilon_0$ $\varepsilon_n = \beta_n \frac{FL}{Eh^3} \geq \varepsilon_0$
Stiffness (δ)	$\delta = 2d / h$	$\delta = \varepsilon l^2 / (3h_d)$ $\nu = 1 / \pi \sqrt{6EI_y g / (Fl^3)}$	$\zeta = \frac{L}{Eh^3} K^{-1} \left[\frac{\bar{F}}{\bar{M}/L} \right]$	$\delta = \frac{\bar{F}}{Eh^3} A l^3$
Eigenfrequency (ν)	$\beta_1 = 2h\Delta / a^2$ $\beta_2 = 2(1 + \mu)\nu$	$\beta_1(M_y) = 8\Delta D l_y / (lh_d I_y)$ $\beta_2(M_y) = 8\eta D l_y / (sl d h_d)$	$\nu = \Omega \cdot f / (2\pi)$	

Fig. 3.1. Basic designs of elastic elements used in forces sensors (Gorinevsky *et al.*, 1997).

{textitStiffness. The external force \bar{F} and torque \bar{M} cause small linear displacement vectors δ and angular displacement vectors φ in the elastic element of the sensor, which are linearly

dependent on \bar{F} and \bar{M} . Let $\xi = [\delta^T, \varphi^T L]^T$ be a six-dimensional vector of generalized coordinates. Then we can write $\xi = \frac{L}{Eh^3} K^{-1} \begin{bmatrix} \bar{F} \\ \bar{M}/L \end{bmatrix}$ where K^{-1} is the nondimensional compliance matrix.

Maltese cross elastic elements (s. Fig. 3.1d) consist of four radial beams of the length l , which connect a rigid hub 2 of the radius r with an outer flange 3. The outer edges of the beams rest upon the flange via membrane joints 4, which restrain the beam ends from rotation and transversal displacement. Thus we can denote $L = r + l, \alpha = r/l$.

3.2.2 Compressive-tensile and shear elastic module

This elastic module is generally employed to measure comparatively large loads (exceeding 500 N). The design of the sensor is shown in Fig.3.1c. Two round flanges 1 are linked by four ($nL, \dots, 4$) elastic struts uniformly positioned along the circumference. The length of each strut is l ; the cross-sectional area is s and the distance $2L$ between the diametrically opposite struts is far greater than the strut dimensions.

3.3 Examples of force sensor designs

3.3.1 Sensors with bending deformation

To measure the longitudinal component of the support reaction not exceeding 500 N, we developed a compact parallelogram elastic module in which, to reduce the cross sizes, force measuring elastic plates are located in the plane parallel to the action of the longitudinal force. Fig. 3.2 shows the circuit of such a module. External (1) and internal (2) flanges are attached to the top of the elastic plates (3, 4); the bottom ends of the plates are connected with a rigid non-deformable crosspiece (5) in which a hole (6) is made on the symmetric axis of the module. A force-transfer element (7) passes through this hole. The element is connected to an internal flange. Strain gauges (8, 9) are mounted along an axial line on the outside of the plates and measure elastic bending deformations.

The module is mechanically insensitive to torques because the strains caused by lateral components of a supporting reaction in the plates have identical signs. Fig. 3.3a and Fig. 3.3b shows the three-component force sensor developed at the Fraunhofer IFF and integrated in legs of the walking robot "Katharina" with a load carrying capacity of about 24 kg.

It consists of a bending beam with a quadratic cross section for measuring lateral components of the support reaction and the module described above for measuring the longitudinal component. The holes are used to form elastic beam plates.

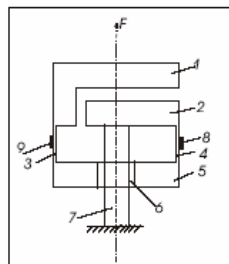


Fig. 3.2. An axial module.

The holes concentrate stress on plates where strain gauges are attached. The module is made of an integral piece of metal, is simple to manufacture and has a small coupling.

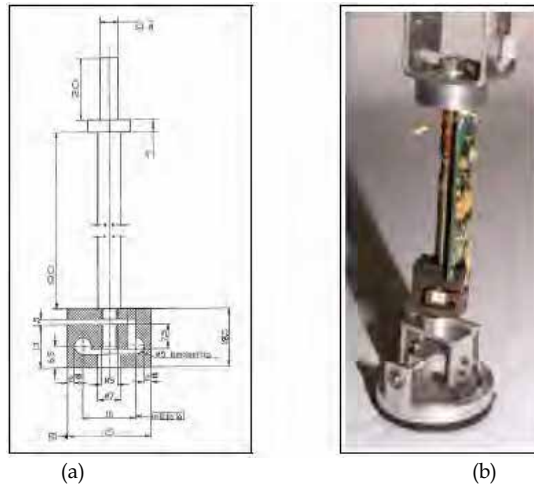


Fig. 3.3. Design of three-component force sensor.

Together with the amplifier, this sensor is mounted in the shank of a leg and the bottom end of the sensor is attached to a gimbal connected to the foot (see Fig. 3.3b). It is designed to be loaded up to 500 N, interference between channels not exceeding 1%. Fig. 3.4 and Fig. 3.5 show the technical solution of the three-component force sensor mounted on a manipulator.

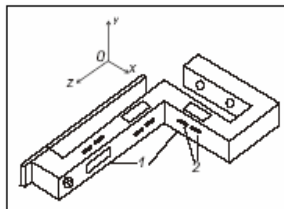


Fig. 3.4. Three-component force sensor composed of parallelogram element.



Fig. 3.5. Three-component force sensor mounted between the arm manipulator and a tool.

3.3.2 Sensor with a compressive-tensile and shear elastic element

Elastic elements of this type are generally employed to measure comparatively large loads (exceeding 500 N).



Fig. 3.6. Three-component force sensor for high loading.

Fig. 3.6 shows the three-component force sensor for the large four-legged walking machine "ALDURO" with a load-carrying capacity of about 700 kg. The sensor is made of a heavy-walled pipe (aluminum alloy D16T) in which four elastic beams have been milled. Strain deformations are measured by eight strain gauge rosettes mounted on lateral surfaces of beams and by four strain gauges mounted on the outsides of each beam. One flange of the sensor is attached to a leg shin, another to the articulated foot joint. Manufactured from an aluminum alloy, the sensor is mounted in the foot of a leg and designed for loads up to 3000 N. The beams have a length of $l = 20\text{mm}$, $r = 30\text{mm}$, a diameter of 90mm and a height of 50mm . The cross-sectional area of the elastic struts measures 126mm^2 .

4. Control system structure

In developing the architecture for the control system, we used data from human and animal biomechanics and the control/organization of their motions. The contributions of N. A. Bernstein, 1967 and his followers (Gurfinkel & Fomin, 1973;

Shik & Orlovsky, 1976) connecting a multilevel organization of control in the locomotion system, a central program of movements and interlevel interaction are essential for engineering a walking robot's control system.

The control system of the robot "Katharina" (Fig. 4.1) consists of upper and lower levels and is based on the idea of the synergy of the quasi-regular gait.

The *upper level* of the control system is supervisory and prescribes such motion parameters as components of linear and angular velocities of the body, gait pattern, track, width, clearance and some cycle parameters of locomotion. Algorithms of the *lower level* are based on the assumption that the vehicle moves slowly enough that its motion can be described kinematically. The *lower level* is made up of a main controller and six leg controllers. The local controllers are connected to the main controller that is connected to a PC by a serial link. All controllers are located inside the robot's body. The **main controller** contains five basic units: master step cycle generator, step cycle modification unit, coordinate transformation unit, sensor signal evaluation and six leg transformation units.

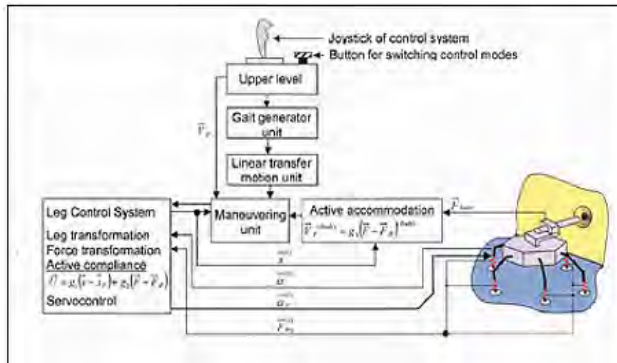


Fig. 4.1. Block-scheme of control system.

The master step cycle generator. The tip of each leg moves relative to the vehicle's body along a closed trajectory called a "step cycle". The unit contains one generator that creates prototypes of plane step cycles for each leg in some auxiliary Cartesian coordinates. The step cycles are interrelated to preserve given relationships between step cycle phases and a sequence of states of the legs.

The step cycle modification unit. This unit has several functions. A global modification subunit modifies global geometric parameters (vertical and horizontal) of step cycles (Devjanin *et al.*, 1983; Schmucker *et al.*, 2002).

An individual modification subunit activates changes for each step cycle individually. A step cycle consists of a support phase and a transfer phase. The transfer phase consists of three subphases: intrinsic transfer phase, lifting phase of a leg from a surface and lowering phase of a leg to a surface. A transition from the transfer phase to the support phase takes place either after the lowering phase finishes or when signals are received from force sensors indicating the moment when the leg encounters an obstacle.

The coordinate transformation unit geometrically links prototype trajectories with the body scheme, scales them and shifts trajectories linearly and angularly in the transfer phase.

Sensor Signal Evaluation. This unit operates with force and angle signals and calculates the algorithms of ground detection, force distribution between the legs locomoting over hard and soft surfaces and obstacle crossing. This unit handles internal and external sensor data collectively.

Six leg transformation units. These units have the six cycle generators as input and produce the programmed trajectories for each leg in Cartesian coordinates of axes $O^{(i)}x_1^{(i)}x_2^{(i)}x_3^{(i)}$ the origins of which coincide with a leg's point of attachment.

The **leg controllers** contain the local leg transformation unit, the sensor transformation unit and the leg control system.

The *local leg transformation unit* transforms Cartesian coordinates of programmed vectors $\vec{r}_p^{(i)}$ into the programmed vectors of joint angles $\vec{\alpha}_p^{(i)}$. The *sensor transformation unit* transforms the force vector $\vec{F}_x^{(i)}$ measured by force sensor in the shanks into vector $\vec{F}_x^{(i)}$ in axes $\vec{x}^{(i)}$ and the vector of measured angles $\vec{\alpha}^{(i)}$ into the vector $\vec{x}^{(i)}$ in Cartesian coordinates. The *leg control unit* consists of the three-dimensional position servo control system for each leg.

The control system algorithms and their technical realization for the robots "Masha" and "Katharina" are described in (Schmucker et al., 1996; Schmucker, 2002).

Each leg of "Katharina" has its own controller based on a microcontroller INTEL 87C196KR which only controls the respective leg. The controller includes the CPU, FLASH-memory and RAM, a PWM output unit (PWM-U), analog input unit (AIU) and a synchronous serial communication unit (SSCU) on the chip. All microcontroller units are connected by a synchronous serial bus.

5. References

- Akozono, J., Iwasaki, M. & Asakura, O.(1989). Development on a walking robot for underwater inspection. In ICAR' 89. Columbus, Ohio, USA.
- Arikawa, K. & Hirose, S.(1996). Development of quadruped walking robot titan. In 8. *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 208-214.
- Artobolevsky, I. I. & Umnov, N.(1969). Some problems of elaboration the walking machines. *Bulletin of Russian Academy of Sciences* 2.
- Bares, J., Hebert, M., Kande, T., Krotkov, E., Mitchell, T., Simmons, R. & Whittaker, W.(1989). Ambler and autonomous rover for planetary exploration. *The Institut of Electrical and Electronics Engineers* 22 (6).
- Baumann, E.(1976). *Elektrische Kraftmesstechnik*. VEB Verlag Technik Berlin.
- Bekker, M.(1969). *Introduction to Terrain-Vehicle Systems*. University of Michigan Press,.
- Berns, K.(2006). The walking machine catalogue.
- Berns, K., Kepplin, V. & Dillmann, R.(1998). *Terrain and obstacle detection for walking machines using a stereo-camera-head*, vol. 2. Aachen, Germany: IECON'98.
- Bessonov, V. & Umnov, N.(1973). The analysis of gaits in six-legged vehicles according to their static stability. In *1st CISM-IFTOMM Symp. on Theory & Practice of Robots & Manipulators*.
- Bray, A., Barbato, G. & Levi, R.(1990). *Theory and Practice of Force Measurement*. Academic Press Limited.
- Bretl, T., Lall, S., Latombe, J.-C. & Rock, S.(2004). Multi-step motion planning for free-climbing robots. In *Workshop on the Algorithmic Foundations on the Robotics (WAFR)*, pp. 1-16. Utrecht, Netherlands.
- Byrd, J. & DeVries, K.(1990). A six-legged telerobot for nuclear applications development. *Int. J. Robot. Res.* 9 (2), 43-52.
- Chebyshev, P.(1945). *Research heritage of Chebyshev, issue, Theory of Mechanisms*. M.-L., Publishing House Academy of Sciences USSR.
- Cheng, F. & Orin, D.(1990). Efficient algorithm for optimal force distribution -the compact-dual lp method.. *IEEE Trans. on Robotics and Automation* 6 (2), 178-187.
- De Santos, P., Galvwez, J., Estremera, J. & Garcia, E. (2003). Silo4: a true walking robot for the comparative study of walking machine techniques. *Robotics and Automation Magazine, IEEE* 10 (4), 23-32.
- Devjanin, E., Gurfinkel, V., Gurfinkel, E., Kartashev, V., Lensky, A., Shneider, A. & Shtilman, L.(1983). The sixlegged walking robot capable of terrain adaptation. *Mechanism and Machine* 18 (4), 257-260.
- Devjanin, E., Kartashev, V., Lensky, A. & Schneider, A. Y.(1982). "Investigation of robotics", chap. Force feedback control of legged vehicle, pp. 147-158. Moscow (in Russian): Press "Nauka".

- Devjanin, E., Lensky, A., Samsonov, V. & Shtil'man, L.(1973). Elaboration of a model of the walking vehicle and correspondin control system. In *Proc. of V IFAC symp. on Automation Control in Space*. Genujy, Italy.
- Efimov, V., Kudriasev, M. & Titov, A.(1982). *Investigation of Robotics Systems*, chap. A Physical Similar of Motion Walking Apparatus, pp. 86–91. Moscow, (in Russian): Publishing "Nauka.
- Emura, T. & Arakawa, A.(1991). Attitude control of a quadrupped robot during two legs supporting. In *Inter.Confer.on Advanced Robotics*. Pisa, Italy.
- Galvez, J., de Santos, P. G. & Pfeiffer, F.(2001). Intrinsic tactile sensing for the optimization of force distribution in a pipe crawling robot. *IEEE/ASME Trans. on Mechatronics* 6 (1).
- Gardner, J.(1992). Efficient computation of force distributions for walking machines on rough terrain. *Robotica* 10 (5), 42 -433.
- Golubev, Y., Kartashov, V. & Schneider, A.(1979). 2 *All-Union conference "Biomechanics Problems"*, chap. Force distribution in legs of legged robot, pp. 152–153. Riga, (in Russian): "Zinatne" Press.
- Golubev, Y. & Korianov, V.(2003). Motion design for six-legged robot overcoming the vertical column by means of friction forces. In *Int. Conf. on Climbing and Walking Robots, CLAWAR'03*, pp. 609-616. Catania, Italy.
- Gorinevsky, D., Formalsky, A. & Schneider, A.(1997). *Force Control of Robotics Systems*, p. 350. N.Y.: CRC Press.
- Gorinevsky, D., Lensky, A. & Schneider, A.(1986). Dependence of the natural frequencies of a force-moment sensor on the design parameters. *Vestnik of Moscow University* 41 (3), 10–16.
- Gorinevsky, D. & Shneider, A.(1987). Dynamics of small motion of a walking robot when there is feedback with respect to the support reactions. *Mechanics of Solids* 22 (6), 37–46.
- Gorinevsky, D. & Shneider, A.(1990). Force control in locomotion of legged vehicle over rigid and soft surfaces. *Int. J. Robot. Res.* 9 (2), 4–23.
- Green, A. & Zerna, W.(1968). *Theoretical Elasticity*, chap. 457. Oxford: Clarendon Press.
- Gurfinkel, V. & Fomin, S.(1973). Biomechanical principles of constructing artificial walking systems. In *Proc. of the First CISM-IFTOMM Sympos., ROMANSY'73*, pp. 28–39. Udine, Italy.
- Gurfinkel, V., Gurfinkel, E., Devjanin, E., Efremov, E., Zhicharev, D., Lensky, A., Schneider, A. & Shtilman, L.(1982). "Investigation of Robotics", chap. Six-legged walking model of vehicle with supervisory control, pp. 98-147. Moscow, (in Russian): Press "Nauka".
- Gurfinkel, V., Gurfinkel, E., Schneider, A., Devjanin, E., Lensky, A. & Shtilman, L.(1981). Walking robot with supervisory control. *Mechanism and Machine Theory* 16, 31–36.
- Halme, A., Hartikainen, K. & Kärkkäinen, K.(1994). Terrain adaptive motion and free gait of a six-legged walking machine. *Control Eng. Practice* 2 (2), 273–279.
- Hirose, S.(1984). A study of the design and control of a quadruped walking vehicle. *Intern. J. of Robotics Research* 3 (2), 113–133.
- Hirose, S., Nacabuko, A. & Toyama, R.(1991). Machine that can walk and climb on floors, walls, and ceilings. In *ICAR*. Pisa, Italy.
- Hirose, S. & Umetani, Y.(1980). The basic motion regulation system for a quadruped walking machine. In *ASME, Design Engineering Technical Conference*. Los Angeles, USA.

- Ignatyev, M., F.M.Kulakov & Mihaeilov, A.(1974). Algorithms for control of robot-manipulators. *Mechanics of Machines* 46, "Nauka", Moscow, (in Russian).
- Inagaki, K. & Kobayashi, H.(1993). A gait for quadruped walking machine. In *Inter. Conference on Intelligent Robots and Systems*, , vol. 1. Yokohama, Japan.
- Ishino, Y., Narusa, T., Sawano, T. & Honma, N.(1983). *icar*, chap. Walking robot for underwater construction. Tokyo.
- Jiang, W., Liu, A. & Howar, D.(2004). Optimisation of legged robot locomotion by control of foot force distribution. *Transactions of the Institute of Measurement and Control* 26 (4), 311-323.
- Kaneko, M., Mizuno, A. & Harada, K.(2002). Torque distribution for achieving a hugging walk. In *Proc. of IEEE/RSJ, Int. Conf. on Intelligent Robots and Systems*, pp. 2613- 2618. Lausanne, Switzerland.
- Kaneko, M., Tanie, K. & Than, M. M.(1988). A control algorithm for hexapod walking machine over soft ground. *IEEE J. of Robot. and Automat.* 4 (3), 294-302.
- Kemurdjian, A.(1995). Planet rover today. In *Proc. of V11 Intern. Conference on Advanced Robotics, ICAR '95*, , vol. 1, pp. 293-299. Spain.
- Kirschner, F. & Spenneberg, D.(2001). *CLAWAR*, chap. Omni-Directional walking in multi-pod robot based on feedback driven oscillators and local reflex mechanisms.
- Klein, C. & Briggs, R.(1980). Use of active compliance in the control of legged vehicles. *IEEE Trans. Sys. Man Cybernet SMC-10* (7), 393-400.
- Klein, C. & Kittivatcharapong, S.(1990). Optimal force distribution for the legs of a walking machine with friction cone. *IEEE Trans. on Robotics and Automation* 6 (1), 73-85.
- Klein, C. & Wahavisan, W.(1984). Use of a multiprocessor for control of a robotic system. *Int. J. Robot. Res.* 1 (2), 45-59.
- Kumar, V. & Waldron, K.(1988). Force distribution in closed kinematics chains. *IEEE J. of Robotics and Automation* 4 (6), 659-663.
- Kumar, V. & Waldron, K.(1990). Force distribution in walking vehicles. *ASME J. of Mechanical Design* 112 (Mach.), 90-99.
- Lehtinen, H.(1994). Force based motion control of a walking machine. PhD thesis, Technical Research Centre, Finland, ESPOO.
- Marhef, D. & Orin, D.(1998). Quadratic optimization of force distribution in walking machines. In *IEEE of IEEE Intern.Conf. on Robotic and Automation*, pp. 477-483.
- Marsh, P.(1985). *Robots*. London: Salamander Book Limited.
- McGhee, R.(1976). *Neural Control of Locomotion*, chap. Robot locomotion, pp. 237-264.
- McGhee, R.(1977). Control of legged locomotion systems. In *Proc. 18th Automatic Control Conf.*, pp. 205-215. San Francisco.
- McGhee, R. & Iswandi, G.(1979). Adaptive locomotion of multi-legged robot over rough terrain. *IEEE Trans. on System Man, and Cybernetics SMC-9* (4), 176-182.
- McGhee, R., Olson, K. & Briggs, R.(1980). *Proc. of 1980 Automotive Engineering Congress. Detroit, Michigan*, chap. Electronic Coordination of Joint Motion for Terrain-Adaptive Robot Vehicles, pp. 1-7. N.Y.: Pergamon Press.
- Miura, H., Shimoyama, I., Mitsuishi, M. & Kimura, H.(1985). *Dynamical walk of quadruped robot..* MIT Press.
- Mocci, U., Petermella, M. & Salinari, S.(1973). chap. Experiences with six-legged walking machines with fixed gait. Belgrade: Advances in External Control of Human Extremities.
- Morrison, R.(1968). *Cornell Aeronautical Lab/ ISTVS Off-Road Mobility Research Symp.*, chap. Iron Mule Train. Washington DC, USA.

- Mosher, R.(1968). Test and evaluation of a versatile walking truck. In *Proc. of Off-Read Mobility Research Symp. Int. Society for Terrain Vehicle Systems*, pp. 359-379. Washington DC, USA.
- Nagy, P., Desa, S. & Whittaker, W.(1992). Predicting force redistribution on walking robots for reliable locomotion: Modelling and experiments". In *ASME Mechanisms Conference, September*, pp. 13-16.
- Okhotsimski, D. & Platonov, A.(1973). Control algorithm of the walking climbing over obstacles. In *Proc. of the Third Intern. Joint Conference on Artificial Intelligence*. Stanford, California.
- Okhotsimski, D. & Platonov, A.(1976). Walker's motion control. In *Proc. of CISM-IFTOMM Symp. "ROMANSY-76"*. Warsaw, Poland.
- Okhotsimski, D., Platonov, A., Gerken-Gubanov, G., Kuznetsov, V., Devjanin, E., Lensky, A., Gurfinkel, E. & Schneider, A.(1978). Integration walking robot simulation and modeling. In *7th Congress IFAC,,*, vol. 2, pp. 917-924. Helsinki: Pergamon Press.
- Okhotsimsky, D. & Golubev, Y.(1984). *Motion mechanics and control of motion an automated walking vehicle*, p. 312. Moscow, (in Russian): Publishing House "Nauka".
- Okhotsimsky, D., Platonov, A., Kiril'chenko, A., Lapshin, V. & Tolstousova, V.(1992). Walking machines. *Advances in Mechanics. Keldysh Institute of Applied Mathematics, Moscow* 15 (1-2), 39-70.
- Orin, D. & Oh, S.(1981). Control of force distribution in robotic mechanisms containing close kinematics chains. *J. Dyn.Syst. Meas. Contr.* 102, 134-141.
- Ozguner, F., Tsai, S. & McGhee, R.(1984). An approach to the use of terrain-preview information in rough terrain locomotion by a hexapod walking machine. *Int. J. of Robot. Research* 3 (2), 134-146.
- Paternella, M. & Salinari, S.(1973). Simulation by digital computer of walking machine control system. In *Proc. of V IFAC Symp. on Automatic Control in Space*. Genoa, Italy.
- Pfeiffer, F., Eltze, J. & Weidermann, H.(1994). The tum-walking machine. In *ISRAM'94*, pp. 1-20. Wailea, Maui, Hawaii, USA.
- Pfeiffer, F., Eltze, J. & Weidermann, H.(1995). Six-legged technical walking considering biological principles. *Robotics and Automation* pp. 22-23.
- Raibert, M.(1986). *Legged Robots that Balance*, p. 227. Cambridge, Mass.: MIT Press.
- Roßman, T. & Pfeiffer, F.(1998). Control of pipe crawling robot. In *Proc. Euromech 375. Biology an Technology of Walking*, pp. 133-140. Munich, Germany.
- Rosheim, M.(1994). *Robot Evaluation.The Developmant of Anthrobotics*. John Wiley and Sons, Inc.
- Roston, G.(1994). Lunar rovers for scientific and entertainment missions. *Acta Astronautica* 25, 397-406.
- Rygg, L.(1893). Mechanical horse. patented no 491,927, feb. 114, 1893. *Tech. Rep.*
- Schmucker, U.(2002). *Schwerpunktprogramm "Autonomes Laufen". Abschlußbericht*, chap. Multisensorielle Verfahren zur Bewegungssteuerung sechsbeiniger Schreitroboter, p. 88. Magdeburg: IFF.
- Schmucker, U., Schneider, A. & Ihme, T.(1996). Hexagonal walking vehicle with force sensing capability. In *Proc. ISMCR '96*, pp. 354-359. Brussels.
- Schmucker, U., Schneider, A. & Ihme, T.(1997). Force control for legged robots. In *ICAR'97 - 8th Int. Conf. on Advanced Robotics. Workshop "New Approaches on Dynamic Walking and Climbing Machines"*, pp. 38-43. Monterey, USA.

- Schmucker, U., Schneider, A., Rusin, V. & Zavgorodniy, Y.(2002). Control signals generation for limbs and body of walking robots. In *Int. Conf. on Climbing and Walking Robots, CLAWAR'02*, pp. 495–500. Paris.
- Schneider, A., Gurfinkel, E., Kanaev, E. & Ostapchuk, V.(1974). *Report No. 5, General and Molecular Physics Series*, chap. A system for controlling the extremities of artificial walking apparatus., pp. 64–80. Moscow: Physico-Technical Institute Moscow.
- Schneider, A., Zeidis, I. & Zimmermann, K.(2004). Stability of a “manipulator-drill” system with force control and time delay. *Technische Mechanik* 24 (1), 51–60.
- Shigley, J.(1961). The mechanics of walking vehicles. In *Proc. 1st Int. Conf. on Mechanics of Soil–Vehicle Systems*. Turin.
- Shik, M. & Orlovsky, G.(1976). Neurophysiology of locomotor automatism. *Physiol. Revs.* 56 (3), 456–501.
- Simmons, R. & Krotkov, E.(1991). *IEEE Inter.Conf. of Robotics and Automation*, chap. An integrated walking sytem for the ambler planetary rover. Sacramento, CA.
- Song, S.-M. & Waldron, K.(1989). *Machines that walk: The adaptive suspension vehicle*, p. 314. The MIT Press.
- Steuer, J. & Pfeiffer, F.(1998). Autonomous control of a six-legged walking machine. In *Proc. Euromech 375 .Biology and Technology of Walking*, pp. 141–148. Munick, Germany.
- Sukhatme, G. & Bekey, G.(1995). An evaluation methodology for autonomous mobile robots for planetary exploration. In *First ECPD Inter. Conference on Advanced Robotics and Intelligent Automation*, pp. 558–563. Greece.
- Thring, M.(1983). *Robots and Telechirs*. New York: Ellis Horwood, Limited.
- Todd, D.(1985). *Walking Machines–An Introduction to Legged Robots*. Essex, England: Anchor Press.
- Tomovic, R. & R, B. M.(1966). A finite state approach to the synthesis of bioengineering control systems. *IEEE Trans. on Human Factors in Electronics* HFE-7, 65–69.
- Umetani, Y., Omichi, T., Ibe, T., Hirose, S., Sirozi, K. & Ishibahi, A.(1985). Four legged walking robot. *Robot* (47).
- Umnov, N.(1981). Theory and methods of constructing efficient drives of multi-legged walking machines. PhD thesis, Inst. Mech. Eng., AN USSR, Moscow, (in Russian).
- Vukobratovic, M., Borovac, B., Surla, D. & Stokic, D.(1990). *Biped Locomotion*. Heidelberg, Berlin, New York: Springer-Verlag.
- Waldron, K.(1986). Force and motion management in legged locomotion. *IEEE J. of Robot. Automat.* 2 (4), 214–220.
- Whitney, D.(1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man-Mach.Syst.* 10, 47–53.

Force Sensing for Multi-legged Walking Robots: Theory and Experiments – Part 2: Force Control of Legged Vehicles

A. Schneider, U. Schmucker
Fraunhofer Institute for Factory Operation and Automation
Germany,

1. Use of force information in legged vehicle control

1.1 Main approaches and principles of force control

Approaches to manipulator control using force information can be subdivided into two major groups. The first group uses *logic branching* of the control when the measured force satisfies certain conditions. The second group introduces continuous force feedback as an *explicit force control or active force feedback method*.

The basic approaches to force feedback control that are already used or can be applied to walking robot motion control are discussed in many studies (Raibert & Craig, 1981; Gorinevsky *et al.*, 1997; Gurfinkel *et al.*, 1982, 1984; Mason & Salisbury, 1985; De Schutter, 1986, De Schutter & Brussel, 1988) and papers. Whitney (Whitney, 1977, 1987) was a pioneer of force control.

Stiffness control (Raibert & Craig, 1981). The simplest method of stiffness control is linear force feedback of the form:

$$U = c(F - F_d), \quad (1.1)$$

where U is a voltage of drive, is an applied force, F is a reference force and c is a compliant force sensor. If the force sensor is stiff, this feedback is equivalent to high-gain position feedback. The damping naturally present in the system may be insufficient for such feedback, thus resulting in a highly oscillatory system. To increase the damping, a velocity feedback should be introduced in the system.

Active or artificial compliance. This method was developed in the late nineteen seventies and early eighties for use in robotic systems as well as for six-legged robots (Whitney, 1977; McGhee *et al.*, 1980; Klein & Briggs, 1980; Devjanin *et al.*, 1982; Salisbury & Craig, 1982; Gurfinkel *et al.*, 1984; De Schutter & Brussel, 1988). The simplest law of this method is the form:

$$x - x_d = c(F - F_d) \quad (1.2)$$

where x is a coordinate of end-effector, x_d is a reference coordinate, F is an applied force, F_d is a reference force and c is a desired compliance.

In (Klein & Briggs, 1980), it is applied to six-legged OSU hexapod force control in the law form:

$$\dot{z} = \dot{z}_p + g_1(z_p - z) + g_F(\dot{F}_p - \dot{F}), \quad (1.3)$$

where z and \dot{z} are the actual vertical leg position and velocity, z_p and \dot{z}_p are the desired vertical position and velocity and g_1, g_F are gains, respectively.

In (Golubev *et al.*, 1979; Devjanin *et al.*, 1982; Gorinevsky & Shneider, 1987), a somewhat different law is used so that an external force acting on the legs of the robot "Masha" will cause a displacement of the end-effector. Fig. 1.1 shows the interaction of the positional and force elements of the control system for a leg of the walking robot "Katharina".

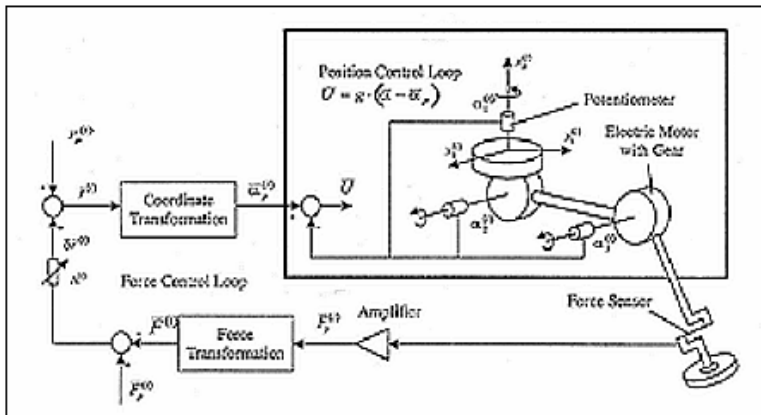


Fig. 1.1. Flow chart of leg force control.

In order to understand the behavior of a system with force feedback, assume that the servo systems track the commanded coordinates to the leg tips with high accuracy.

Write the radius vector $\bar{r}^{(i)}$ of the i -leg in the body's fixed coordinate system as $\bar{r}^{(i)} = \bar{r}_p^{(i)} + \delta\bar{r}^{(i)}$, where $\bar{r}_p^{(i)}$ is the commanded position calculated by the leg motion control algorithms and $\bar{r}_p^{(i)}$ is the force correction. The force feedback transmits the leg position to the positional servo system. The leg position differs from $\delta\bar{r}^{(i)}$ as:

$$\delta\bar{r}^{(i)} = \Lambda^{(i)}(\bar{N}^{(i)} - \bar{N}_p^{(i)}), \quad (1.4)$$

where $\Lambda^{(i)}$ is the symmetric positive definite feedback gain matrix and $\bar{N}^{(i)}$ and $\bar{N}_p^{(i)}$ are actual and commanded force vectors, respectively. If the force acting on the leg differs from the commanded value, it causes additional leg displacement proportional to the difference. Such system behavior is similar to that of an elastic spring with a compliance of $\Lambda^{(i)}$. Active compliance can be controlled by varying the elements of the matrix $\Lambda^{(i)}$.

The active compliance method is widely used to control the motion of walking robots' legs and bodies (Klein & Briggs, 1980; Devjanin *et al.*, 1982; Gorinevsky & Shneider, 1990; Alexandre *et al.*, 1998). For instance, the active compliance of a leg for the walking robots "Masha" and "Katharina" was in a range of 0.01cm/N to 0.03 cm/N.

Active accommodation or generalized damping control (Whitney, 1977; Mason & Salisbury, 1985; Schmucker *et al.*, 1997). The desired end-effector behavior is also often specified as damper behavior in the form

$$\dot{x} - v_d = g(F - F_d), \quad (1.5)$$

where \dot{x} is an end-effector velocity, v_d is a reference velocity, F_d is a reference force, and g is a damping factor. It follows that an externally applied constant force acting on the end-effector will generate a steady state motion with a *velocity proportional to the force*.

The theoretical analysis of this method and experiments has been treated many times over including in (Gorinevsky *et al.*, 1997).

The active accommodation method based on information on the main force and torque vectors acting on the vehicle body has been used for plane-parallel displacements and orientation changes of the body with resting support feet. Let the commanded vectors of linear and angular body velocities depend uniquely (for example, linearly) on the external force and torque as in

$$\bar{V} = G_f(\bar{F} - \bar{F}_p), \bar{\omega} = G_\omega(\bar{M} - \bar{M}_p), \quad (1.6)$$

where $\bar{V}, \bar{\omega}$ are the measured and commanded values of linear and angular body velocities, respectively, and G_f, G_ω are the matrices of accommodation. If a body's commanded vector velocities ($\bar{V}, \bar{\omega}$) depend on the external force or torque, then a vehicle body will be displaced or turned corresponding to the "accommodation".



(a) Downward-motion of body (b) Upward-motion of body (c) Left-right motion of body

Fig. 1.2. "Katharina".

Figures 1.2a - 1.2c show the plane-parallel displacements of the body of the robot "Katharina" due to an external force. The dynamometer imposed the force parallel to the Cartesian coordinates related to the body. The displacement of the body was a response to the external reaction.

Impedance control. The relation between the external force and manipulator motion can generally be specified as a desired *impedance* of the manipulator (Hogan, 1985; Kazerooni *et al.*, 1986; De Schutter & Brussel, 1988; Tzafestas *et al.*, 1995; Palis *et al.*, 2001).

The impedance can be defined as a transfer function between the external force acting on the manipulator and its displacement. The specified impedance can be achieved by different means using implicit or explicit force control.

Hybrid position/force control (Raibert & Craig, 1981; Salisbury & Craig, 1982; Sinha & Goldenberg, 1993). Some degrees of freedom of the end-effector are position controlled and others are force controlled. This method is based on the concept of a selection matrix, which is a diagonal 6x6 matrix with zeros and ones on the diagonal. **Artificial Neural Networks (ANN)** These networks include a large variety of control methods (Haykin, 1994). The

“Cerebellar modeled articulation controller” is one example of an ANN that has been used for the hybrid position/force control of a quadruped (Frik, 1996; Lin & Song, 1997; Cruse *et al.*, 1998).

Force control in theory and practice involves a number of different force control methods, surveys of which can be found, for example, e.g., in (De Schutter, 1986; Gorinevsky *et al.*, 1997; Sciavicco & Siciliano, 2000; Surdilovic & Vukobratovic, 2001).

1.2 Force control for step adaptation

Moving a vehicle over structured terrain requires adapting each leg to different ground clearance. The step cycle must be modified in order to obtain the correct ground contact. Foot force information is used to obtain the ground contact information.

While there is contact with the ground, the foot force increases as a function of the ground properties: quickly for solid ground, slowly for soft ground. The ground contact phase ends when the desired foot force distribution is attained. Together with active compliance, this produces an adaptable step. Analyzing the foot force dependent on foot position enables measuring information about soil softness. This is needed to adjust the step cycle in the transfer phase for sufficient foot clearance.

A similar algorithm is used for obstacle detection and navigation. During the transfer phase, the touch detection algorithm is activated in the direction of transfer. An obstacle is detected if the foot force reaches a predefined value. At this moment, the foot should be stopped (Devjanin *et al.*, 1983).

When obstacle detection is combined with active compliance, the foot begins to stop as the acting force increases and before the force level for obstacle detection is reached. In this case, a hard impact against an obstacle is prevented.

1.3 Use of the active compliance method for force component distribution of legs

The coordinate system $OX_1X_2X_3$ (Fig. 1.3) is used to describe leg motion.

Assume the voltages of the leg drives are

$$\bar{V}^{(i)} = G_i^{(i)}(\bar{V}_p^{(i)} - \bar{V}^{(i)}), \quad (1.7)$$

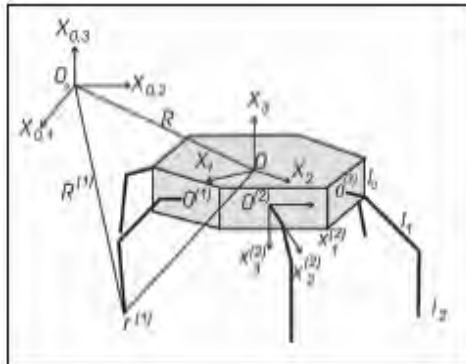


Fig. 1.3. Diagram of vehicle and coordinate systems.

where the program velocity vector is:

$$\bar{V}_p^{(i)} = -G_2^{(i)}(\bar{r}^{(i)} - \bar{r}_p^{(i)}) - G_3^{(i)}(\bar{N}^{(i)} - \bar{N}_p^{(i)})$$

Here, i is the leg number and $G_1^{(i)}, G_2^{(i)}, G_3^{(i)}$ are the (3x3) feedback gain matrices of velocity, position and force reaction, respectively. $V_p^{(i)}, \bar{V}^{(i)}$ are commanded and measured velocity vectors, $\bar{r}^{(i)}, \bar{r}_p^{(i)}$ i.e. radius vectors of realized and commanded position of i -th leg (vector $\bar{r}^{(i)}$ is determined by the joint angles), and $\bar{N}^{(i)} - \bar{N}_p^{(i)}$ are measured and commanded values of the force reaction in i -th leg. Expression (1.7) can be transformed into

$$\bar{U}^{(i)} = -G_r^{(i)} [(\bar{r}^{(i)} - (\bar{r}_p^{(i)} + \delta\bar{r}^{(i)}))] - G_1^{(i)}\bar{V}^{(i)} \quad (1.8)$$

where $\delta\bar{r}^{(i)}$ is equal (1.4), and $G_r^{(i)} = G_1^{(i)}G_2^{(i)}$. Vector $\delta\bar{r}^{(i)}$ can be considered a position correction of the commanded position of i -th leg tip, $\Lambda^{(i)} = (G_2^{(i)})^{-1}G_3^{(i)}$ is the matrix of the mechanical compliance of the leg. Value $\bar{r}_p^{(i)} = \bar{r}_p^{(i)} + \delta\bar{r}^{(i)}$ denotes a commanded position input to; the servo system.

It follows from (1.5) and (1.8), that when a measured force reaction $\bar{N}^{(i)}$ coincides with a commanded force reaction $\bar{N}_p^{(i)}$, the leg tip position $\bar{r}^{(i)}$ is equal to the commanded value $\bar{r}_p^{(i)}$. Since the leg joints of the experimental walking vehicles ("Masha" and "Katharina") are equipped with joint angle potentiometers, the commanded trajectories $\bar{r}_p^{(i)}$ notated as Cartesian coordinates have to be transformed into commanded joint angles $\bar{\alpha}^{(i)}$.

Both $(\bar{r}^{(i)} - \bar{r}_p^{(i)}) = J_r^{(i)}(\bar{\alpha}^{(i)} - \bar{\alpha}_*^{(i)})$ and the expression (1.8) can be written as:

$$\bar{U}^{(i)} = G_r^{(i)}J_r^{(i)}(\bar{\alpha}^{(i)} - \bar{\alpha}_*^{(i)}) - G^{(i)}\bar{\alpha}^{(i)}, \quad (1.9)$$

Where $J_r^{(i)}$ is the Jacobean from the joint angles to the Cartesian coordinates of the leg tip and $\bar{\alpha}^{(i)}$ is the three-component vector of measured joint angle values. In this case, the velocity feedback is not effective because the drives have high damping due to their high reduction. Consequently, in (1.9), $G^{(i)} = 0$ and the expression (1.9) for voltage can be written as

$$\bar{U}^{(i)} = G_2^{(i)}(\bar{\alpha}^{(i)} - \bar{\alpha}_*^{(i)}), \quad (1.10)$$

where $G_2^{(i)} = G_r^{(i)}J_r^{(i)}$ is the feedback gain matrix of the position.

The coordinate system $OX_1X_2X_3$, rigidly connected with the body of the robot and a world coordinate system $OX_{01}X_{02}X_{03}$ are used to evaluate algorithms for the control of foot reaction forces. The desired motion of the body can be described by a radius vector \bar{R}_p , describing the desired position of the body's center (point0), and a matrix $A_p^{(i)}$, consisting of the directional cosines between $OX_1X_2X_3$ and $OX_{01}X_{02}X_{03}$.

Assuming no slippage occurs between ground and feet, the programmed positions of the ends of supporting legs have to be specified according to

$$\bar{R}_p + A_p^{(i)} \cdot \bar{r}_p^{(i)} = \bar{R}^{(i)}, \quad (1.11)$$

where $\bar{R}^{(i)}$ is the radius vector of i -th foot contact point in the system $OX_{01}X_{02}X_{03}$.

Let the movement of the robot be controlled according to the formulas (1.5), (1.8) and (1.9), i.e. the feet have an artificial compliance. The robot's actual movement will then be

somewhat different than programmed. This means the radius vector $\bar{r}^{(i)}$ of the end points of the feet differs from $\bar{r}_p^{(i)}$ and (1.11) can be replaced by (1.8):

$$\bar{R} + A^{(i)} \cdot \bar{r}^{(i)} = \bar{R}^{(i)}, \quad (1.12)$$

where \bar{R} is the radius vector of the actual position of central point 0; $A^{(i)}$ is the actual matrix of directional cosines between $OX_1X_2X_3$ and $OX_{01}X_{02}X_{03}$.

If the deviation between programmed and actual paths is small, then, according to (1.11) and (1.12), the following equation holds for the supporting legs with an accuracy up to second order terms:

$$\Delta\bar{R} + \Delta\bar{\chi} \times \bar{r}_p^{(i)} + \Delta\bar{r}^{(i)} + \Delta\bar{r}_0^{(i)} + \delta\bar{r}^{(i)} = 0 \quad (1.13)$$

Here, $\Delta\bar{R} = A^{-1}(\bar{R} - \bar{R}_p)$ is the radius vector, characterizing a small linear deviation of point 0 from the programmed value, $\Delta\bar{\chi}$ is the vector of small angular deviation of body orientation, $\Delta\bar{r}^{(i)}$ is the radius vector of foot deflection due to elastic deformations, $\Delta\bar{r}_0^{(i)}$ is the radius vector of the error affected by the control system and $\delta\bar{r}^{(i)}$ is the correction value for the foot deflection added to the programmed value calculated from the force feedback according to (1.5).

Vectors $\Delta\bar{r}^{(i)}, \Delta\bar{r}_0^{(i)}, \delta\bar{r}^{(i)}$ are given in the coordinate system $OX_1X_2X_3$. Supporting forces $\bar{N}^{(i)}$ in the i -th leg and their elastic deformations are connected by $\Delta\bar{r}^{(i)} = (C^{(i)})^{-1}\bar{N}^{(i)}$, where $C^{(i)}$ is a positive definite matrix of the leg's mechanical stiffness.

The following assumes that the foot deflection $\delta\bar{r}^{(i)}$ resulting from artificial compliance is much larger than the deflections $\Delta\bar{r}^{(i)}, \Delta\bar{r}_0^{(i)}$. Neglecting their influence, the force distribution equation (1.13) can be written as:

$$\Delta\bar{R} + \Delta\bar{\chi} \times \bar{r}_p^{(i)} + \delta\bar{r}^{(i)} = 0 \quad (1.14)$$

Assuming the walking robot moves slowly, the influence of dynamic factors on the force distribution may be neglected too. The static equilibrium conditions are added to equation (1.5):

$$\sum_{i=1}^6 (\bar{r}_p^{(i)} + \delta\bar{r}^{(i)} \times \bar{N}^{(i)} = \bar{M}, \sum_{i=1}^6 \bar{N}^{(i)} = \bar{N} \quad (1.15)$$

where \bar{N} is gravitational force and \bar{M} is the general torque resulting from the gravitational force in a body's fixed coordinate frame.

Conditions (1.5), (1.14), (1.15) yield a closed system of equations to determine $\Delta\bar{R}$, $\Delta\bar{\chi}$, $\delta\bar{r}^{(i)}$, $\bar{N}^{(i)}$. The body's deviation from the nominal position (vectors $\Delta\bar{R}$, $\Delta\bar{\chi}$), the reactions $\bar{N}^{(i)}$ in the supporting legs and the vectors $\delta\bar{r}^{(i)}$ are defined by the parameters $\bar{r}_p^{(i)}, \bar{N}_p^{(i)}, \Lambda^{(i)}, \dots$. The values $\bar{r}_p^{(i)}$, which define the programmed movement of the leg ends, are calculated based on the position control element in the control system. The additional values $\delta\bar{r}^{(i)}$ according to (1.5) are calculated in the force control element of the entire system. Force feedback should occur with constant coefficients $\Lambda^{(i)}$. If the programmed movements of the legs (vectors $\bar{r}_p^{(i)}$) and the compliance matrix $\Lambda^{(i)}$ are given, the

programmed reaction forces $\bar{N}_p^{(i)}$, which can be provided as an input to the system, uniquely define the support forces $\bar{N}^{(i)}$ and the deviation of the body from the nominal position. According to (1.5), (1.14) and (1.15), the robot's actual movement coincides with the programmed movement ($\Delta\bar{x} = 0, \Delta\bar{y} = 0$) if the programmed reaction forces $\bar{N}_p^{(i)}$ meet the static equilibrium conditions. In this case, the actual reaction forces $\bar{N}^{(i)}$ are identical to the programmed values $\bar{N}_p^{(i)}$.

1.4 Distribution of vertical force components

First, we shall consider a situation in which the robot moves forward with a three point gait with a step length of about 10 cm over an even solid surface. There is no force control of distribution. The vertical force components of support reactions in the legs are plotted in Fig.1.4. Since the system is statically indeterminate with respect to the forces acting on the legs (more than three legs can be on a support) and the control system lacks force feedback, support reactions change randomly. This can lead to significant mechanical loading on separate legs.

The force distribution control essentially improves the robot's pattern of locomotion and increases its stability of motion.

Force distribution can be determined by several methods within the framework of static indeterminacy. The situations in which the walking robot moves relatively slowly have been considered and therefore the influence of dynamic factors on force distribution may be disregarded. Assuming the support surface is slightly uneven and the vehicle body is in the horizontal position, then the commanded horizontal force components are zero. The commanded vertical force components were computed from the body's orientation relative to the gravity vector and from the leg configuration.

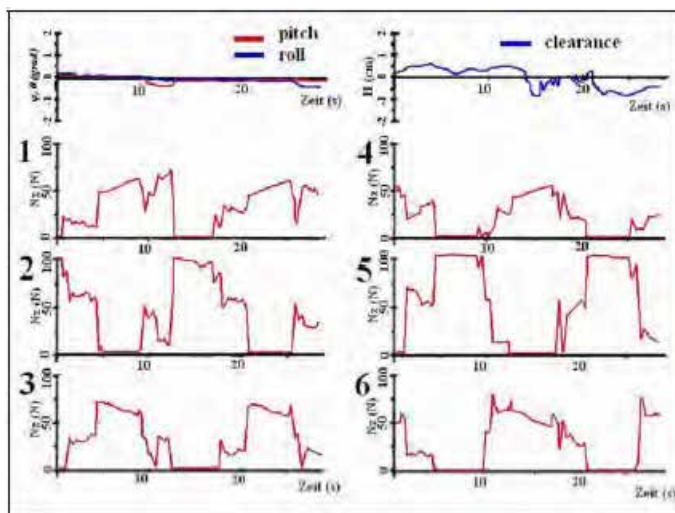


Fig. 1.4. Experimental results of foot forces in locomotion over solid surface.

Let the horizontal force components be zero. Then the vertical force components must satisfy the static equilibrium equations:

$$\sum_{i \in I} N_i^{(0)} = P, \sum_{i \in I} N_i^{(0)} x_i^{(0)} = PX, \sum_{i \in I} N_i^{(0)} y_i^{(0)} = PY, \quad (1.16)$$

where P is the vehicle weight, $x_i^{(0)}, y_i^{(0)}$ are coordinates of the i -th leg tip and X, Y are coordinates of the vehicle center of mass. Summation in equations (1.16) is performed over set of I the supporting legs.

Forces in n supporting legs should satisfy the three equations in (1.16). If $n > 3$, then there is more than one unique solution. There are different ways to eliminate the indeterminacy. Assume the vertical force components are required to satisfy

$$\sum_{i \in I} (N_i^{(0)})^2 \rightarrow \min \quad (1.17)$$

The purpose of this condition is energy optimization (Klein & Wahavisan, 1984). The exact condition of energy optimization is more complex and has been considered in (McGhee *et al.*, 1980; Okhotsimsky & Golubev, 1984).

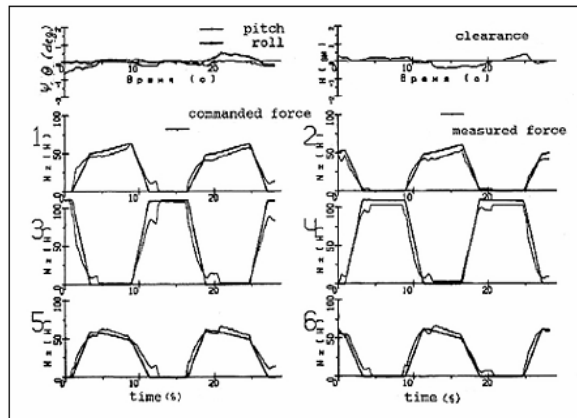


Fig. 1.5. Experimental results of vertical forces distribution.

Solving the equations (1.16) requires calculating the coordinates (X, Y) of the vehicle center of mass in terms of leg configuration. The simplified model of mass distribution contains the legs with the mass point m located on the leg ends and the body's mass $M_B = M - 6m$, where M is the total mass of the vehicle (Fig.1.6)

(Gorinevsky & Shneider, 1990). The solution of equation (1.16) under condition (1.17) can be obtained by LaGrange multiplier method (Gantmacher, 1960).

Experimental results have been obtained for the locomotion of the multi-legged robot "Masha" over an even, solid surface with tripod gait motion. The experimental results of the control of vertical force distribution in robot locomotion are plotted in Fig. 1.5. During the joint support phase, one set of legs is loaded and another is unloaded. Loads are redistributed smoothly, without jumps, as opposed to a situation in which forces are not controlled. More detailed results are presented in (Gorinevsky & Shneider, 1990).

1.5 Constrained motion control

The robot's body and the work tools attached to it can be used to perform manipulation tasks. In such cases, it is necessary to control the force reactions and position or velocity of tool motions along constraints. Required motions can be achieved, for instance, by adjusting the matrix elements of a force feedback control law depending on programmed or actual movements. Many service and process operations may be considered motions with a *mechanical constraint imposed* on the manipulated objects. Control algorithms for manipulator systems with motion along constraints are discussed in numerous studies (Mason & Salisbury, 1985; De Schutter & Brussel, 1988; Gorinevsky *et al.*, 1997; Lensky *et al.*, 1986). Similar situations arise out of the motion control of *legged platforms* as manipulation robots. Such operations should be performed by control of contact forces. Hence, it is necessary to have a *force system* which measures the vectors of main force (\vec{F}) and torque (\vec{M}) acting on the robot (see section 7.1).

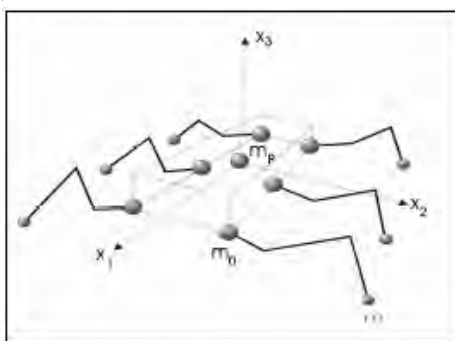


Fig. 1.6. Simplified model mass.

The generalized approach to synthesizing manipulator system motion along an object contour as a superposition of two basic motions at the normal and tangent to the contour is described (Gorinevsky *et al.*, 1997). A force sensor controls end-effector motion in the directions normal to the constrain.

A motion “along the constraint”, i.e. tangential to the constraint, is under positional control. The basic motion motion is maintaining contact with an object. Accordingly, the “desired” velocity vector \vec{V}_p of a tool or manufacturing equipment is as a sum of two terms:

$$\vec{V}_p = \vec{V}_n + \vec{V}_\tau = g(F_n - F_p)\vec{n} + V\vec{tau} \quad (1.18)$$

The first term \vec{V}_n in (1.18) is a vector directed towards the object if $F_n < F_p$ and from the object if $F_n > F_p$. The second component $\vec{V}_\tau = const > 0$ in the expression (1.18) is the programmed velocity tangential to the surface of the object (e.g. stick, tool). $F_p = const > 0$ is the programmed value of the normal component of the force with which, for example, a tool presses against the surface of an object, F_n is the normal force vector, component \vec{n} and \vec{tau} are the normal and tangential unit vectors, $\lambda > 0$ is a constant feedback gain and is a commanded value of the tool velocity tangential to the surface of the object.

If the friction between the tool and the surface is absent or there is Coulomb friction, the vectors \vec{n} and \vec{tau} can be determined from a force sensor.

For many service operations such as assembly, moving tools along a surface, opening hatches, rotating various handles, etc., it is expedient to use the law of (1.18). In other cases when there is no movement along the constraint, e.g. drilling operations, it is possible to assume $v_{\bar{r}} = 0$.

If the constraint is not known exactly, the motion of the body with the manipulator can be extended in all directions. In this case, the body's compliance motion can be implemented with active compliance or damping controls. Relationship (1.18) represents a nonlinear method of control. Although V_n is calculated using linear force feedback, the control law (1.18) also involves the product $V \cdot \bar{n}$.

These force control approaches have been used in problems of locomotion, to maneuver walking robots and for various service operations (Schmucker *et al.*, 1997; Schneider & Schmucker, 2000).

2. Locomotion over soft soil and soil consolidation

A ground deformation under the supporting leg leads to a modification of robot motion. Vertical sinking of supporting feet into a soft soil necessitates maintaining the correct position of the body (angular and vertical velocities) relative to the bearing surface during every step. A shear deformation of the ground leads to reduced vehicle velocity in absolute space (Bekker, 1969; Okhotimsky *et al.*, 1985; Wong, 1993; Gaurin, 1994). Thus, it is necessary to work out the method and algorithms for locomotion over soft soil and stabilization of the vehicle body.

2.1 Determination of mechanical properties of soil by means of legs

In an investigation of ground passability, the resistance to compressive and shear forces can be used to measure ground bearing capabilities. One of the most widely used ground bearing capability measurement devices is the bevameter (Bekker, 1969; Wong, 1993; Kemurdjian & *et. al.*, 1993; Manko, 1992).

A bevameter consists of a penetration plate and drive to implement various dependencies between the penetration plate's sinking and lateral shift and also to record the compressive load and shear. To obtain a more accurate picture of ground bearing while a vehicle is moving, the dimensions and form of the penetration plate should correspond to its support surface. The walking vehicle leg equipped with a foot force sensor and a joint angle potentiometer constitutes an ideal bevameter. The laboratory robot "Masha" was used to study soil properties (Gorinevsky & Shneider, 1990; Schneider & Schmucker, 2001).



Fig. 2.1. A view of experiment "load-sinkage".

The “load-sinkage” curves for different soils and some artificial materials were obtained experimentally. In the experiments, all legs but one stayed on the rigid support; the remaining leg was placed on the soil for analysis (Fig.2.1). The load on this leg was repeatedly changed from zero to a maximum value (about 100 N) and *vice versa*. Joint angle sensors determined leg sinkage and force sensors measured the load. The maximum load on the leg was 120 with a foot area of 30cm^2 , equal to a specific pressure of 40 kPa. Some of the experimental “load-sinkage” relations of different soil types are shown in Fig.2.2a,b. As Fig. 2.2b and the literature make clear, sinkage in natural soils is irreversible. “Load-sinkage” relations for artificial materials are virtually unique and linear.

2.2 Basic approaches to locomotion over soft soil

The easiest way for a robot to walk over soft soil is to fix its locomotion cycles. The inhomogeneity of the soil’s mechanical properties and the unevenness of the surface may disturb vehicle motion considerably.

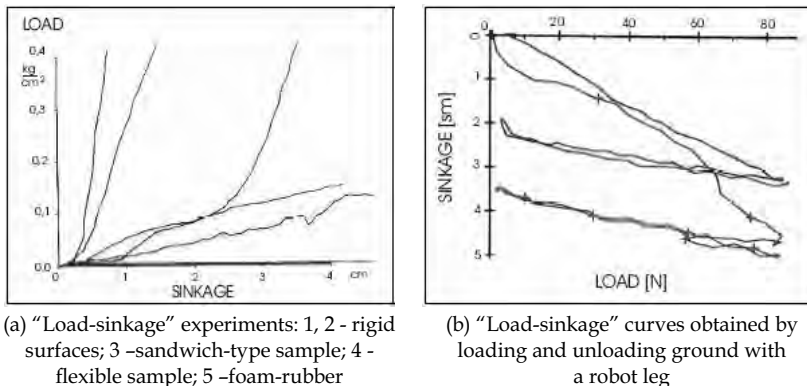


Fig. 2.2. “Load-sinkage” curves.

A more complicated approach is a locomotion cycle with an adaptation zone (Gurfinkel *et al.*, 1981). In order for a leg to strike the supporting surface in the adaptation zone, body position must be stabilized in relation to the support surface (roll, pitch and clearance), compensating for leg sinkage. To obtain smooth motion, the motion of each leg has to be corrected individually based on its sinkage.

Another demand on control algorithms is leg sinkage control in the joint support phase. If one of the support legs destroys the soil under it and sinks into the soil beyond the permissible level, the vehicle must be stopped and the sunken leg unloaded. A body’s movements during vehicle motion over a non-rigid surface need to be stabilized in relation to the supporting surface. A variety of *approaches* can accomplish this.

The *first approach* is to control body movement along the normal to the surface. To do this, the position of the body in relation to the supporting surface (pitch, roll, clearance) must be known. In this case, the supporting legs are controlled in the same way as on a rigid surface, i.e. the supporting polygon remains stiff.

However, most motion disturbance occurs within relatively short time periods when the set of supporting legs is changing. Therefore, to stabilize body movement, it is better to correct the programmed displacements of legs in relation to the body, corresponding to the setting of the legs into the soil. This is the *second approach* to stabilizing body movement during vehicle motion over soft soil.

2.3 Motion control algorithms

The sinkage of a leg into the soil depends on the load put on that leg. Accordingly, three methods to control leg sinkage suggest themselves.

The *first method* uses force feedback to control foot force reactions as shown in (Schneider, 1999). In this case, the stabilization of body movement requires allowing for a leg's sinkage into the soil when motion is generated. If the "load-sinkage" characteristics for each leg are known *a priori*, the sinkage can be computed from the programmed load on the leg. With this approach, the angular and the linear positions are corrected continuously during the motion.

The *second method* also assumes *a priori* knowledge of the "load-sinkage" characteristics for each leg. In this case, there is no force feedback and each leg's sinkage into the soil is controlled instead. To program sinkage, the corresponding force reactions must satisfy the static equilibrium equations.

The *third and most complicated approach* is to control bearing reactions and leg sinkage simultaneously. In this case, programmed leg motion is corrected for current leg sinkage into the soil. Although the mechanical properties of soil are not assumed to be known *a priori* in this case, the magnitude of sinkage into the soil must be known for each leg.

When soil is loaded, both reversible and irreversible deformations generally occur. To work out algorithms, let us assume that the relation of the sinkage to the applied load is the same for each point of the bearing area. Based on such an assumption, only the first two control methods, i.e. either control of bearing reactions or leg sinkage into the soil, have been worked out. Only two simplified types of soil shall be considered. For the *first type of soil*, all deformations are reversible and sinkage depends uniquely on load. Such an "elastic" surface might be found in a peat bog, for instance, where a layer of peat covers water. Although such situations are not widespread in nature, the problem of locomotion over elastic soil is of interest in and of itself.

The second type of soil has completely irreversible deformations. Most natural soils approximate this model. Such soil behaves as an absolutely rigid support if the load on the foot becomes less than a maximum value already achieved. The properties of naturally consolidating soils may differ considerably, even for adjacent points of terrain (Gorinevsky & Shneider, 1990). Thus, the algorithm for motion control on such surfaces is based on the third method, which does not assume the soil characteristics are known *a priori*.

2.3.1 Locomotion on linear elastic soil

This algorithm is based on the assumption that the soil properties are known *a priori*.

Let us assume that the force $N_i^{(i)}$ depends on the linear sinkage $s_i^{(i)}$ of the *i*-th leg as

$$N_i^{(i)} = C_s s_i^{(i)}, \quad (2.1)$$

where soil stiffness C_s is equal for all the legs and $N_i^{(i)}$ is the vertical foot force.

Then, the motion of each supporting leg is corrected for its sinkage computed from the programmed foot force $N_p^{(i)}$

$$S_p^{(i)} = N_p^{(i)} / C_s, \tag{2.2}$$

where the programmed foot-force $N_p^{(i)}$ is calculated using the force distribution algorithm of locomotion over a rigid surface.

2.3.2 Locomotion on consolidating ground

Let us assume that the mechanical properties of soil are not known *a priori*. Then the motion of each leg should be corrected for its sinkage. This cannot be computed beforehand. Rather, it can only be measured.

The algorithm is based on the assumption that soil deformation is *completely irreversible*. The leg may be considered to be on the rigid surface if the load on the leg is less than a maximum value already achieved. The absolute displacement of the body may then be determined from these leg positions.

Angular and linear displacements of the body must be known to determine leg sinkage into the soil. Let $\vec{r}^{(i)}(x_1^{(i)}, x_2^{(i)}, x_3^{(i)})$ be the radius vector of the *i*-th leg in the body's fixed coordinate system, $\vec{R}(x_{0,1}, x_{0,2}, x_{0,3})$ be the vector of displacements of the body's center in a world coordinate system and vector, $\vec{\varphi}(\psi, \theta)$ be small deviations of the roll and pitch of the body from its initial horizontal position and ΔH be a change in clearance resulting from leg sinkage.

Let $\vec{s}^{(i)}(s_1^{(i)}, s_2^{(i)}, s_3^{(i)})$ denote the foot displacement of the *i*-th leg in the body's-fixed, and in the world coordinate systems, respectively. Then the following ensues

$$\vec{S}^{(i)} = -\vec{\varphi} \times \vec{r}^{(i)} - \vec{R} + \vec{s}^{(i)} \tag{2.3}$$

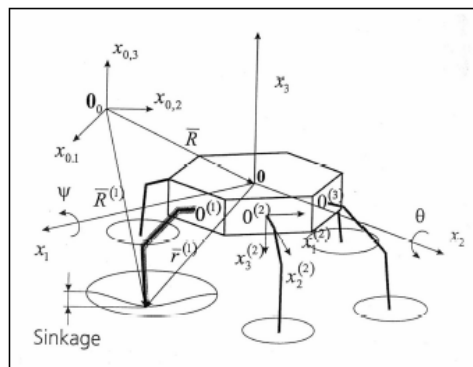


Fig. 2.3. Soil consolidation coordinate systems.

Let us assume that n ($n \geq 3$) legs are on absolutely rigid soil. For these legs, $\vec{\sigma}_{0,3}^{(i)}$ and equation (2.3) may be used to determine the angular displacements (ψ, θ) and linear displacements ΔH of the body. For a small vertical displacement $s_3^{(i)}$ of the leg after contact with soil, the foot displacement in the world

$$\sigma_3^{(i)} = \Delta H - \psi x^{(i)} + \theta y^{(i)} + s_3^{(i)} \quad (2.4)$$

As the load on the legs is redistributed, the sinkage of the legs standing on soft soil will increase. Using information from the position sensors, the sinkage of each of these legs can be determined with equation (2.4). The angular (ψ , θ) and linear (ΔH) displacements of the body can be determined by applying the LaGrange multiplier method.

2.4 Locomotion over consolidating ground with intermediate loading

The sinkage control algorithm for moving on a non-rigid irreversibly deforming bearing surface (see section 6.3) has a number of disadvantages. In the transport phase, when the sinkage of supporting legs is not controlled, the displacement of the vehicle's center of mass may cause the load on the legs to increase. This may cause the legs to impact the soil uncontrollably and even the vehicle body to "land" on the soil or stability to be lost if some of the legs sink too deeply into the soil.

To circumvent this drawback, a walking algorithm with intermediate soil loading was developed.

During the *joint support phase*, each leg is loaded intermediately to make contact with soil up to a maximum achievable load. In this case, leg sinkage may exceed the permissible value only in the joint support phase when the body position is controlled by the remaining legs standing on the consolidated soil. After such loading, the soil under these legs may be considered fully consolidated.

For the algorithm to function successfully, provision must be made for redistributing the vertical support reaction components among the supporting legs in such a way that the maximum load would be on a leg planted on the surface. Several solutions to the problems of vertical foot force redistribution are known, e.g. (Okhotsimsky & Golubev, 1984; Waldron, 1986; Kumar & Waldron, 1990) but the problem of load maximization for a given leg has obviously not been taken into account.

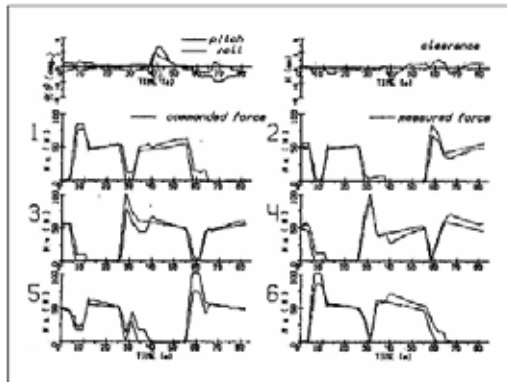


Fig. 2.4. Force distribution during locomotion on consolidating soft soil with intermediate loading (experiment with six-legged robot "Masha").

The distribution of vertical reactions by three supporting legs is unique and can be calculated, for instance, with (Gorinevsky & Shneider, 1990) if the number of supporting

legs is given. A newly planted leg is loaded as follows. The supporting triples corresponding to a maximum loading of each newly planted leg are determined at the beginning of the full support phase. The support phase is divided into equal time intervals, where is the number of newly planted legs. During the first intervals, the forces are smoothly redistributed to load each of the newly planted legs. In the last interval, the legs to be transferred are unloaded.

The algorithm for walking robot motion over a soft surface with soil consolidation has been tested (Fig. 2.4). The vehicle moved with a diagonal gallop gait. Since two legs are placed on the soil at a time, the full support phase is divided into two parts: consolidation of soil under each of the two legs and transition to the final state (i.e. unloading the legs to be transferred). It can be seen that the quality of tracking of the commanded forces and the stabilization of the body's motion has been improved.

3. Force control of body motion for service operations

3.1 Definition of the force-moment vector by means of support reactions

Synthesis of a control requires calculating the main vector of the force-moment arising from the process equipment's contact with an external object. The force-moment vector may be defined in two ways. A force sensor is placed between the robot body and the process equipment or the information from the force sensors in the robot legs is used. We shall consider the first method of calculating a force-moment vector.

Let the external force \vec{F} at the point O (Fig.3.1) be applied to the body of the legged robot standing on a rigid surface. The following orthogonal coordinate systems are introduced: the coordinate systems fixed to the surface ($O_0X_{01}X_{02}X_{03}$), fixed to the body ($OX_1X_2X_3$) and fixed to the attachment point of the legs ($O^{(i)}x_1^{(i)}x_2^{(i)}x_3^{(i)}$).

Assuming the displacement of the body due to flexible deformation of the force sensors is insignificant, the influence of dynamic factors can be disregarded. Therefore, the *quasi-static* equations are applied to calculate \vec{F} and \vec{M} . In this case, the condition for the robot's equilibrium is that all actual external forces and torques equal zero:

$$\sum_{i=1}^n \vec{N}^{(i)} + \vec{P} + \vec{F} = 0, \quad (3.1)$$

$$\sum_{i=1}^n [\vec{r}^{(i)} \times \vec{N}^{(i)}] + [\vec{r} \times \vec{F}] + \vec{M}_F = 0, \quad (3.2)$$

where $\vec{N}^{(i)}$ is the vector of support reactions measured with the aid of force sensors mounted in the legs, \vec{P} is the vehicle weight, \vec{F} is the external force vector acting on the body (or on the tool mounted on the body) and \vec{M} is the moment of external forces.

For the majority of operations performable by a manipulation robot or an adaptive legged platform, the geometric parameters of body and tools and the coordinate of the point of force application are assumed to be known. If the point of force application is known beforehand, the active external components of force and torque can be defined.

The equation (3.1) can be used to define three components F_x, F_y, F_z of force vector \vec{F} . Accordingly, the directional cosines of vector \vec{F} are

$$\alpha = \frac{F_x}{|\vec{F}|}, \beta = \frac{F_y}{|\vec{F}|}, \gamma = \frac{F_z}{|\vec{F}|}, \text{ where } |\vec{F}|^2 = F_x^2 + F_y^2 + F_z^2.$$

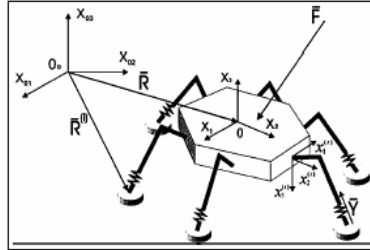


Fig. 3.1. Kinematics structure of the robot "Katharina".

If the external torque \bar{M} is equal to zero, then the three scalar equations obtained with torque equation (3.2) can be used to define the coordinates of a point $A_F(a_{FX}, a_{FY}, a_{FZ})$ of applied external force \bar{F} .

To determine the main force (\bar{F}) and the torque (\bar{M}) in the coordinate system connected with the center of the body, the force reaction from axes $P^{(i)}g_1^{(i)}g_2^{(i)}g_3^{(i)}$ and the linear displacement must be transformed to axes $OX_1X_2X_3$.

3.2 Assembly operation by body displacement: Tube-in-hole insertion

An assembly task is demonstrated here by inserting a tube with a diameter d_0 into a funnel-shaped hole of an external object, the position of which is unknown (Fig.3.2).

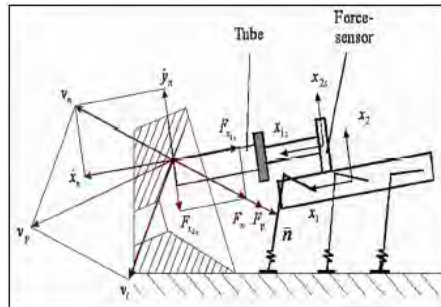


Fig. 3.2. Inserting a tube into a hole.

A force sensor rigidly connects the tube to the body of a hexapod vehicle. Its axial direction is parallel to the OX_1 axis, which is rigidly related to the body and its origin in the body's center. The external object's surface is a funnel-shaped hole with a diameter $d > d_0$. The task was studied using the linear motion of the robot's body and a method based on measuring the reaction force components generated by contact between the tube and the funnel-shaped surface.

The body of the robot is moved in such a way that the reaction forces are minimized. The tube moves toward the hole and touches the inside of the funnel. The force components are measured during this motion.

In this approach, force vector components are measured by establishing contact between the tube end and the funnel-shaped surface, maintaining a constant contact force equal to the programmed value and moving the robot's body with the tube in the direction of the hole.

Two modes of force control have been investigated: independent translational body motions and the superposition of body motions. The elements of the accommodation matrix are adjusted so that they are large for movement's perpendicular to the hole and small for movements along the tube axis.

3.2.1 An algorithm with accommodation of independent motions

Components of the body's velocity vector $\bar{V} = \text{col}(V_{xP}, V_{yP}, V_{zP})$ were calculated conforming to the expression $\bar{V}_p = G_F(\bar{F}_p - \bar{F}_c)$, where $G_F = \|g_{ij}\|$, \bar{F} is the measured value of contact force between the tube end and funnel-shaped surface and \bar{F}_c is the commanded force. This algorithm transforms the motion of the body into three independent translational motions. The experiments have demonstrated that a loss of contact can occur between the tube and the surface of the funnel-shaped hole. In the force reaction measurements, this phenomenon is observed as sudden changes of the force components, in particular changes of the longitudinal force component.

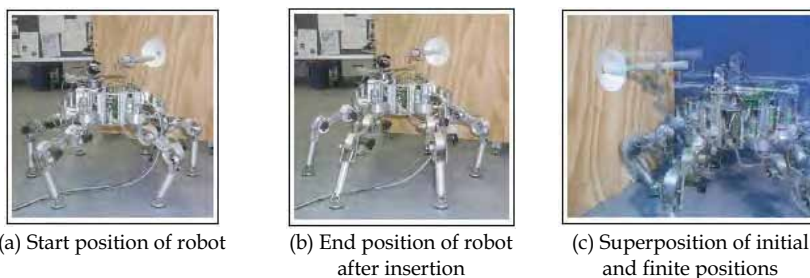


Fig. 3.3. Positions of the robot during the experiment.

3.2.2 An algorithm with accommodation and superposition of motions

This approach employed the control law as a *superposition of two basic motions*, i.e. motion normal and tangential to the funnel-shaped surface. A constraint was also added to the effect that the motion along a normal should maintain a constant force contact between the tube and the funnel surface. Accordingly, the programmed value of the tube velocity may be represented as the sum of the two components according to equation (1.18).

To utilize the control system described in section 4, the value of the force F_n as well as \bar{n} and $\bar{\tau}$ which describe the direction of the body with the tube displacements must be evaluated. To determine F_n , \bar{n} , and $\bar{\tau}$, let us assume the longitudinal axis of the tube and the hole are collinear and that the tube moves over the funnel shape without friction.

Under the assumptions made, values F_n and \bar{n} can be evaluated with the help of the force sensor as in (Lensky *et al.*, 1986; Gorinevsky *et al.*, 1997).

Coefficients V_r , F_p , λ , and F_{pX} have been selected experimentally: $V_r = 0,7 \text{ cm/s}$, $F_p = 30 \text{ N}$, $\lambda = 0,14 \text{ cm/sN}$, $F_{pX} = 20 \text{ N}$.

The results revealed that, in the case of the *second control law*, tube movement along the funnel-shaped hole was more even and smooth. Contact was not lost between the tube and the funnel-shaped surface. Fig.3.3a, b and c show positions of the robot during the experiment.

Obtained during the motion of the tube along the funnel-shaped hole force components $F - X, F_y, F_z$ are plotted in Fig.3.4. There, x, y, z are displacements of the body and tube

along axes OX , OY , OZ over time. Values of Δx , Δy , Δz were calculated relative to the stationary feet standing on the surface. Other important values are the diameter of the tube $d = 32mm$, the diameter of the hole $d_0 = 40mm$, the diameter of the funnel $D = 140mm$, and the height of the funnel $h = 50mm$.

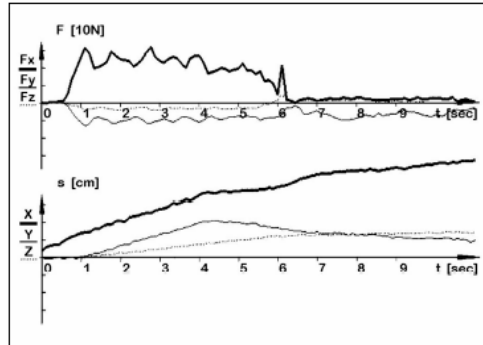


Fig. 3.4. Inserting a tube into a hole.

Some characteristic stages can be distinguished in the graph: The motion of the tube as it comes into contact with the funnel-shaped surface, the motion of the tube along this surface toward the hole and finally the insertion of the tube into the hole. After contact occurred between the tube's end and the surface, a force resulted and the program controlled the body's motion. Contact was established between the tube and the funnelshape with a normal force F_n and the tube was moved along the conical surface. As the tube began to move into the hole, the value of the longitudinal force component F_x decreased to zero. This was the signal to switch to accommodation control based on (1.18).

3.3 Implementation of body motion control for a rotating a handle

Let us consider a task of rotating a handle with a rod mounted with a force sensor on the body. Tasks similar to rotating a handle or steering a wheel by means of a manipulator were studied, e.g. in (De Schutter, 1986; Gorinevsky *et al.*, 1997).

In our experiments, the force sensor is attached to the end of the manipulator arm fixture on the body of the legged robot. The other end of the sensor is connected to the tube that comes in contact with the handle. In this task, only two body translation degrees of freedom are used to control the motion of the handle and the manipulator is used as a bearing structure to which the tube is attached.

The position of the tube tip inserted in the grip of the handle is restricted to the circumference. Rotating the handle requires moving the body (together with the tube) along the circumference. In doing so, each point of the mast should move along a curve closely approximating a circle. We assume that the position of the circle's center and radius are not known *a priori*.

Therefore, to avoid a loss of contact between the tube and grip of the handle, the commanded force F_p of contact with the object (see 1.18) must be positive. For the problem considered here, the constraint is binding (bilateral). Hence, using the same control expression (4.18) to solve this problem, we can take $F_p = 0$:

$$\bar{V}_p = gF_n\bar{n} + v\bar{\tau} \quad 3.3$$

In the preceding equation, $V_p = (V_{XP}, V_{YP})$ is a commanded velocity vector for the body together with the tube motion, \bar{n} and $\bar{\tau}$ are vectors of normal and tangent to the constraint (circle), F_n is the projection of the force applied to the sensor onto the normal \bar{n} , $v = const$ is the commanded velocity of the handle rotation (contouring velocity) and $g > 0$ is a constant gain. The direction of rotation is determined by the direction of the vector $\bar{\tau}$ and sign of v . The normal \bar{n} in (3.3) can be chosen as a vector directed either away from or toward the center of the handle.

If the friction in the handle axis and handle mass are negligible, then the force the handle exerts on the sensor is always in a direction tangent to the radius. Measuring the vector \bar{F} of this force determines the vector of normal $\bar{n} = \bar{F}/F$ and thus the vector of tangent $\bar{\tau}$. Then we can rewrite (3.3) as follows:

$$\bar{V}_p = g\bar{F} + v\bar{\tau} \quad 3.4$$

When the force of resistance to handle rotation is nonzero but not too large, (3.4) may also be used.

If the load resistance of handle rotation is large, then (3.3) should be used to calculate the accommodated velocity vector. *A priori* information about friction in the handle axis may be used to calculate the vectors of normal and tangent to the constraint.

The position sensors in the legs can also be used to determine these vectors. As the handle rotates, each point of the tube moves together with the body along a curve closely approximating a circle. By measuring consecutive positions of the tube, a secant to the curve can be determined to build vectors close to $\bar{\tau}$ and \bar{n} . The vectors of normal and tangent should be given at the start of the motion.

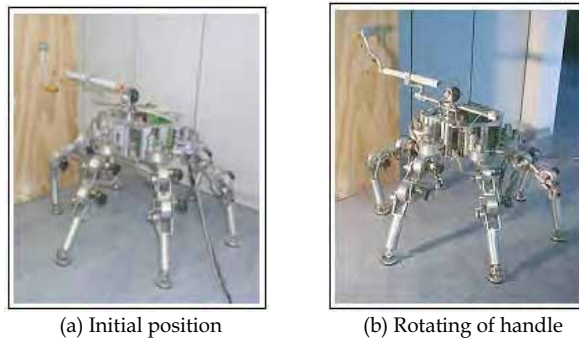


Fig. 3.5. The experimental scheme.

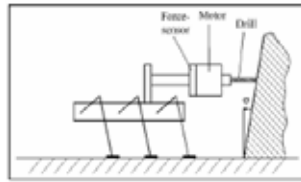
The control laws presented were successfully applied in the experiments. The experimental scheme is shown in Fig. 3.5a (start position) and Fig. 3.5 b (rotation of handle).

3.4 Drilling operation

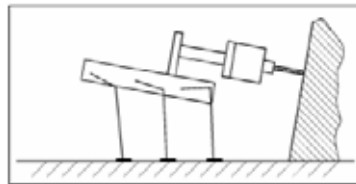
Another task was body motion for drilling operations. Using the body together with a drill as a working device requires controlling body movement in such a way that the longitudinal axis of the drill is collinear to the normal direction of a part's surface. Fig. 3.6a and 3.6b illustrate the initial and end positions of robot body orientation when drilling.

At first, the robot moves parallel to the OX axis to come in contact with the surface. The moment of contact is taken from the force sensor data. The body's displacement X_1 and clearance H_1 are evaluated at this moment. The change in the coordinates of the supporting leg ends describe the displacement.

Next, the body changes its clearance to H_2 through plane-parallel movements. Later, the body again moves in the axis OX . The body's displacement X_2 is evaluated at the moment of contact. The angle of the slope φ of the working surface in relation to the $O_1X_1Z_1$ plane (angle of relation around O_1Y_1) is provided by:



(a) Initial position of robot drilling



(b) End position of robot drilling

Fig. 3.6. Illustration of initial and end positions of robot body.

$$\beta = \arctan \frac{X_1 - X_2}{H_1 - H_2}, \quad 3.5$$

Next, the body changes its clearance to H_1 and moves parallel to the OX axis to come into contact with the working surface. The programmed velocities of body rotation around the O_1Y_1 axis are

$$\omega_{PY} = k_Y \cdot \sin \beta, V_{PX} = \omega_{PY} \cdot R, \quad 3.6$$



Fig. 3.7. Robot "Katharina" by drilling.

The inclination sensor measures the actual tangential deviation θ of the body during its movement. When $(\varphi - \theta) \leq \Delta$ is smaller than the given value Δ , the movement of the body is stopped. The tool then moves to touch the working surface. The direction of motion is parallel to the axis. The program controls the value of the force during contact. The program starts the drilling operation after initial contact.

In drilling operations, the pressing force determines the cutting force, which has to be controlled. The cutting force has to be constant in the direction of normal force F_N at all times. This force has to be close to F_{PX} . To prevent jams and tool breakage, the control system can compensate for the lateral components of interaction forces that arise during the drilling operation. Force control of the body for technological operations has been developed and experimentally tested. Control of a moving body can be solved in terms of the force and torque vectors acting on the drill. If it is assumed that the commanded vectors \vec{V}_P and $\vec{\omega}_P$ of the body depend on the force and torque expressed as:

$$V_{PX} = k_X(F_X - F_{PX}); \omega_{PY} = -k_Z F_Z; \omega_{PZ} = k_Y F_Y; V_{PY} = -\omega_{PZ} R; V_{PZ} = -\omega_{PY} R, \quad 3.7$$

where F_X, F_Y, F_Z force components are measured by the force sensor, k_X, k_Y, k_Z is the feedback gain and R is a constant describing the distance between the coordinate frame and drill end, then the vehicle body will move in a specified direction.

The accuracy of the preceding control algorithm cannot be expected to be very high because of the contact friction between the drill end and the working surface. To improve this situation a more effective algorithm was elaborated based on:

- Measuring several position points on the working surface,
- Evaluating the slope angle of this surface in relation to the fixed coordinate frame and the angles (pitch and course) of the vehicle body at the first point of contact between the drill and the working surface. The longitudinal displacement of a drill and the reaction to force components during drilling experiments are plotted in Fig. 3.8.

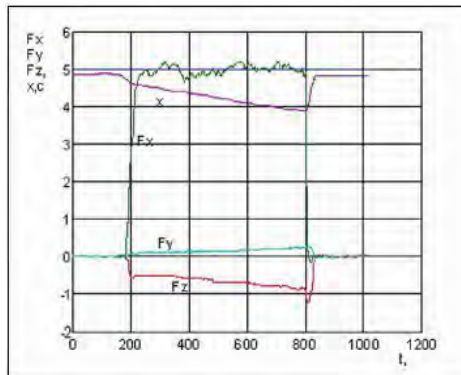


Fig. 3.8. Plots of the longitudinal displacement (x) and force components (F_x, F_y, F_z).

However, the experiments also demonstrated that sustained oscillations arise in the mechanical system whenever force feedback gains are large or force sensor stiffness is high and the feedback loop has a time delay. A simplified mathematical model of translation motion of the manipulator together with the drill maintaining contact with a rigid object

(wooden plate) is described in (Schneider *et al.*, 2004). This model was used to derive the stability criteria of the “drill-manipulator”, system for the legged robot.

4. Acknowledgements

Initial research began at the Institute of Mechanics together with the Institute for Problems of Information Transmission (Moscow) at the initiative of Professors D.E. Okhotsimsky, V.S. Gurfinkel, members of the Russian Academy of Sciences and Professor E.A. Devjanin. Research is continuing at the Fraunhofer Institute for Factory Operation and Automation (IFF, Magdeburg) and is being supported by the Germany Research Foundation DFG.

We would like to express special thanks to Professor D. M. Gorinevsky, together with whom the basic results of the force distribution algorithms have been achieved and also Professors A. Formalsky, and Yu. Golubev. The author is deeply grateful to Doctors A. Lensky, E. Gurfinkel and L. Shtilman.

5. Summary

Years of research and experimental development of sensing system and control algorithms allow drawing the following conclusions:

- The devices and control algorithms developed can be used as a basis for developing both new laboratory models and full-scale walking robots, their sensors, force systems and control algorithms;
- The implementation of impedance force control (e.g. active compliance, active accommodation, hybrid position/force control) simplifies the series of algorithms and increases the fields of application for walking robots;
- The information on foot force interactions between robot legs and a surface used by the control system improves adaptation to terrain roughness and provides for uniform distribution of forces between supporting legs;
- The control systems based on interaction force information are suited for solving the problem of moving a legged robot or displacing a body along a constraint (most applications require this type of control).

The information on the main force and the torque vectors acting on the vehicle body coming into contact with an external object and maintaining the specified contact force is used for assembly and drilling operations.

6. References

- Alexandre, P., Doroftei, I. & Preumont, A. (1998). An autonomous micro walking machine with articulated body. In *3rd IFAC Symp. on Intell. Autonom. Vehicles*, pp. 61–66. Madrid.
- Bekker, M. (1969). *Introduction to Terrain-Vehicle Systems*. University of Michigan Press.
- Cruse, H., Kindermann, T., Schumm, M., Dean, J. & Schmitz, J. (1998). Walknet—a biologically inspired network to control six-legged walking. *Neural Networks* 11, 1435–1447.
- De Schutter, J. (1986). Compliant robot motion: Task formulation and control. PhD thesis, Katholieke Univ. Leuven.
- De Schutter, J. & Brussel, H. V. (1988). Compliant robot motion. *Inter. J. Robot. Res.* 7 (4), 3–33.

- Devjanin, E., Gurfinkel, V., Gerfinkel, E., Kartashev, V., Lensky, A., Shneider, A. & Shtilman, L. (1983). The six-legged walking robot capable of terrain adaptation. *Mechanism and Machine* 18 (4), 257–260.
- Devjanin, E., Kartashev, V., Lensky, A. & Schneider, A. Y. (1982). "Investigation of robotics", chap. Force feedback control of legged vehicle, pp. 147–158. Moscow (in Russian): Press "Nauka".
- Frik, M. (1996). Adaptive neural control of a walking machine. In *7th German-Japanese Seminar on Nonlinear Problems in Dynamical Systems -Theory and Applications*, pp. 141–148. Reisenburg, Germany.
- Gantmacher, F. (1960). *The Theory of Matrices*. N.Y.: Chelsea Publishing Company.
- Gaurin, G. (1994). Control of walking robots on natural terrain. PhD thesis, ETH, Zurich, No.10898.
- Golubev, Y., Kartashev, V. & Schneider, A. (1979). 2 *All-Union conference "Biomechanics Problems"*, chap. Force distribution in legs of legged robot, pp. 152–153. Riga, (in Russian): "Zinatne" Press.
- Gorinevsky, D., Formalsky, A. & Schneider, A. (1997). *Force Control of Robotics Systems*, p. 350. N.Y.: CRC Press.
- Gorinevsky, D. & Shneider, A. (1987). Dynamics of small motion of a walking robot when there is feedback with respect to the support reactions. *Mechanics of Solids* 22 (6), 37–46.
- Gorinevsky, D. & Shneider, A. (1990). Force control in locomotion of legged vehicle over rigid and soft surfaces. *Int. J. Robot. Res.* 9 (2), 4–23.
- Gurfinkel, V., Devjanin, E., Lensky, A., Mozhevelov, S., Formalsky, A. & Schneider, A. (1984). Force feedback in manipulator control system. *Mechanics of Solids* 19 (6), 52–59.
- Gurfinkel, V., Gurfinkel, E., Devjanin, E., Efremov, E., Zhicharev, D., Lensky, A., Schneider, A. & Shtilman, L. (1982). "Investigation of Robotics", chap. Six-legged walking model of vehicle with supervisory control, pp. 98–147. Moscow, (in Russian): Press "Nauka".
- Gurfinkel, V., Gurfinkel, E., Schneider, A., Devjanin, E., Lensky, A. & Shtilman, L. (1981). Walking robot with supervisory control. *Mechanism and Machine Theory* 16, 31–36.
- Haykin, S. (1994). *Neural Networks*. Prentice Hall.
- Hogan, N. (1985). Impedance theory control - an approach to manipulation. *Trans. ASME J. of Dynam. Syst., Measur. and Control*, 107 (1), 1–24.
- Kazerooni, H., Hout, P. & Sheridan, T. (1986). Robust compliant motion for manipulators, part 1 and part 2. *Trans. of IEEE J. Robotics and Autom.* 2 (2), 83–105.
- Kemurdjian, A. & et.al. (1993). *Planet Rovers*, p. 319. Moscow, (in Russian): Mashinostroenie.
- Klein, C. & Briggs, R. (1980). Use of active compliance in the control of legged vehicles. *IEEE Trans. Sys. Man Cybernet SMC-10* (7), 393–400.
- Klein, C. & Wahavisan, W. (1984). Use of a multiprocessor for control of a robotic system. *Int. J. Robot. Res.* 1 (2), 45–59.
- Kumar, V. & Waldron, K. (1990). Force distribution in walking vehicles. *ASME J. of Mechanical Design* 112 (Mach.), 90–99.
- Lensky, A., Lizunov, A., Formalsky, A. & Schneider, A. (1986). Manipulator motion along a constraint. *Robotica* 4 (4), 247–253.

- Lin, Y. & Song, S.-M.(1997). Learning hybrid position/force control of a quadruped walking machine using a cmac neural network. *J. Robotic Systems* 14 (6), 483-499.
- Manko, D.(1992). *A General Model of Legged Locomotion on Natural Terrain*, p. 113. Kluwer Academic Publishers.
- Mason, M. & Salisbury, J.(1985). *Robot hands and the mechanics of manipulation..* Cambridge, Mass: MIT Press.
- McGhee, R., Olson, K. & Briggs, R. (1980). *Proc. of 1980 Automotive Engineering Congress. Detroit, Michigan*, chap. Electronic Coordination of Joint Motion for Terrain-Adaptive Robot Vehicles, pp. 1-7. N.Y.: Pergamon Press.
- Okhotsimsky, D. & Golubev, Y.(1984). *Motion mechanics and control of motion an automated walking vehicle*, p. 312. Moscow, (in Russian): Publishing House "Nauka".
- Palis, F., Rusin, V. & Schneider, A. (2001). Adaptive impedance/force control of legged robot systems. In *4th Int. Conf. on Climbing and Walking Robots, CLAWAR'01*, pp. 324-330. Karlsruhe.
- Raibert, M. & Craig, J. (1981). Hybrid position/force control of manipulators. *ASME J. of Dynam. Syst., Measur. and Control* 103 (2), 126-133.
- Salisbury, J. & Craig, J.(1982). Articulated hands: force control and kinematics issues. *Int. J. of Robot. Res.* 1 (1), 4-17.
- Schmucker, U., Schneider, A. & Ihme, T. (1997). Force control for legged robots. In *ICAR'97-8th Int. Conf. on Advanced Robotics. Workshop "New Approaches on Dynamic Walking and Climbing Machines"*, pp. 38-43. Monterey, USA.
- Schneider, A.(1999). Control algorithms of interaction between walking robot legs and soft soil. In *2nd Inter. Conf. on Climbing and Walking Robots, CLAWAR'99*, pp. 473-481. Pothsmouth, UK.
- Schneider, A. & Schmucker, U. (2000). Adaptive six-legged platform for mounting and service operation. In *3rd Int. Conf. on Climbing and Walking Robots, CLAWAR'00*, pp. 193-200. Madrid.
- Schneider, A. & Schmucker, U. (2001). Force legged platform "katharina" for service operations. In *4 Inter. Conf. on Climbing and Walking Robots, CLAWAR'01*, pp. 1029-1036. Karlsruhe, Germany.
- Schneider, A., Zeidis, I. & Zimmermann, K. (2004). Stability of a "manipulator-drill" system with force control and time delay. *Technische Mechanik* 24 (1), 51-60.
- Sciavicco, L. & Siciliano, B. (2000). *Modelling and Control of Robot Manipulators*, p. 378. Springer Verlag.
- Sinha, P. & Goldenberg, A. (1993). A unified theory for hybrid control of manipulator. In *IEEE Int. Conf. on Robotics and Automation*, pp. 343-348.
- Surdilovic, D. & Vukobratovic, M.(2001). *Mechanical Systems Design Handbook*, chap. Control of robotic systems in contact tasks, pp. 587-638. N.Y.: CRC Press.
- Tzafestas, C., Guihard, M. & M'Sirdi, K. (1995). Two-stage adaptive impedance control applied to a legged robot. In *Int. Conf. on Intelligent Robots and Systems*, pp. 173-178.
- Waldron, K. (1986). Force and motion management in legged locomotion. *IEEE J. of Robot. Automat.* 2 (4), 214-220.
- Whitney, D. (1977). Force feedback control of manipulator fine motions. *Trans. ASME J. of Dynam. Syst., Measur. and Control* 99 (2), 91-97.
- Wong, J. (1993). *Theory of Ground Vehicles*. N.Y.: John Wiley & Sons, Inc.

Force Sensors in Hexapod Locomotion*

Sathya Kaliyamoorthy¹, Sasha N. Zill² and Roger D. Quinn¹

¹Case Western Reserve University Cleveland,
OH, USA

<http://biorobots.case.edu>

²Marshall University Huntington,
WV, USA

<http://medix.marshall.edu/~zill>

1. Preface and Introduction

1.1 Preface[†]

The field of biorobotics has developed and flourished as a multidisciplinary interface between biologists and engineers (Ritzmann et al., 2000; Webb & Consi, 2001; Ayers et al., 2002). Collaborative work in the field initially arose from the recognition that studies of animal locomotion could provide insights in solving complex problems in the design and control of mobile machines (Beer et al., 1998). For example, some animals are remarkably successful at efficiently traversing irregular and non-horizontal terrains with great agility. Understanding the principles of leg and foot structure and the neural mechanisms of control of limb movements provides insights that can be emulated in the construction and regulation of legged robots to similar advantage. These insights have been successfully applied both to legged machines that resemble animals in their design and to robots that incorporate these principles more abstractly (Quinn et al., 2003).

These interactions have also been mutually beneficial in providing biologists with new methods to analyze living systems (Ritzmann et al., 2000; Webb & Consi, 2001). Computer simulations of walking have been completed for a variety of animals (cats, insects, humans) using control systems that reproduce elements of biological sense organs, muscles and neural pattern generators (Dürr et al., 2004; Ekeberg et al., 2004). In some studies, robots and computer control systems have been directly interfaced with animal nervous systems to analyze the properties of neural networks (Szücs et al., 2000).

The engineering method of Finite Element Analysis has also emerged as an important tool in analyzing biological structures. It has been applied to problems as diverse as understanding the biomechanics of extinct animal species (Alexander, 2006) and the effective design of artificial joints in humans (Ramos et al., 2006). It has also been utilized in neuroscience in analysis of biological sense organs to understand their mechanisms of transduction of mechanical forces (Hossl et al., 2006). This chapter describes studies we have

* Reprinted by permission of Sage Publications Ltd from *The International Journal of Robotics Research* (Special Issue on the 6th International Conference on Climbing and Walking Robots - CLAWAR 2003) 24(7) pp563-574 (Copyright Sage Publications 2005)

[†] Preface is original to this publication

performed using Finite Element Analysis to model the information provided by sense organs during locomotion. These studies have provided insights that could not be obtained using methods of contemporary neurophysiology due to technical limitations. In addition, they represent an approach that combines data from neurophysiological and behavioral studies with results from the application of engineering methods to provide insights in the encoding of forces that can be used in the design of legged vehicles.

1.2 Introduction

The control of forces that occur during locomotion is a problem that is common to animals and legged robots. In both systems, it is necessary to regulate forces to actively support weight, produce propulsion and counter unexpected perturbations due to loss of friction with the substrate. Force regulation becomes a formidable problem when traversing irregular or non-horizontal terrains. Adjustments to changes in the orientation of the gravitational vector must be integrated with mechanisms controlling body height and leg position. In a terrain with unknown geometric and material properties, a force control strategy is more appropriate than a position control strategy. Impedance control of each leg and utilization of positive load feedback (Prochazka, 1996) tremendously increase the performance of walking and climbing. Some animals, including insects, are able to make these adjustments based upon information provided by sense organs in the legs, without separate mechanisms (e.g. gyroscopes or inclinometers) to detect their orientation in the gravitational field. In recent years, biologists have increasingly appreciated that force or load sensing plays a vital role in insect locomotion (Duysens et al., 2000; Delcomyn et al., 1996). During locomotion, insects invariably sense not only the positions of their joints but they also indirectly measure the load in each of their legs via strain sensors, as they walk, climb, and negotiate obstacles. They also adjust muscle tensions and joint compliance as they change speed or load conditions (Bässler & Büschges, 1998).

In general, load sensors fulfil four major functions in animal systems (Prochazka, 1996). These sense organs can:

- a) Detect the level of force and its rate of change during locomotion – Sense organs that detect forces in animals are activated by mechanical deformation (Zill, 1990). In the legs, these receptors are found in muscle tendons or are embedded in the exoskeleton (in invertebrates). The coupling of the sensors to these structures limits their responses and allows them to specifically detect the magnitude and rate of change of forces acting upon the legs.
- b) Provide information about force direction during stance – Most force detecting sensors in biological systems are directionally sensitive (Zill, 1990). The maximal responses occur when forces or strains are in a specific orientation (along the muscle tendon or in a particular direction of compression of the exoskeleton). In walking, these forces are largest during stance, when the load of the body is placed upon the legs and forces are generated to support, propel and stabilize the animal.
- c) Detect sudden force decreases (as in slipping or instability) – Most biological sensors produce continuous signals (trains of action potentials) that reflect the magnitude of forces applied to the leg. Decreases in these signals can therefore be used as indicators of loss of friction or leg slipping.
- d) Enhance muscle activity through feedback during support or propulsion – The signals from biological sensors are processed in the central nervous system (Bässler

& Büschges, 1998). Many force sensors can elicit reflexes that produce rapid activation of leg muscles through these connections.

The studies described in this paper provide insights into mechanisms by which load sensors in insect legs can contribute to each of these functions. A better understanding of these mechanisms may lead to improvements in their applications in legged robots and potentially enhance the locomotory capabilities of these vehicles.

This paper describes how finite element analysis (FEA) was used to develop detailed structural models of cockroach trochanters, a small segment connecting the coxa and femur that incorporates four groups of strain receptors. FEA models of front and rear legs incorporating these trochanters were loaded with representative foot forces predicted from dynamic simulations of modelled cockroaches walking and climbing. Strains at the receptor locations were measured and compared for different loading conditions and conclusions were made regarding how it is possible for the cockroach to decode these data, determine its detailed loading condition, and use it to perform the four functions described above.

2. Insect strain sensors

Insects possess biological strain sensors, called campaniform sensilla, on their exoskeleton (Zill, 1990). These receptors have been extensively studied as a model system for understanding force detection. These sense organs are similar to strain gages and encode forces through compressive strains that are generated in the insect exoskeleton. A campaniform sensillum (Figure 1) consists of a sensory neuron located in the epidermis. The nerve cell has a dendrite that passes through a canal in the cuticle and attaches to a thin cuticular cap at the surface of the exoskeleton. The design effectively functions as a local stress concentrator that distorts the cap when strains are transmitted through the cuticle. The cap, however, is asymmetrical and several studies have shown that campaniform sensilla respond best to forces that act perpendicular to the cap's long axis. There are eleven groups of campaniform sensilla on the cockroach leg. Four groups are found on a small segment (the trochanter) and form the largest aggregation of receptors in the leg (Zill et al., 1999). The cuticular caps of each trochanteral group have a specific orientation with all individual receptors within a group arranged approximately in parallel.

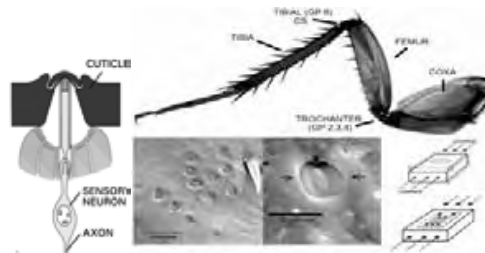


Fig. 1. Structure and locations of strain receptor groups in a cockroach leg; Each receptor consists of a sensory neuron whose process attaches to a cap embedded in the cuticle. The caps are oval shaped and organized in groups (as seen in the scanning electron micrographs in the center). Each receptor functions like a strain gauge and is excited by compressive strains in the exoskeleton that act perpendicular to the cap long axis (small arrows). Most strain receptors are found on the trochanter, a small segment in the leg (photograph of cockroach hindleg).

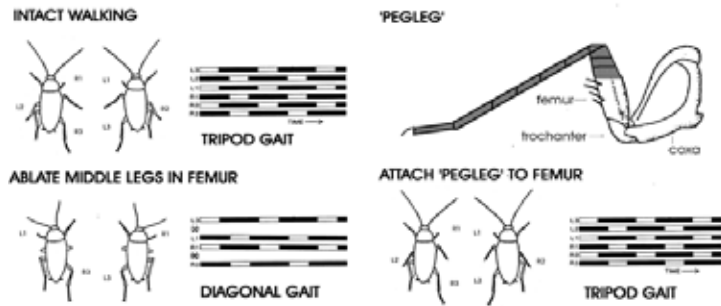


Figure 2: Peg leg experiments demonstrate the importance of sense organs of the trochanter in walking. The diagrams show the patterns of leg lifting in swing (dark bars) and placement in support during stance (light bars) as well as drawings of the legs in stance. Intact animals often walk in a pattern in which legs are lifted in groups of three, leaving a tripod of support. After ablation of the middle legs (leaving small stumps) the pattern of movement shifts to a diagonal gait, in which front and hindlegs of one side alternate. Small “pegleg” extensions were attached to the stumps, which still had intact trochanteral segments. Animals were able to use the prostheses in support and gaits returned to the pattern seen in intact animals. These experiments support the idea that signals from the proximal leg are necessary and sufficient to allow for limb use in support.

A number of studies have shown that the trochanteral groups can have decisive effects in the regulation of leg use (Wendler, G. 1966; Noah et al., 2004). The importance of the trochanteral campaniform sensilla in locomotion has been demonstrated by studies that have examined walking after partial leg ablation and addition of “pegleg” prostheses (Wendler, 1966). In intact animals, legs are often lifted in swing in pairs or groups that can produce “tripod” gaits. After ablation of the middle legs (leaving the trochanteral segments intact but unable to contact the substrate) the pattern of movement is altered and legs assumed a diagonal gait, similar to quadrupeds (Figure 2). However, addition of a prosthesis that permits contact and force development restores leg use. Noah et al. have extended these studies by recording from muscles in the “pegleg” and shown that normal patterns of activity occur if the leg can engage the substrate (Noah, 2004). However, when leg stumps were too short to reach the walking surface, abnormal multiple bursts occurred in single steps (Hughes, 1952; Delcomyn, 1985). These studies strongly suggest that signals from the trochanteral campaniform sensilla (and potentially other receptors of the proximal leg) are sufficient to allow for muscle activities to be sustained and for the leg to be used in support and propulsion in walking.

However, the activities of the trochanteral receptors have only been recorded in restrained or dissected preparations. Studies by a number of authors (Zill et al., 1999; Delcomyn, 1991) showed that the receptors could be strongly excited by forces applied to the leg when joint movements were resisted. Furthermore, these results suggested that the forces were encoded by different groups as an array rather than as sensors dedicated to particular force vectors. For example, two groups of sensilla (Groups 3 and 4) were shown to encode forces applied to the cockroach leg in the plane of joint movement of the adjacent coxo-trochanteral joint (Zill et al., 1999). The receptors within a group exhibited fixed patterns of

recruitment that could differentially indicate when force levels are adequate to provide support and propulsion during walking. Thus, these studies implied the sense organs could readily encode forces during locomotion. However, recordings of sensory activities could not be performed in freely moving animals due to technical constraints.

3. Modeling receptor excitation using finite element analysis

The anatomy and directional sensitivity of the receptors have provided the basis for modeling the responses of the trochanteral groups using Finite Element Analysis. The detailed structure of the exoskeleton was first reproduced by imaging the cuticle via confocal microscopy (Zill et al., 2000). This technique permits optical sectioning of solid materials as a series of digital images. The images were assembled and the cuticle was reconstructed in three dimensions using commercial software (Voxblast). Surface extraction algorithms were used to create files that mapped the inner and outer surfaces of the trochanteral segment. Those files were converted to a format compatible for stress analysis (Figure 3, meshed shell surfaces). These models preserved the values of thickness of the cuticle. It was also possible to precisely determine the locations and orientations of the individual groups of campaniform sensilla as nodes in the model. These FEA model trochanters were then incorporated into model "legs" of appropriate dimensions.

Forces were applied to the models at levels and orientations that occur during walking and climbing (Ramasubramanian et al., 1999; Kaliyamoorthy et al., 2001). Values for material properties of the cuticle were utilized from previously published studies. The strains at the specific locations and orientations of the cuticular caps were measured. Compressions that were greater than the threshold for receptor activation (as determined from physiological studies (Zill, 1999, *et. al*) were considered to produce excitation of the groups. These techniques have also been applied to forces that mimic the actions of the animal's leg muscles, which generate support and propulsion in posture and locomotion.

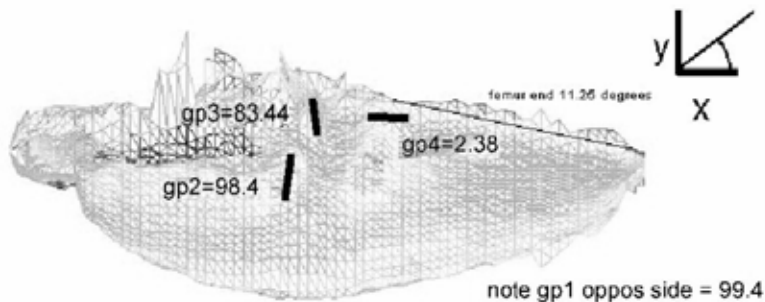


Fig. 3. Finite element model of the trochanter. A model of the trochanter of the cockroach front leg is shown. This model was derived from a three dimensional reconstruction of the exoskeleton obtained through confocal microscopy. The image shows the thickness of the exoskeleton as a grey scale code (thickest regions are dark, thinnest are light). Receptor groups are identified by their location and the numbers (and bars) indicate the angle of the orientations of the long axis of the caps of each group.

The error in the reduction of resolution of the data (polygon file) generated by the confocal microscopy software to a mesh compatible with FEA was estimated by a convergence method to be 3.25 percent (Flannigan, 1998). Also, there are local variations in the mechanical properties of cuticle surrounding the receptors (Chapman et al., 1973) that would tend to increase their sensitivity (cuticle is more compliant) so that calculations of responses probably represent underestimations. Lastly, while the force levels used in the model during walking and climbing are derived from simulations based upon inverse dynamics (calculated with assumptions, e.g. frictionless joints, rigid segment structure), these forces are in general agreement with data on ground reaction forces experimentally measured during cockroach walking (Full et al., 1991).

4. FEA of receptor responses during walking and climbing

A series of FEA experiments was conducted on the models of the trochanters of both the front and rear legs of cockroaches. Forces were applied to the models using data from dynamic simulations of cockroach walking (Nelson, 1995) and climbing (Choi, 2000) and the results were analysed to model the patterns of activation of the receptors.

4.1 Walking Studies

The results of the studies from simulations of walking for the front and hind legs are shown in Figures 4 and 5, respectively. The front and hind legs of cockroaches are similar in structure but differ in their size and use (Full et al., 1991; Kram & Full, 1997). Both legs have the same (homologous) groups of campaniform sensilla on the trochanter. However, they show different patterns of strain that would lead to excitation of the receptors during walking. In the front leg, analysis of strains using FEA (Figure 4) showed compressive strains (at levels that would result in activation) at the locations of specific groups of receptors (Group 4 and Group 1). The strains measured at these groups were high during both mid stance and late stance. Measurements at the locations of other groups showed only tensions that would not produce activity in the sensors. Thus, the different groups of sensors on the trochanter showed differential sensitivities to forces that occurred during walking.

A different pattern of activation was found in the rear leg (Figure 5). Compressive strains (at the orientation to produce sensor responses) were found predominantly in a single group (Group 3). Group 3 strains were high during the early phases of stance while showing reduced activation during late stance. In addition, much lower levels of excitatory strains were seen in Group 1. Group 4, which was strongly activated in the front leg, showed no periods of compression in the rear leg. These differences may be related to the orientations and uses of the front and hind legs that result in different forces. The plane of movement of the intrinsic leg joints is more vertically oriented in the front leg than in the hind leg. This orientation is suitable for the front leg to generate forces in the vertical plane (F_z) and to potentially provide support through a large part of the stance phase despite its relatively small size. In contrast, the rear leg is almost horizontal in orientation and is larger and more powerful. The rear leg generates much of the forward propulsive force in locomotion (F_x) (Full et al., 1991).

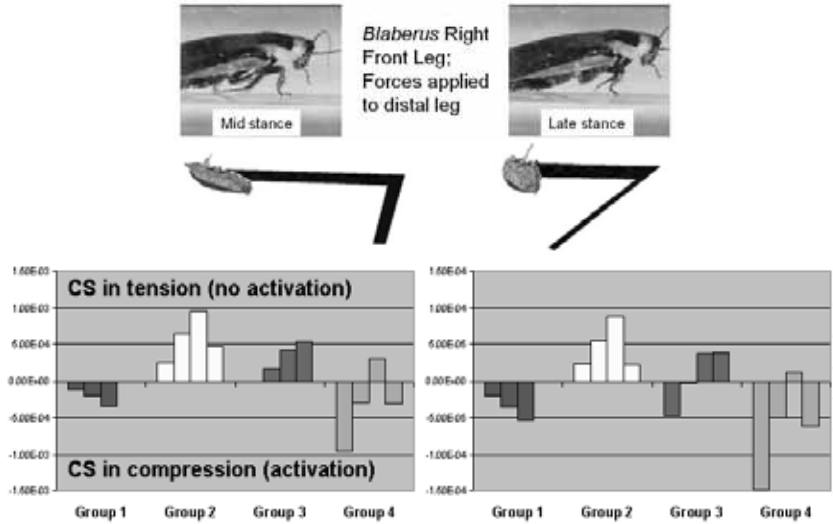


Fig. 4. Strains occurring in the cockroach front leg during walking - The model shown in Figure 3 was attached to structures that were of similar size and proportion to the other segments of the leg. Forces were applied to the distal end of the “leg” at levels and orientations determined from a simulation of cockroach walking. The lower graphs show the strains at the locations of the groups of trochanteral sensors (Groups 1–4) during mid and late stance. Compressions perpendicular to the cap long axis are negative and would produce activation of the sensors. Specific groups of sensors are activated during walking (see text for details).

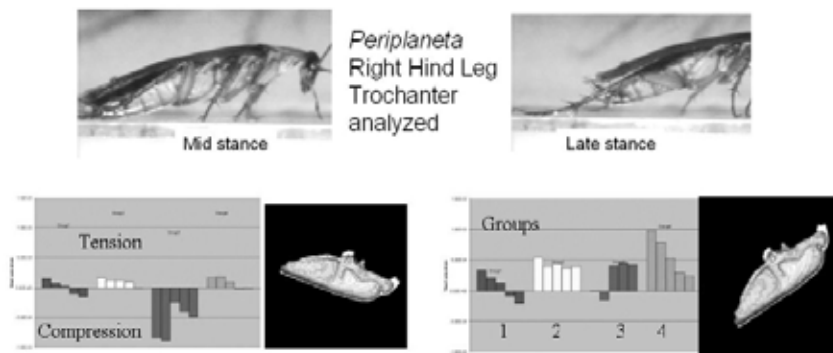


Fig. 5. Strains in the cockroach hind leg during walking (similar to Figure 4) - A model was also constructed from confocal images of the cockroach hindleg trochanter. Different groups of sensors show strain excitation in the hindleg than are activated in the front leg. Activation is maximal in mid stance, when the leg provides greatest support of body weight.

4.2 Climbing Studies

Studies of the patterns of strains occurring in the front and hind legs during climbing have provided insights into how the array of sensors can encode specific force vectors during adaptive locomotion. The front legs of the cockroach are the first to interact with obstacles during a climb (Watson et al., 2002). Some of the receptor groups in the front leg (for example Group 3), which did not show activation during walking, were excited during specific phases of climbing (Figure 6). These receptor groups could be activated by the additional forces needed to surmount an obstacle. This differential excitation prompted the question whether the receptors encode specific vectorial forces during walking and climbing. Figure 7 is a plot of the strains occurring at nodes of Groups 2 and 3 in the front leg trochanter vs. the magnitude of forces that are exerted in propulsion (F_x). There is a strong correlation in that both Groups 2 and 3 show excitation when the forces are positive (accelerating the animal forward) and are inhibited when the force direction is reversed (braking forward movement). Thus, the specific activation of the groups of trochanteral sensilla may be related to the particular forces occurring during climbing; the front legs which normally predominantly decelerate forward motion in walking may act to propel the animal over the obstacle during climbing. Other studies, in which individual force vectors were doubled or reduced to zero, also indicate correlation between receptor excitation and preferred direction of vectorial forces. Thus, the system may be able to detect the magnitude of specific force vectors based upon the preferential sensitivities of individual groups to particular force directions.

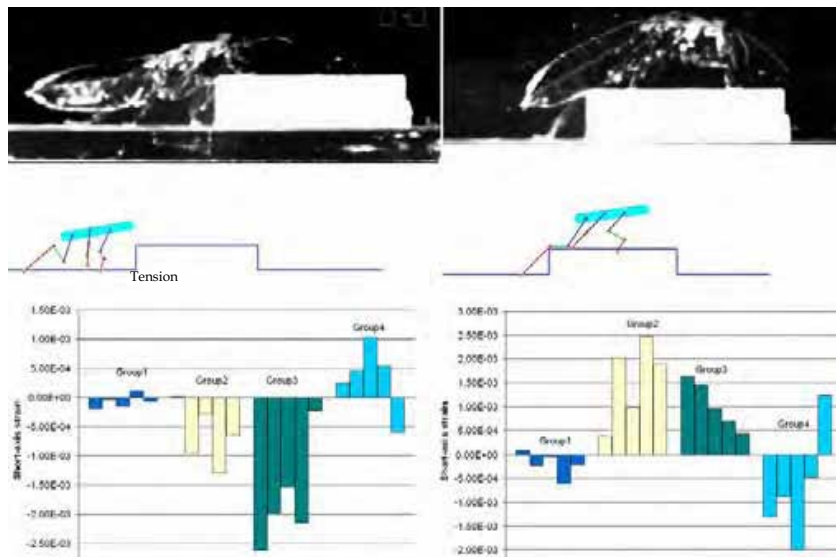


Fig. 6. Patterns of strain and group excitations occurring in the front leg during climbing – Forces were applied to the model at levels and orientations derived from a simulation of a cockroach ascending a small block. Different groups of sensors showed compressive strains during climbing, than had been found during walking. The pattern of excitation of groups was specific to the phase of climbing.

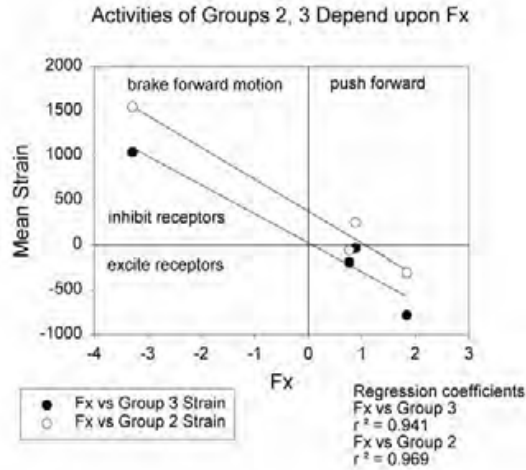


Fig. 7. The amplitude of strains is correlated with the magnitude of specific force vectors – Plot of the mean strain occurring in Groups 2 and 3 vs. amplitude of forces acting in propulsion. Excitation of the sensors during climbing is strongly correlated with forces that push the animals forward; the receptors are inhibited when forces act to brake forward motion.

5. Summary of findings

The results presented above support the idea that receptors on the exoskeleton can serve as specific load indicators during walking and climbing. Table I is a qualitative summary of the activation patterns of the groups of trochanteral sensors on the front and hind legs, as determined in the present tests and in a number of previous studies (Ramasubramanian et al., 1999; Kaliyamoorthy et al., 2001). Individual groups are scored as being strongly (indicated by X) or weakly (x) activated according to the magnitude of the compressive strains that were measured at the locations in the FEA model. “(-x)” denotes mixed response of sensors (some receptors activated within a group, others inhibited). The “pegleg” tests were performed by applying forces to the leg in the femoral segment (adjacent to the trochanter, see Figure 1) similar to the situation occurring in the classic “pegleg” experiments. Forces were applied to the model in directions that would produce flexion or extension of the coxo-trochanteral joint. The lower half of the figure summarizes results obtained in studies using simulations of walking and climbing. In addition, in the pegleg and walking studies, we also tested responses to forces applied at the points of attachment of leg muscles. Other tests were performed in which forces in individual directions were doubled, to test the response of sensors to specific force vectors in the body reference frame.

These studies support the idea that load sensors (campaniform sensilla) can fulfil the following functions:

- a) Detect the level of force and its rate of change during locomotion: The FEA tests produced compressive strains at the locations of most groups of campaniform sensilla, indicating that the sensors can readily encode forces that result from the effects of body weight and inertia. These results are consistent with the biological literature on campaniform sensilla. Zill and colleagues (Zill & Moran, 1981; Ridgel et al., 2000) have recorded the activities of the tibial campaniform sensilla in the cockroach hindleg. They found that one subgroup, the proximal tibial sensilla, showed a discharge during walking similar to that found by FEA analysis in the present study. The proximal sensilla fired after the onset of leg contact and the discharge declined later in the stance phase. These receptors have also been shown to be highly sensitive to the rate of force application (Ridgel et al., 2000). Similar patterns of activity have also been recorded from sense organs that monitor strains in the exoskeleton in crabs: funnel canal organs of the dactyl show discharges that also reach maximum rates immediately following leg contact and continue at lower firing rates later in stance (Akay et al., 2001). Thus, the physiological data from a number of invertebrates suggest that the receptors that encode forces are active during walking and that activities are maximal during the stance phase.
- b) Provide information about force direction during stance: Both the FEA tests in which forces were applied to the femur (pegleg tests) and the results obtained from simulations show that forces are encoded by the sensors as an array. It is important to note that sense organs in biological systems show adaptation to sustained levels of force and hysteresis to force decrements (Ridgel et al., 2000). If individual groups were dedicated to specific force vectors and only single measurements were made, difficulties might arise in accurately detecting dynamic increases and decreases in force levels. One advantage that the array of sensors may offer is that it could allow for finer discrimination by making multiple measurements of the changes in forces with sense organs of different sensitivities and thus, different degrees of adaptation and hysteresis. However, individual groups show preferential sensitivities to particular directions (F_x , F_z) that could be utilized to organize compensatory responses to sudden changes in load. Furthermore, the responses of groups in the front and hind legs reflect the similarities and differences in the forces they generate and undergo in locomotion. Both legs support the body weight during walking and climbing. Group 1 in the front and rear legs shows a high sensitivity to F_z (the gravitational vector). Information from this group could be used to assure that the support of body load is maintained in locomotion over diverse terrains. Information from the Group 1 sensilla in different legs could also be used in coordinating leg movements. However, the front and rear legs differ significantly in their contribution to forward propulsion. This is reflected in the differences in sensitivities seen in the front and rear legs to the effects of contractions of leg muscles and to F_x . As shown in Table 1, strains produced by the extensor muscle or by doubling of forces in propulsion (F_x) produce strong excitation of group 3. The same stimuli produce only modest effects in groups in the front leg.
- c) Detect sudden force decreases (as in slipping or instability): The Finite Element tests also indirectly support the conclusion that the receptors could readily function to detect leg slipping. The magnitudes of the strains that occurred at the locations of the sensilla were closely related to the amplitudes of the forces that were applied to the leg. Thus, the receptor discharges should decrease rapidly if

forces in the leg suddenly decline and cease if contact with the substrate is not maintained. The indication of decreasing forces may be important in detecting the onset of leg slipping to allow for adjustments of muscle activities or leg position (Ridgel et al., 2000). These functions could be fulfilled by strain or force sensors both in biological and potentially in robotic systems.

- d) Enhance muscle activity through feedback during support or propulsion: Physiological studies have shown the trochanteral receptors in the cockroach can strongly excite a leg extensor muscle that is a principal generator of leg movement and force (the trochanteral extensor, Ds) (Duysens et al., 2000; Zill, 1990). Thus, these results suggest that the sensors can provide feedback that could be used to enhance and sustain the activities of muscles to support body load and generate propulsion (Akay et al., 2001; Schmitz, 1993). Receptors in the crab also have been shown to produce reflex effects in leg muscles (Libersat et al., 1987). These sense organs are postulated to adjust activities of leg muscles to load and to aid in the coordination of leg movements. Thus, the physiological data from a number of invertebrates suggest that the receptors that encode forces are active during walking and can rapidly activate leg muscles at specific times during the stance phase.

Many groups of receptors also showed compressive strains both from external loads and from muscles that are active in propulsion. This may offer the advantage of rapidly monitoring both changes in load and the effectiveness of muscle contractions in countering those loads. This type of encoding is also seen in receptors of limbs of vertebrates (Golgi tendon organs) (Prochazka, 1996).

Table 1. Qualitative summary of the FEA experiments.

		Front Leg				Hind Leg			
		Group 1	Group 2	Group 3	Group 4	Group 1	Group 2	Group 3	Group 4
Peg Leg	Force Flex		x	X	(x)		x	X	x
	Force Ext	X			X				x
	Extensor muscle	x	x					X	X
Simulate Walking	Flexor muscle	x	x			X		X	
	Walking Load	X			X	x		X	
	Walking Extensor Muscle			X		X		X	
Simulate Climbing	Climbing elevate		x	X				X	
	Climbing rise			(x)	X			X	
Force Double	Fx				(-x)			X	
	Fy			X				x	x
	Fz	X				x			

Table 1. This table qualitatively summarizes the patterns of activation of different groups of sensors on the front and hindlegs in a number of tests, including the simulations of walking and climbing. In pegleg tests, forces were applied to the leg with joint movement resisted. (X - Strong activation; x - Moderate; (x)/(-x) - Mixed response). See text for discussion.

6. Principles for application in animal modeling and robotics

We have identified four load sensing and control principles based on the findings of these FEA studies that have particular applications to modelling of animal locomotion and control of legged vehicles. These are:

- 1) Positive Load Feedback - In the hindleg, the data in Table 1 clearly demonstrated that Group 3 sensilla show high levels of strain both to loads imposed upon the leg that mimic resistance to extension and to contractions of the extensor muscle that is active in generating propulsion. These elements could readily provide positive feedback in a walking animal and function to enhance the magnitude of muscle contractions and increase stiffness after the onset of stance (Zill et al., 2004). Positive feedback has been used in generating and regulating leg movements in a number of recent simulations (Ekeberg et al., 2004) and controllers (Kindermann, 2002; Dürri et al., 2004) that are based upon insect walking, and have also been utilized in simulations of vertebrate locomotion. In kinematically based models, signals of joint angular velocity are used as feedback signals (Dürri et al., 2004). However, direct use of force feedback may be particularly advantageous when traversing non-horizontal terrains, such as in walking on slopes or climbing or in carrying large loads. Positive feedback loops have often been viewed as undesirable as they are prone to be unstable. In some systems, muscle properties (length-tension relationships) that inherently limit force outputs may serve to limit instabilities (Prochazka, 1996). Studies of FEA models of the trochanteral campaniform sensilla suggest that leg geometry may provide an inherent limitation to feedback from sensors in the exoskeleton, as the mechanical advantage of the extensor declines as the joint angle extends (Nelson, 2002). This could readily produce high levels of positive feedback at the start of stance, but limit force generation as stance progresses. A similar pulse increase in feedback has been tested in control of Robot III, a cockroach inspired robot (Nelson, 2002).
- 2) Selective detection of force: support vs. propulsion - Individual groups of sensilla in the array are preferentially sensitive to force vectors in particular directions. This finding implies that the sense organs could readily provide specific signals to differentiate between loads that result from body weight versus loading that occurs to increased resistance in propulsion (as in specific phases of climbing (Watson et al., 2002). This would allow for selective use of specific force feedback in support vs. in propulsion. Support of body load has been regulated by negative feedback based upon positional signals (Dürri et al., 2004), but recent experiments suggest that information about both force and kinematic variables are integrated in the animal in generating support of body weight in standing and walking (Noah, 2004) in insects; (Pratt, 1995) in vertebrates). The problem of integration of kinematic and force data has been an area of intense research both in neurobiology and robotics (Zill et al., 2004; Ekeberg et al., 2004). These findings also represent additions that could be incorporated into behavior based control models (Kindermann, 2002). The model of Cruse, for example, is largely based upon control of kinematic variables, with few direct effects of load (e.g. Rule 5, (Dürri et al., 2004)). The addition of sensors that directly detect forces to these control systems would allow rapid adjustments to variations in load without potentially disruptive changes in joint angles. These sensors could also allow for use of both kinematic and force variables in controlling leg movements, as is shown by biological systems (Zill et al., 2004; Duysens et al., 2000). For example, in

animals, the onset of swing movements in walking depends both upon the position reached in extension (posterior movement) and upon unloading of the leg. The addition of sensors that preferentially detect a decrease in body load upon the leg (as occurs in the Group 1 trochanteral campaniform sensilla) could allow for incorporation of information from both variables into the control system.

- 3) Monitoring Forces Close to Body – A key observation on the organization of cockroach force detectors that may be of interest in design applications to roboticists, is that *monitoring forces at the hip or knee permits flexible use of the foot*. In the design of force sensors, sense organs that were equivalent to a load cell placed at the end of the leg might seem optimal. However, the structure of the insect foot (tarsus) provides a number of surfaces that can interact with the substrate (Larsen et al., 1997). The claws can actively grasp it and the arolium (a large sticky pad) can be actively pushed onto a surface by muscle contraction (Gorb & Beutel, 2001). Furthermore, on some surfaces, insects only use the pulvilli, which are sticky pads similar to the arolium but that are found under the tarsal segments (1–5). These pads maximize the surface area of contact of the tarsus. These features imply that the flexibility of the system prevents there being a simple point at the foot at which force could accurately be detected in all behaviours. For machines to gain similar flexibility of contact with the substrate, it may also be advantageous to utilize force sensors located in a leg some distance from the point of substrate contact. It could also be noted that the location of the sensors in the trochanter also allows for minimal temporal delays in conduction to the nervous system. This arrangement minimizes the calculations needed when force vectors are determined in the body coordinate system, in contrast to sensors placed at the foot.
- 4) Advantage in Redundancy: Increased Safety Factor – One further characteristic in the organization of strain sensors in insects is that they occur in groups and provide multiple measurements of force parameters. Insects apparently utilize this redundancy of measurements to produce signals from a number of sensors when forces are high (range fractionation, (Zill et al., 1999)). This is in contrast to the approach widely used in robotics, which is to have single sensors of high sensitivity. That approach has the disadvantage of being subject to potential loss of measurements if sensor failure occurs and is also subject to noise in the system. The use of multiple measurements endows the insect system with a considerable safety factor and may decrease noise in the signals of force parameters.

7. Implementation in legged robots

The four principles described above can be used to enhance the locomotion performance of legged robots. In fact, force sensors have been incorporated into a number of legged robots. As an example, Robot II sensed load at each of its joints and its control system used this information to enable its legs to comply with irregular terrain. TUM and Robot II reacted to load sensed in their legs to perform an elevator reflex to swing their legs over obstacles (Weidemann & Pfeiffer, 1995; Espenschied et al., 1996). The elevator reflex in these robots was triggered only during the swing (return) phase of the leg. Both of these robots used only position information to switch their legs to and from stance and swing. They could have benefited from using load sensing to make this switch as in the second principle described above. The switch from stance to swing should depend upon load to avoid suddenly swinging a leg that is supporting a significant portion of the body weight.

Robot III, a cockroach inspired robot, makes use of the third principle listed above, in having load sensors close to the body. It has strain gauges (Fig. 8) at locations equivalent to those where strain sensors are found in the cockroach (Nelson & Quinn, 1999). Strain gauges were positioned on the proximal end of the leg segment corresponding to the femur (Fig. 8, trochanteral sensors), similar to the location of the trochanter and trochanteral campaniform sensilla in cockroaches. The gauges were also oriented to detect bending strains in the plane of joint movement, similar to the cockroach trochanteral Group 3 receptors. Other strain gauges were located on the proximal tibia, at location corresponding to the well studied tibial group of receptors (Ridgel et al., 2000). Stance forces applied on its feet causes bending strains at these locations from which three-dimensional foot force vectors can be determined.

Several problems were encountered when implementing strain gauges on Robot III, which are typical in load cell design, but largely avoided in the animal's design because of the fourth principle described above (redundancy in measurements). Foil gauges were chosen over semiconductor gauges to avoid temperature sensitivity. They were placed in bridges to eliminate strain from loading other than from bending. The strain signals are weak and require amplification. Further, the signals are noisy and have to be filtered. Another problem is that the gauges are easily damaged and the loss of one sensor is a catastrophic failure of the entire load cell. Finally, they must be carefully calibrated. The fourth principle above suggests a different strain sensor implementation that would alleviate some of these problems. Instead of using a few carefully placed and well calibrated sensors, many sensors could be mounted in distinct groups on the leg. Noisy signals from individual sensors could be tolerated because the summation of their signals would filter the noise. Furthermore, the system could function well even if a number of sensors were damaged. This implementation requires micro strain gauges be manufactured in quantity on a membrane so that they could be mounted on the robot's legs. Unfortunately, we are unaware of the existence of such sensors.

Although experiments continue with strain gauges used in designs that closely follow the campaniform sensilla, we also have demonstrated that a pressure transducer located down-line from a pneumatic actuator produces a sufficient signal to determine actuator force (Kingsley et al., 2003). We found that strain gauges properly placed on the mounting elements of the actuators can produce sufficient signals for force feedback. However, pressure transducers can provide smoother signals, do not require amplifiers, and do not exhibit cross talk; all disadvantages of strain gauges (Kingsley et al., 2003). Force feedback can be attained through pressure measurements from the actuators, which, given actuator length, can be used to determine actuator force. However, we plan to utilize information from these sensors in accordance with general principles delineated by the FEA studies.

Force signals have also been implemented in controller design to resolve problems in regulation of leg movements. These systems incorporate positive load feedback and differential sensitivity to force vectors demonstrated in the FEA analysis. Positive load feedback, the first principle described above, has been shown to provide rapid load compensation by augmenting both the actuator's passive damping and the control system's active stiffness (Nelson, 2002). The positive load feedback loop used in the controller of Robot III is depicted in Figure 9 (see (Nelson, 2002) for details).

Furthermore, some authors have argued that measurements of these outputs are needed for adaptive locomotion. There is more to walking than gait coordination. The original stick insect mechanisms and network reviewed by Cruse (Cruse, 1990) simply decides when to transition legs between stance and swing phases. Nelson (Nelson, 2002) writes that "The algorithm tells you nothing about what to do *during* those phases to stabilize body posture

and motion.” Indeed, gait coordination is a small part of the controllers for Robot II and Robot III. We anticipate that further work on force control based on these FEA results will provide the information needed to elegantly control legs during their stance phases and to therefore robustly support and propel the body dynamically over obstacles.

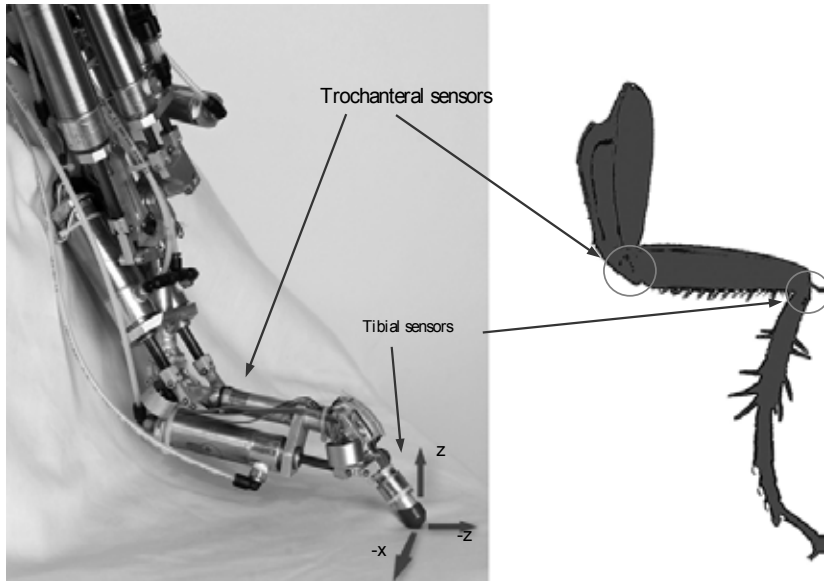


Fig. 8. Equivalent load sensor locations on the right front leg of Robot III. Note that the foot has been removed from Robot III.

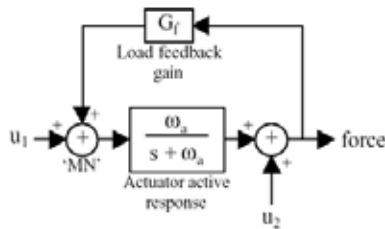


Fig. 9. Positive Feedback from Load Sensors.

8. Conclusions

Finite element analysis (FEA) was used to develop a detailed structural model of the cockroach trochanter, a small segment connecting the coxa and femur that incorporates four groups of strain receptors. The detailed structure of the exoskeleton was first reproduced by imaging the cuticle via confocal microscopy. The images were assembled and the cuticle

was reconstructed in three dimensions. Surface extraction algorithms were used to create files that mapped the inner and outer surfaces of the trochanteral segment, which were then converted to a format compatible for stress analysis. Nodes were identified in the FEA model at the locations of the strain receptors in the trochanter. FEA model trochanters for front and rear cockroach legs were then incorporated into simplified FEA model legs of appropriate dimensions so that they could be loaded with representative foot forces predicted from dynamic simulations of modelled cockroaches walking and climbing. Strains at the receptor locations were measured and compared for different loading conditions. The results presented in this paper (in light of previous work) indicate that the strain receptors in the exoskeleton can serve as specific load indicators during walking and climbing. Specifically, the load sensors (campaniform sensilla) can detect the level of force and its rate of change during locomotion, provide information about force direction during stance, detect slipping or instability, and enhance muscle activity through positive and/or negative feedback during support or propulsion. These functions are also desirable for load sensors for legged robots. Therefore, the load sensing mechanisms and associated control circuits found in animals can provide good models for implementation into legged robots. This work was supported by Office of Naval Research URISP Grant N00014-96-1-0694 and NSF Grant IBN-0235997.

9. References

- Akay, T., Bässler, U., Gerharz P., and Büschges A. 2001. The role of sensory signals from the insect coxa-trochanteral joint in controlling motor activity of the femur-tibia joint. *Journal of Neurophysiology* 85:594-604.
- Alexander, R.M. (2006) Dinosaur biomechanics. *Proc Biol Sci.* 7:273(1596):1849-55.
- Ayers J., Davis J. L., Rudolph A. (2002) Neurotechnology for Biomimetic Robots. MIT Press
- Bässler, U., and Büschges, A. 1998. Pattern generation for stick insect walking movements – multisensory control of a locomotor program *Brain Research Reviews* 27, 65–88.
- Beer, R.D., Chiel, H.J., Quinn, R.D., and Ritzmann, R.E. (1998), Biorobotic Approaches to the Study of Motor Systems. *Current Opinion in Neurobiology*, Vol. 8, pp. 777-782.
- Chapman, K.M., Duckrow, R.B. and Moran, D.T. 1973. Form and role of deformation in excitation of an insect mechanoreceptor. *Nature (Lond.)* 244: 453-454.
- Choi, J. 2000. *Dynamic simulation of cockroach climbing*, M.S.Thesis, Case Western Reserve University.
- Cruse, H. 1990. What mechanisms coordinate leg movement in walking arthropods? *Trends in Neurosciences* Vol.13, 15-21.
- Delcomyn, F. 1985. Walking and running, In: Kerkut, GA, Gilbert L. I., (eds), *Comprehensive insect physiology, biochemistry and pharmacology*, 5, chap. 11, Pergamon Press, London, 439-466.
- Delcomyn, F. 1991. Activity and directional sensitivity of leg campaniform sensilla in a stick insect. *Journal of Comparative Physiology A* 168, 113-119.
- Delcomyn, F., Nelson, M.E., and Cocatre-Zilgien, J.H. 1996. Sense organs of insect legs and the selection of sensors for agile walking robots, *The International Journal of Robotics Research*, 15 [2] 113-127.
- Dürr, V., Schmitz, J., and Cruse, H. 2004. Behaviour-based modeling of hexapod locomotion: linking biology and technical applications. *Arthropod Structure and Development* 33:237-250 (Special Issue: Arthropod Locomotion Systems: from Biological Materials and Systems to Robotics, RE Ritzmann, SN Gorb and RD Quinn, eds).
- Duysens, J., Clarac, F., and Cruse, H. 2000. Load-regulating mechanisms in gait and posture: comparative aspects. *Physiological Reviews* 80:83-133.

- Ekeberg, O., Blümel, M., and Büschges, A. 2004. Dynamic simulation of insect walking. *Arthropod Structure and Development* 33:287-300 (Special Issue: Arthropod Locomotion Systems: from Biological Materials and Systems to Robotics, RE Ritzmann, SN Gorb and RD Quinn, eds.)
- Espenschied, K. S., Quinn, R. D., Chiel, H. J., and Beer, R. D. 1996. Biologically-Based Distributed Control and Local Reflexes Improve Rough Terrain Locomotion in a Hexapod Robot. *Robotics and Autonomous Systems*, Vol. 18, 59-64.
- Flannigan, W.C. 1998. Finite element modeling of arthropod exoskeleton Master's thesis, Case Western Reserve University.
- Full, R.J., Blickhan, R., and Ting, L.H. 1991. Leg design in hexapedal runners. *Journal of Experimental Biology* 158:369-90.
- Gorb, S. N., and Beutel, R. G. 2001. Evolution of locomotory attachment pads of hexapods. *Naturwissenschaften* 530-534.
- Hossli B., Bohm, H.J., Rammerstorfer, F.G., Mullan, R., Barth, F.G. (2006) Studying the deformation of arachnid slit sensilla by a fracture mechanical approach. *J Biomech.* 39(10):1761-1768.
- Hughes, G. M. 1952. The coordination of insect movements. I. The walking movements of insects, *Journal of Experimental Biology* 156 215-231.
- Kaliyamoorthy, S., Zill, S.N., Quinn, R.D., Ritzmann, R.E., and Choi, J. 2001. Finite Element Analysis of strains in the *Blaberus* cockroach leg segment while climbing. *Intelligent Robots and Systems IEEE/RSJ Proceedings* 2:833-838.
- Kindermann, T. 2002. Behavior and adaptability of a six-legged walking system with highly distributed control. *Adaptive Behavior* 9:16-41.
- Kingsley, D.A., Quinn, R.D., and Ritzmann, R.E. 2003. A cockroach inspired robot with artificial muscles. *International Symposium on Adaptive Motion of Animals and Machines (AMAM)*, Kyoto, Japan.
- Kram, R., and Full R.J. 1997. Three-dimensional kinematics and limb kinetic energy of running cockroaches. *Journal of Experimental Biology* 200:1919-1929.
- Larsen, G.S., Frazier, S.F., and Zill S.N. 1997. The tarso-pretarsal chordotonal organ as an element in cockroach walking. *Journal of Comparative Physiology A* 180:683-700.
- Nelson, G. M., and Quinn, R. D. 1999. Posture Control of a Cockroach-like Robot, *IEEE Control Systems*, Vol. 19 (2).
- Nelson, G.M. 1995. *Modeling and simulation of an insect-like Hexapod*, M.S. Thesis, Case Western Reserve University.
- Nelson, G.M. 2002. Learning about control of legged locomotion using a hexapod robot with compliant pneumatic actuators. Doctoral Thesis, Case Western Reserve University.
- Noah, J.A., Quimby, L., Frazier, S.F. and Zill, S.N. 2004. Sensing the effect of body load in legs: responses of tibial campaniform sensilla to forces applied to the thorax in freely standing cockroaches. *Journal of Comparative Physiology A* 190:201-215.
- Noah, J.A., Quimby, L., Frazier, S.F., and Zill, S.N. 2004. Walking on a 'peg leg': extensor muscle activities and sensory feedback after distal leg denervation in cockroaches. *Journal of Comparative Physiology A* 190:217-231.
- Pratt, C.A. 1995. Evidence of positive force feedback among hindlimb extensors in the intact standing cat. *Journal of Neurophysiology* 73:2578-2583.
- Prochazka, A. 1996. Proprioceptive feedback and movement regulation, In: *Handbook of Physiology*, edited by L. Rowell and J. Shepard. New York: American Physiol. Soc., pp. 89-127.

- Quinn, R.D., Nelson, G.M., Ritzmann, R.E., Bachmann, R.J., Kingsley, D.A., Offi, J.T. and Allen, T.J. (2003), "Parallel Strategies For Implementing Biological Principles Into Mobile Robots," *Int. Journal of Robotics Research*, Vol. 22 (3) pp. 169-186.
- Ramasubramanian, S., Flannigan, W.C., Nelson, G., Quinn, R. and Zill, S.N. 1999. Modeling of load sensing during the stance phase of cockroach walking. In *Climbing and Walking Robots*. Eds: Virk, GS, Randall, M and Howard, D. Univ. of Portsmouth, Professional, Engineering Publishing Ltd, London. Pp. 17-28.
- Ramos, A., Fonseca, F., Simoes, J.A. (2006) Simulation of Physiological Loading in Total Hip Replacements. *Biomech Eng*. 128(4):579-587.
- Ridgel, A.L., Frazier, S.F., DiCaprio, R.A., and Zill S.N. 2000. Encoding of forces by cockroach tibial campaniform sensilla: implications in dynamic control of posture and locomotion. *Journal of Comparative Physiology A* 186:359-374.
- Ritzmann, R.E., R.D. Quinn, J.T. Watson, S.N. Zill (2000) Insect walking and biorobotics: A relationship with mutual benefits. *Bioscience*, Vol. 50, No. 1, Jan. 2000.
- Schmitz, J. 1993. Load-compensating reactions in the proximal leg joints of stick insects during standing and walking. *Journal of Experimental Biology* 183, 15-33. Libersat, F., Clarac, F., and Zill, S.N. 1987. Force-sensitive mechanoreceptors of the dactyl of the crab: single-unit responses during walking and evaluation of function. *Journal of Neurophysiology* 57:1618-1637.
- Szücs, A., P. Varona, A. Volkovskii, H. D. I. Abarbanel, M. I. Rabinovich, and A. I. Selverston (2000). Interacting Biological and Electronic Neurons Generate Realistic Oscillatory Rhythms, *Neuroreport*, 11:563-569.
- Watson, J.T., Ritzmann, R.E., and Pollack, A.J. 2002. Control of climbing behavior in the cockroach, *Blaberus discoidalis*. II. Motor activities associated with joint movement. *Journal of Comparative Physiology A* 188, 55-69.
- Webb B, Consi TR. (2001). *Biorobotics - Methods and Applications*. AAAI Press / MIT Press.
- Weidemann, H.J., and Pfeiffer, F. 1995. The Six-Legged TUM Walking Robot, *Intelligent Robots and Systems*, V.Graefe (Editor), Elsevier Science, 1026-1033.
- Wendler, G. 1966. The co-ordination of walking movements in arthropods. In *Nervous and Hormonal Mechanisms of Integration. Symposia of Society for Experimental Biology* 20, 229-249.
- Zill, S.N. 1990. Mechanoreceptors and proprioceptors., in: *Cockroaches as Models for Neurobiology: Applications in Biomedical Research*. Huber I, Masler, E P and Rao, B R, eds. CRC Press, Boca Raton, Florida, Vol. II, pp. 247- 267
- Zill, S., Schmitz, J., and Büschges, A. 2004. Load sensing and control of posture and locomotion. *Arthropod Structure and Development* 33:273-286 (Special Issue: Arthropod Locomotion Systems: from Biological Materials and Systems to Robotics, RE Ritzmann, SN Gorb and RD Quinn, eds.)
- Zill, S.N., and Moran, D.T. 1981. The exoskeleton and insect proprioception. I. Responses of tibial campaniform sensilla to external and muscle regenerated forces in the American cockroach, *Periplaneta americana*. *Journal of Experimental Biology* 91:1-24.
- Zill, S.N., Frazier, S.F., Neff, D., Quimby, L., Carney, M., DiCaprio, R., Thuma, J., and Norton, M. 2000. Three dimensional graphic reconstruction of the insect exoskeleton through confocal imaging of endogenous fluorescence. *Microscopy Research Technique* 48:367-384.
- Zill, S.N., Ridgel, A.L., DiCaprio, R.A., and Frazier, S.F. 1999. Load signalling by cockroach trochanteral campaniform sensilla, *Brain Research* 822 271-275.

Novel Robotic Applications using Adaptable Compliant Actuation. An Implementation Towards Reduction of Energy Consumption for Legged Robots

Björn Verrelst, Bram Vanderborght*
Ronald Van Ham, Pieter Beyl and Dirk Lefeber
*Robotics&Multibody Mechanics Research Group
Vrije Universiteit Brussel
Belgium*

1. Introduction

Currently, robotics is one of the fields undergoing great scientific innovation and expansion. Automation, via robotic applications, in production industries has grown increasingly. In particular, multifunctional manipulators executing several industrial assembly tasks, i.e. large, fixed-base industrial robots executing “pick-and-place” tasks, with control strategies focused on precision end-effect positioning. Additionally, high precision robotic applications in the framework of medical (surgical) and microchips production are needed especially regarding high precision position control. Electrical motors combined with adapted reduction elements envelop the most commonly used actuator technology at the moment. The developed control algorithms are specifically optimized for position control, possibly in combination with force control. New developments in the area of robotics, however, have currently allowed for the development of multi-legged mobile robots and service robots oriented towards closer human/robot interaction. Especially in Japan, where a large number of corporations each work on a private humanoid robot, strong developments steer towards new applications in social service, domotics, rehabilitation and prosthesis. Important is that these domains require distinct features from the robotic system, such as special dynamical properties for reasons of chock absorption, energy consumption, and flexible and safe human/robot interaction.

Present robots mainly are actuated with electrical drives, with the required reduction units, since this technology is well known. However, the use of these reduction units renders the joint to be more rigid, encompassing several disadvantages vis-à-vis applications of human/robot interactions. In this manner, the structure of the reduction elements is heavily burdened through jolting of highly reflected inertia values. Furthermore, this manner of actuation leaves small room for “soft” robot/human interaction and complicates the exploitation of the system’s natural dynamics. The latter encompasses generating robot movements closely associated to the natural, passive movement of the system. However, this requires compliant joints of which ideally stiffness can be manipulated. In a similar way, humans set up their joint stiffness, through relaxation of the muscles, according to the action or task performed. One of the positive consequences of movement within the natural dynamics

of the system is the decreased energy consumption, especially important for mobile robots that require an onboard energy source. If the desired movements fall within passive dynamics, certain dynamic limits of the actuator/application frame can moreover shift.

One way to deal with compliance control is by implementing active impedance control using standard electrical actuation. With this kind of control the stiffness is set artificially, but the response time of these systems is fairly low due to several limitations such as sensor resolution, motor control bandwidth, sensor noise, etc.... Moreover, the exploitation of natural dynamics, by means of energy storage in an elastic element, and reduction of shock effects is not possible with such an approach. Reduction of impact effects is essential first for protection with respect to the robot structure itself, e.g. during touch-down impacts of multi-legged robots, and secondly for protection of humans, close to the robot, during unplanned interactions. An alternative for the implementation of compliance control, which tackles the disadvantages previously mentioned, is the use of actuation with inherent mechanical impedance control.

In this context, the "Robotics & Multibody Mechanics" research group of the "Vrije Universiteit Brussel" (VUB) started research, some ten years ago, for alternatives by developing the pleated pneumatic artificial muscle (Daerden et al., 2001). The compressibility of the air in this muscle ensures for the inherent compliance. For a joint driven by two muscles, positioned antagonistically, the joint position can be controlled while adapting compliance. Such muscles are currently being used for the actuation of the biped robot "Lucy" (Verrelst et al., 2005c) and a manipulator arm with direct human/robot interaction (Van Damme et al., 2005), and will be used to power a semi-active ankle foot prosthesis and an exoskeleton for the lower extremities to assist during rehabilitation of paraplegic patients. These experimental setups are under study to develop specific control strategies for steering a robot with compliant joints, and more specifically adaptive compliance. In this framework, the robot "Lucy" is steered through trajectory control in the joints in combination with exploitation of the natural dynamics of the robot system, using the adaptive compliance characteristic of the actuators.

This publication will report on the implementation of compliant actuation in the biped "Lucy" and a one degree of freedom pendulum setup, focusing on the reduction of energy consumption by exploitation of natural dynamics. In the current state of this research the tracking control strategies developed for the compliant actuators allow the biped to walk continuously at moderate speeds while changing walking speed and step length. The control strategy to combine compliance setting with trajectory control, influencing the eigenfrequency of the system, so far has been successfully implemented in a reduced configuration such as an experimental pendulum setup actuated by one antagonistic pair of two artificial muscles.

First an overview is given on the research of legged robots in combination with exploitation of natural dynamics, followed by an overview of existing variable compliant actuation systems. Next, a description of the pleated pneumatic artificial muscle, which is the present choice to implement adaptable compliance, is given. Subsequently, the low-level control strategies allowing both position and compliance control in an antagonistic setup of two such artificial muscles are described. The implementation of these control strategies in a one degree of freedom pendulum structure is the main subject, for which the reduction of energy consumption is shown by experimental results. Finally, the extension of the tracking control to the complete biped is discussed briefly and its experimental validation on joint trajectory tracking for continuous walking of the biped "Lucy" on a treadmill is given.

1.1 Legged Locomotion and Natural Dynamics

Legged locomotion can be classified in terms of their overall control strategy. The more recent robots are dynamically balanced machines, whereas the older machines were statically balanced. Statically balanced robots keep the centre of mass within the polygon of support in order to maintain postural stability. To avoid inertial effects these machines move rather slow, contrary to dynamically balanced robots where the inertial effects are taken into account in the different control strategies. When the control unit not only takes these inertial effects into account but also exploits them, the term natural dynamics arises.

Natural dynamics or passive dynamics is the unforced response of a system under a set of initial conditions. In general, for legged locomotion these natural dynamics are not or only partially exploited. Examples of exploitation of the natural dynamics are the swing-leg swinging freely without hip actuation or the body and stance-leg pivoting as an inverted pendulum around an un-actuated ankle. Legged systems that walk completely without actuation are the so called "Passive Walkers". These machines are only powered by gravity and they are mechanically tuned in order to walk down a sloped surface. These "Passive Walkers" could be pointed out as very energy efficient but unfortunately they are of little practical use. A minimum actuation should be provided to walk on level ground to overcome friction and impact losses. Anyway, it is important to exploit the natural dynamics by trying to incorporate the unforced motion of a system instead of ignoring or avoiding it. Doing so could positively affect energy consumption and control efforts.

One of the first to incorporate passive dynamics for legged locomotion was Matsuoka (Matsuoka, 1980) and later Raibert (Raibert, 1986). The latter became one of the pioneers in dynamic walking with his several one or more legged hopping robots. For these systems he used a pneumatic spring in the telescopic legs to influence and exploit the passive dynamics in the vertical direction. Exploiting passive dynamics by means of the stance leg pivoting freely as an inverted pendulum was incorporated in the biped walkers of Takanishi (Takanishi, 1985), Lee and Liao (Lee et al., 1988). At the end of the eighties Thompson and Raibert (Thompson et al., 1989) studied the additional exploitation of natural dynamics of the hip motion by placing a torsional spring. At the same time McGeer (McGeer, 1990) built and studied a passive walker without compliant elements. Later he analysed theoretically and by means of simulation walkers with hip compliance but with the body being a point-mass. During the nineties the group of Andy Ruina (1998) (Garcia et al., 1998) studied in more detail the models of McGeer and extended the two dimensional model to three dimensions while building several Passive Walkers. In the second half of the nineties the group of Buehler (Gregorio et al, 1997) built a legged hopping monopod following the examples of Raibert using electrical actuation combined with a torsional spring in the hip and a linear spring in the telescopic leg. The control strategy was to calculate the passive dynamic trajectories with the correct initial conditions as a function of the desired forward speed while in parallel with these trajectories standard active control was used to cope with imperfections of the modelling of the passive dynamics. A more intuitive control but still focussing on exploiting natural dynamics was done at MIT by Pratt making use of the "Series Elastic Actuators" (Pratt et al., 1995) in the two legged robot "Spring Flamingo" (Pratt et al., 1998). After 2000, Quartet III was built (Osuka et al., 2000), this quadruped starts walking down a sloped surface in an active way and gradually decreases control input to transfer to passive walking. Asano and Yamakita introduced the "virtual gravity field" (Asano et al., 2000) for horizontal walking in order to exhibit virtual passive walking based

on McGeers models but with hip and ankle actuation. Wisse and van der Linde (Wisse et al., 2001) studied a 3D Passive Walker with a pelvic body at the university of Delft.

Most of these models use only the inertial properties to determine the eigenfrequency and additionally fixed compliance of mechanical linear or rotational springs. As a result the eigenfrequency of these systems is set during construction which limits the different passive walking patterns and walking speeds. Flexibility, with the ability to change this natural frequency, is increased by implementing passive elements with variable compliance. In this context the group of Takanishi developed the two-legged walker WL-14 (Yamagushi, 1998), where a complex non-linear spring mechanism makes changes in stiffness possible. A more elegant way to implement variable compliance is to use pneumatic artificial muscles in an antagonistic setup, where the applied pressures determine stiffness. Research on this topic was done by Van der Linde (van der Linde, 1998), Wisse (Wisse, 2001) and Caldwell (Davis et al., 2001) by implementation of McKibben muscles.

1.2 Variable Compliance Actuators

An important contribution in the research towards soft actuators has been given by Pratt with the development of the "series elastic actuator" (Pratt et al., 1995). It consists of a motor drive in series with a spring. The disadvantage of such a setup is that the stiffness can not be changed. Nowadays, more and more research is performed to make the compliance of the actuator adaptable. The Robotics Institute at Carnegie Mellon University developed the "Actuator with Mechanically Adjustable Series Compliance" (AMASC) (Hurst et al., 2004). It has fibreglass springs with a large energy storage capacity. The mechanism has two motors, one for moving the position and the other for controlling the stiffness. However the mechanism is so complex and heavy that it can't be used in mobile robots. The novel electro-mechanical "Variable Stiffness Actuation" (VSA) motor (Bicchi et al., 2004) of the university of Pisa has been designed for safe and fast physical human/robot interaction. A timing transmission belt connects nonlinearly the main shaft to an antagonistic pair of actuator pulleys connected to position-controlled back-drivable DC motors. The belt is tensioned by springs. Concordant angular variations control displacements of the main shaft, while the opposite variations of the two DC motors generate stiffness variations. The "Biologically Inspired Joint Stiffness Control" (Migliore et al., 2005) can be described as two antagonistic coupled Series Elastic Actuators, where the springs are made non-linear. At Northwestern University the "Moment arm Adjustment for Remote Induction Of Net Effective Torque" (MARIONET) (Sulzer et al., 2005) actuator has been developed. This rotational joint uses cables and transmission to vary the moment arm such that the compliance and equilibrium position is controllable. Special is that this device doesn't use an elastic element, the system moves against a conservative force field created by a tensioner. Hollander and Suger (Hollander, 2004) describe several variable stiffness systems by changing the "effective" structure of spring elements. One of them uses a spring leave of which the active length is changed by an extra actuator. This principle has also been used for the "Mechanical Impedance Adjuster" (Morita et al., 1995) developed at the Sugano Lab. And a complete 3D joint with variable compliance, by using only 2 motors, has been developed (Okada et al., 2001) for a shoulder mechanism, but the compliances of the different DoF's are coupled. Another recent approach is the MACCEPPA actuator (Van Ham et al., 2005) which uses two motors and only one spring element. The compliance and rest position of the actuator can be set independently, each by a separate motor, due to a specific geometric positioning of the spring. Finally, as already mentioned in the previous section, variable compliance can be

achieved with pneumatic artificial muscles in an antagonistic setup. In fact, any non-linear compliant element in an antagonistic setup will do for this purpose, but artificial muscles provide an elegant and compact solution.

2. Pleated Pneumatic Artificial Muscle

2.1 Force Characteristic

A pneumatic artificial muscle is essentially a membrane that will expand radially and contract axially when inflated, while generating high pulling forces along the longitudinal axis. Different designs have been developed. The best known is the so called McKibben muscle (Schulte, 1961). This muscle contains a rubber tube which expands when inflated, while a surrounding netting transfers tension. Hysteresis, due to dry friction between the netting and the rubber tube, makes control of such a device rather complicated. Typical for this type of muscle is a threshold level of pressure before any action can take place. The main goal of the new design (Daerden et al., 2001) was to avoid friction, thus making control easier while avoiding the threshold. This was achieved by arranging the membrane into radially laid out folds that can unfurl free of radial stress when inflated. Tension is transferred by stiff longitudinal fibres that are positioned at the bottom of each crease.



Fig. 1. Pleated Pneumatic Artificial Muscles at 3 different contraction levels.

A photograph of the inflated and deflated state of the Pleated Pneumatic Artificial Muscle (PPAM) is given in Fig. 1. If we omit the influence of elasticity of the high tensile strength material used for the fibres, the characteristic for the generated force is given by:

$$F = pl^2f(\varepsilon, R/l) \quad (1)$$

where p is the applied gauge pressure, l the muscle's full length, R its unloaded radius and ε the contraction. The dimensionless function f , which depends only on contraction and geometry, is given for different values of broadness R/l on the left graph of Fig. 2.

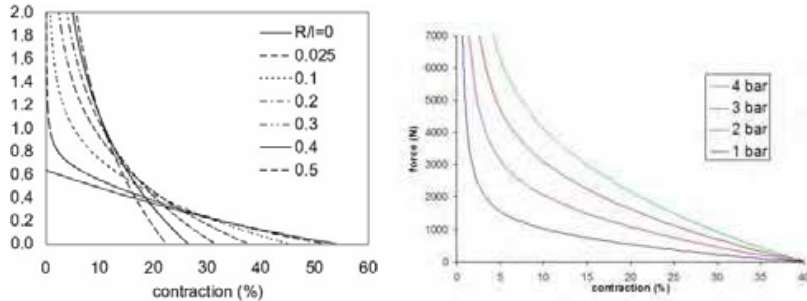


Fig. 2. (left) Dimensionless force function for different broadness ; (right) Generated force of a muscle with initial length 11cm and unloaded diameter 2.5cm.

The thicker the muscle, the less it contracts and the higher the forces it generates. Contraction can reach up to 54% in a theoretical case with $R/l=0$. At low contraction forces are extremely high causing excessive material loading, and the generated forces drop too low for large contraction. Thus contraction is bounded between two limits, 5 and 35%, in practise. The graph on the right of Fig. 2 gives the generated force for different pressures of a muscle with initial length 11cm and unloaded diameter 2.5cm. Forces up to 5000N can be generated with gauge pressure of only 3bar while the device weighs about 100g.

2.2 Antagonistic Muscle Setup: Generated Torque

Pneumatic artificial muscles can only pull. In order to have a bi-directionally working revolute joint one has to couple two muscles antagonistically. The muscle connections -pull rods and lever mechanism- are designed such that the muscle's highly non-linear force-length characteristic is transformed to a more flattened torque-angle characteristic. The left picture of Fig. 3 shows the straightforward connecting principle.

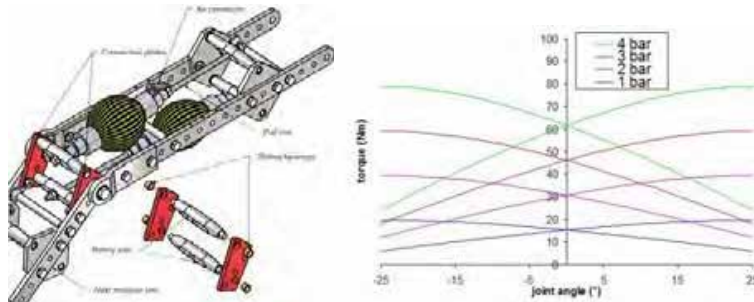


Fig. 3. (left) straightforward connection of the antagonistic muscle setup ; (right) generated torques by each muscle of an antagonistic setup at different pressure levels.

Depending on the desired function of the joint, the dimensions can be chosen in order to meet the needs of the specified joint function, not only in torque levels but also in range of motion. The torque characteristics of the pendulum (discussed below) are shown on the right of Fig. 3. Both torques are given for different pressure values. Taking into account

equation (1) and if r_1 and r_2 define the leverage arm of the agonist and antagonist muscle respectively, the joint torque (T) is given by following expression

$$T = T_1 - T_2 = p_1 l_1^2 r_1 f_1 - p_2 l_2^2 r_2 f_2 = p_1 t_1(\theta) - p_2 t_2(\theta) \quad (2)$$

with p_1 and p_2 the applied gauge pressures in agonist and antagonist muscles respectively which have lengths l_1 and l_2 . The dimensionless force functions of both muscles are given by f_1 and f_2 . The functions t_1 and t_2 , in equation (2), are determined by the choices made during the design phase and depend on the joint angle θ . Thus joint position is influenced by weighted differences in gauge pressures of both muscles.

2.3 Antagonistic Muscle Setup: Compliance

The PPAM has two sources of compliance: gas compressibility and the dropping force to contraction characteristic. The latter effect is typical for pneumatic artificial muscles while the first is similar to standard pneumatic cylinders. Joint stiffness, the inverse of compliance, for the considered revolute joint can be obtained by the angular derivative of the torque characteristic in equation (2):

$$K = \frac{dT}{d\theta} = \frac{dp_1}{d\theta} t_1 + p_1 \frac{dt_1}{d\theta} - \frac{dp_2}{d\theta} t_2 - p_2 \frac{dt_2}{d\theta} \quad (3)$$

The terms $dp/d\theta$ represent the share in stiffness of changing pressure with contraction, which is determined by the action of the valves controlling the joint and by the thermodynamic processes taking place. If the valves are closed and if we assume polytropic compression/expansion the pressure changes inside the muscle are a function of volume changes:

$$P_i V_i^n = P_{i0} V_{i0}^n \text{ with } P_i = p_i + P_{atm} \quad (4)$$

leading to:

$$\frac{dp_i}{d\theta} = -n(P_{atm} + p_{i0}) \frac{V_{i0}^n}{V_i^{n+1}} \frac{dV_i}{d\theta} \quad (5)$$

With P_i , V_i the absolute pressure and volume of muscle i , P_{i0} the absolute initial pressure, V_{i0} the initial volume when muscle i was closed and p_i , p_{i0} the gauge pressure and initial gauge pressure. n is the polytropic index and P_{atm} the atmospheric pressure.

Taking the torque characteristics as an example the following reasoning can be made for closed muscles. An increase of the angle θ results in an increase of the torque generated by the extensor muscle (see right Fig. 3) while its volume decreases (expanded muscles have lower torques). Thus $dt_1/d\theta > 0$ and $dV_1/d\theta < 0$. For the flexor muscle the actions will be opposite. Combining equations (3), (4) and (5) with this information gives:

$$K = k_1 p_{i0} + k_2 p_{20} + k_{atm} P_{atm} \quad (6)$$

with:

$$k_i = n t_i \frac{V_{i0}^n}{V_i^{n+1}} \left| \frac{dV_i}{d\theta} \right| + \frac{V_{i0}^n}{V_i^n} \left| \frac{dt_i}{d\theta} \right| > 0 \quad i = 1, 2 \quad (7)$$

$$k_{atm} = k_1 + k_2 - \left| \frac{dt_1}{d\theta} \right| - \left| \frac{dt_2}{d\theta} \right|$$

The coefficients k_1 , k_2 , k_{atm} are a function of the angle and are determined by the geometry parameters of the joint and muscles. From equation (6) the conclusion is drawn that a passive spring element is created with an adaptable stiffness controlled by the weighted sum of both initial gauge pressures when closing the muscle. Since stiffness is depending on a sum of gauge pressures while position is determined by differences in gauge pressure, the angular position can be controlled while setting stiffness.

3. Trajectory control with adaptable compliance

In this section the proposed strategy of combining adaptable passive behaviour with trajectory control is illustrated. The control architecture consists of two parts: the joint trajectory tracking controller which calculates the necessary torque to track a desired trajectory and the compliance controller to reduce energy consumption and control efforts.

3.1 Joint Trajectory Tracking Controller

This paragraph discusses the joint trajectory tracking controller of which the separate units are depicted in Fig. 4. The tracking controller is divided into a computed torque controller, a delta-p unit and a pressure bang-bang controller. The computed torque controller calculates the required joint torque based on the pendulum dynamics. The delta-p unit translates these calculated torques into desired pressure levels for the two muscles of the antagonistic set-up. Finally the bang-bang pressure controller determines the necessary valve actions to set the correct pressures in the muscles. In the next sections the different elements of the control structure are discussed in more detail.

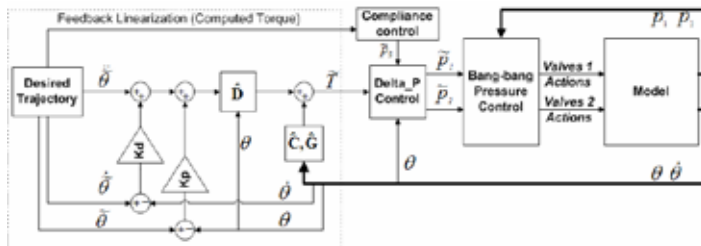


Fig. 4. Trajectory tracking control scheme.

3.1.1 Computed Torque

In the first module, the joint torques are calculated using the popular computed torque technique consisting of a feedforward part and a PID feedback loop. This results in the following calculation:

$$\tilde{T} = C_e(\theta, \dot{\theta})\ddot{\theta} + G_e(\theta) + D_e(\theta) \left(\ddot{\theta} - K_p(\theta - \tilde{\theta}) - K_i \int (\theta - \tilde{\theta}) - K_d(\dot{\theta} - \dot{\tilde{\theta}}) \right) \quad (8)$$

The parameters D_e , C_e and G_e contain the estimated values of the inertia, centrifugal and gravitational parameters of the system under study. The feedback parameters K_p , K_i and K_d are manually tuned.

3.1.2 Delta-p Unit

The computed torque \tilde{T} is fed to the delta-p control unit, which calculates the required pressure values to be set in the muscles. These two gauge pressures are generated as follows:

$$\tilde{p}_1 = \frac{\tilde{p}_s}{\tilde{f}_1(\theta)} + \Delta\tilde{p} \quad (9a)$$

$$\tilde{p}_2 = \frac{\tilde{p}_s}{\tilde{f}_2(\theta)} - \Delta\tilde{p} \quad (9b)$$

with p_s a parameter that is used to influence the sum of pressures and consequently the joint stiffness and $\Delta\tilde{p}$ a parameter that influences the difference in pressure of the two muscles in order to control the generated torque. The functions $\tilde{f}_i(\theta)$ ($i=1,2$) represent the torque characteristics of the antagonistic muscle setup and are calculated with estimated values of the muscle force functions and geometrical parameters. Expression (2) allows to link the required torque to the required pressure values in the muscles:

$$\tilde{T} = \tilde{p}_1\tilde{f}_1(\theta) - \tilde{p}_2\tilde{f}_2(\theta) = (\tilde{f}_1(\theta) + \tilde{f}_2(\theta))\Delta\tilde{p} \quad (10)$$

If the calculated pressure values \tilde{p}_i ($i=1,2$) of equations (9) are set in the muscles, the generated torque depends only on $\Delta\tilde{p}$ and is independent of the joint stiffness parameter p_s , in case the modelling would be perfect. This means that joint stiffness is changed without affecting the joint angular position. Feeding back the knee angle θ and introducing the torque \tilde{T} , expression (10) can be used to determine the required $\Delta\tilde{p}$:

$$\Delta\tilde{p} = \frac{\tilde{T}}{\tilde{f}_1(\theta) + \tilde{f}_2(\theta)} \quad (11)$$

The delta-p unit is actually a feedforward calculation from torque level to pressure level, using the kinematic model of the muscle actuation system. The calculated $\Delta\tilde{p}$ affects the torque required to track the desired trajectory, while p_s is introduced to determine the sum of pressures which influences the stiffness of the joint which will be discussed in section 3.2.

3.1.3 Bang-bang Pressure Controller

The weight of the valves controlling the muscles should be taken as low as possible. But since most pneumatic systems are designed for fixed automation purposes where weight is not an issue at all, most off-the-shelf proportional valves are far too heavy for this application. Thus a proper design of the pressure control is of great importance. In order to realize a lightweight rapid and accurate pressure control, fast switching on-off valves are used. The pneumatic solenoid valve '821 2/2 NC' made by Matrix weighs only 25g. With their reported opening times of about 1ms and flow rate of 180 Stdl/min, they are about the fastest switching valves of that flow rate currently available.

To pressurize and depressurize the muscles, which have a varying volume up to 400ml, a number of these small on-off valves are placed in parallel. Obviously the more valves used the higher the electric power consumption, the price and also the weight will be. Simulations of the pressure control of a constant volume resulted in a choice of two inlet and four outlet valves. The different number between inlet and outlet comes from the asymmetric pressure conditions between inlet and outlet, combined with the aim to create equal muscle's inflation and deflation times.

To connect the 6 valves into one compact pressure regulating valve, two special collectors were designed. These collectors replace the original aluminium connector plates of the valves, resulting in a weight of the complete pressure regulating valve of about 150gr. The left of Fig. 5 shows a picture of the pressure valves with their speed-up circuit. The speed-up circuit reduces opening and closing times of the valves. The pressure control in a volume is achieved with a bang-bang controller with various reaction levels depending on the difference between measured and desired pressure (right of Fig.5.). If this difference is large, two inlet or four outlet valves, depending on the sign of the error, are opened. If this difference is small, only one valve is switched and when the error is within reasonable limits no action is taken, leaving the muscle closed. More detailed information on the valve system can be found in the work of Van Ham (Van Ham et al., 2005).

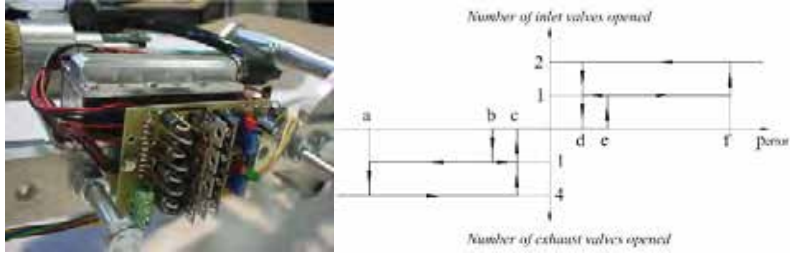


Fig. 5. (left) Picture of valve collector; (right) Multi-level bang-bang control scheme with dead zone ($a = -60$, $b = -25$, $c = -20$, $d = 20$, $e = 25$, $f = 60$ mbar).

3.2 Compliance Controller

In this section a strategy to calculate an appropriate value of p_s is described. The idea is to fit the natural compliance with the required ones. The desired stiffness related to the desired trajectory \tilde{K}^{Traj} , the inverse of the compliance, is calculated as the derivative of the torque \tilde{T} , required to track the trajectory, with respect to the desired joint angle $\tilde{\theta}$. The torque \tilde{T} is found by inverse dynamics.

$$\tilde{K}^{Traj} = \frac{d\tilde{T}}{d\tilde{\theta}} = \frac{d}{d\tilde{\theta}} \left(D_e(\tilde{\theta})\ddot{\tilde{\theta}} + C_e(\tilde{\theta}, \dot{\tilde{\theta}})\dot{\tilde{\theta}} + G_e(\tilde{\theta}) \right) \quad (12)$$

The natural actuator stiffness K^{Act} can be found by deriving equation (10) with respect to the desired joint angle $\tilde{\theta}$:

$$K^{Act} = \frac{d\tilde{p}_1}{d\tilde{\theta}} \tilde{t}_1 + \tilde{p}_1 \frac{d\tilde{t}_1}{d\tilde{\theta}} - \frac{d\tilde{p}_2}{d\tilde{\theta}} \tilde{t}_2 - \tilde{p}_2 \frac{d\tilde{t}_2}{d\tilde{\theta}} \quad (13)$$

The required pressure slopes are a combination of equation (5), valid for closed muscles, with equations (9), in which the initial pressure p_0 is set equal to \tilde{p}_1 :

$$\frac{d\tilde{p}_i}{d\tilde{\theta}} = - \left[\frac{\tilde{p}_S}{\tilde{t}_i} + \left(P_{atm} + (-1)^{i+1} \Delta p \right) \right] \left(n \frac{\hat{V}_{i_0}^n}{\hat{V}_i^{n+1}} \frac{d\hat{V}_i}{d\tilde{\theta}} \right) \quad i = 1, 2 \quad (14)$$

Combining equations (13) and (14) while setting expected actuator stiffness K^{Act} equal to the desired stiffness \tilde{K}^{Traj} , the optimal actuator stiffness parameter \tilde{p}_S^{Opt} can be calculated:

$$\tilde{p}_S^{Opt} = \frac{K^{Traj} - \Delta\tilde{p}A - P_{atm}B}{C} \quad (15)$$

with:

$$A = -\frac{n\dot{t}_1}{\dot{V}_1} \frac{d\hat{V}_1}{d\theta} - \frac{n\dot{t}_2}{\dot{V}_2} \frac{d\hat{V}_2}{d\theta} + \frac{d\dot{t}_1}{d\theta} + \frac{d\dot{t}_2}{d\theta} \quad (16a)$$

$$B = -\frac{n\dot{t}_1}{\dot{V}_1} \frac{d\hat{V}_1}{d\theta} + \frac{n\dot{t}_2}{\dot{V}_2} \frac{d\hat{V}_2}{d\theta} \quad (16b)$$

$$C = -\frac{n}{\dot{V}_1} \frac{d\hat{V}_1}{d\theta} + \frac{n}{\dot{V}_2} \frac{d\hat{V}_2}{d\theta} + \frac{1}{\dot{t}_1} \frac{d\dot{t}_1}{d\theta} - \frac{1}{\dot{t}_2} \frac{d\dot{t}_2}{d\theta} \quad (16c)$$

Note that the initial volume V_{i_0} when closing a muscle, is set equal to the actual volume V_i , analogously as for the pressure. Using the value \tilde{p}_S^{Opt} in the calculation of the control variables in equations (9) reduces the energy consumption. However, there still exists a strong dependence on the controller parameters, such as bang-bang pressure control reaction levels and feedback gains of the computed torque controller. E.g. decreasing the reaction levels of the bang-bang controller for opening a valve would increase valve switching, due to the delay time of the valves.

4. Simulation and Experimental Results

In this section the proposed compliance controller in combination with the trajectory tracking controller is discussed both through simulations and experiments on a physical pendulum setup. This single pendulum, driving a load at its endpoint, is actuated by one antagonistic pair of artificial muscles as described in section 2.2.

4.1 Calculation of Energy Consumption

The energy consumption for compressed air depends not only on the air mass flows but is related to the thermodynamic conditions of the compressed air supply source. One way to give an idea of energy consumption is to calculate the exergy associated with the particular pneumatic air mass flow. Exergy is in fact the maximum amount of energy, with respect to the surrounding environment, which can be transformed into useful work. For example, the air mass flow entering the muscle system comes from a compressed air reservoir at certain temperature and pressure level. The surrounding environment is regarded as the atmosphere. The exergy of the pneumatic power supply is then calculated as the minimal work needed to compress the atmospheric air to the pressure supply conditions. For a compressor, the minimal work needed to compress air from pressure level p_{atm} to p_1 is done at isothermal conditions and can be calculated as follows (Rogers et al., 1992):

$$\dot{W}_{isotherm} = \dot{m}_{air} r T_{atm} \ln \frac{p_1}{p_{atm}} \quad (17)$$

hereby assuming the air to behave as a perfect gas. The symbol \dot{m}_{air} represents the total air mass flowing through the compressor, r is the dry air gas constant and T_{atm} is the temperature of the atmosphere expressed in Kelvin. The absolute supply pressure level is set at 7bar, the atmospheric absolute pressure at 1bar and the atmospheric temperature is 293K. This is just

one way to calculate the energy consumption. The absolute energy consumption is not the most important aspect, however. It is the reduction of the energy consumption that one can achieve, when matching system and trajectory compliances, which is the main focus.

4.2 Simulation Results

4.2.1 Simulator

Most parts of the system are highly non-linear: force/torque-contraction of the muscles, the thermodynamic processes, the mechanical load of the pendulum, the pressurized air flows through the discrete number of valves,... which makes it difficult to perform stability/convergence analysis. In order to evaluate the proposed control strategy while taking into account non-linearity, a hybrid simulator was created, meaning that both the pneumatics and mechanics are put together in one dynamic simulation model. The pressure variation inside the muscle is represented by first order differential equations deduced from the first law of thermodynamics for an open system while assuming a perfect gas for the compressed air. The orifice valve flows are derived from the model presented by ISO6358 standard (International Standard, 1998). The integration of these first order differential equations coupled with the mechanical differential equations of the pendulum and the muscle torque/contraction characteristics results in the calculated torques. Doing so includes the muscle actuator characteristics, which allows the limitations in the motion of the pendulum to be studied. Detailed information on the simulation model can be found in (Verrelst, 2005a) for a one degree of freedom setup and in (Verrelst, 2005b) for the complete biped, elaborated on in section 5. To test the robustness, model parameter uncertainties are introduced in the simulation model, e.g. 5% on the mass and CoG (Centre of Gravity), 10% on the inertia and 5% on both force functions of the antagonistic set-up. A time delay of 1ms for closing and opening of the valves and a sampling time of 2ms of the controllers are regarded.

4.2.2 Results and discussion

To evaluate the proposed control architecture a sinusoidal trajectory is imposed with a linearly increasing frequency from 1.5 Hz to 2 Hz (chirp function). Due to the increasing frequency the stiffness has to be adapted continuously. To show the effectiveness of the proposed compliance controller the results of an optimal p_S^{Opt} as calculated with equation (15) have been compared with the results obtained with some deviations on it: $p_S = 1.5p_S^{\text{Opt}}$ and $p_S = 0.5p_S^{\text{Opt}}$. The influence of the parameter p_S on the exergy consumption and mean angle tracking error is given in Table 1. This shows that tracking the desired trajectory with p_S^{Opt} consumes less energy, up to 6 times less than the considered deviated states (in simulation), while the tracking error is not deteriorated.

p_S	Exergy consumption (J)	Mean angle error (°)
$p_S = 0.5p_S^{\text{Opt}}$	5.28	0.09
$p_S = p_S^{\text{Opt}}$	0.87	0.064
$p_S = 1.5p_S^{\text{Opt}}$	5.63	0.122

Table 1. Exergy consumption and mean angle error for different p_s values.

The effect of choosing the optimal p_S^{Opt} can also be seen in the graphs of Fig. 6. These graphs show the valve actions taken by the bang-bang pressure controller. Note that in these figures closed valves are represented by a horizontal line depicted at respectively 1.5, 2.0 and 2.6bar pressure level, while a small peak upwards represents one opened inlet valve, a small peak downwards one opened exhaust valve.

The number of valve actions is significantly lower when the compliance setting is at the optimal value. On the right, the desired and actual pressures are depicted. In the optimal case the desired pressure is nearly the natural pressure already present in the muscle. Only at certain instants a little energy input has to be provided to the system in order to increase the swing frequency. In the cases where the compliance setting is not optimal, significantly more valve switching is required. The imposed trajectory differs substantially from the natural movement of the pendulum causing a lot of valve switching and consequently energy dissipation. The actual motion, however, is the same because it is controlled by the joint trajectory tracking controller.

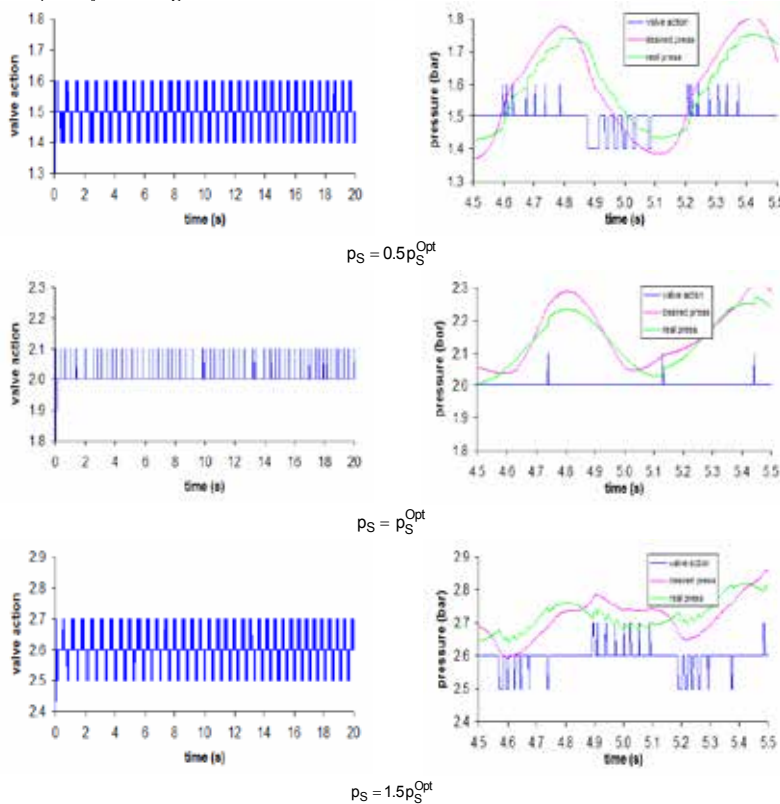


Fig. 6. Valve action and detail of the pressure changes for the 3 different settings of p_s .

4.3 Experimental Results with a single DoF Pendulum

4.3.1 Pendulum

The complete set-up of the pendulum is shown in Fig. 7. The structure is made of a high-grade aluminium alloy, AlSiMg1, and the design is exactly the same as the limbs of the robot "Lucy" (see section 5), only the connection parameters of the pull rod and lever mechanism are specific. The actual length of the swinging part of the pendulum is 45cm, and a weight of 10kg is attached at the end.

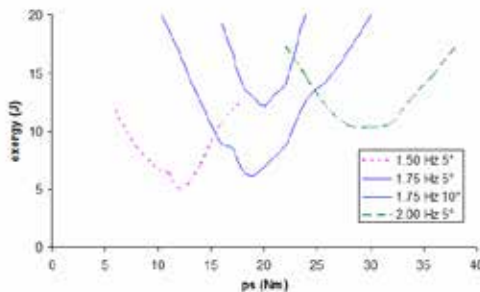
The sensors are Agilent HEDM6540 incremental encoder for reading the joint position and two pressure sensors (Honeywell CPC100AFC), mounted inside each muscle. The controller is implemented on a PC and a National Instruments (AT-MIO-16E-10) is used for data acquisition, sampling at 500Hz.



Fig. 7. CAD drawing and photograph of the pendulum setup.

4.3.2 Results and Discussion

Fig. 8 shows the total experimentally determined exergy as a function of the parameter p_s for pure sinusoidal trajectories with different frequencies and amplitudes. The graphs show that an optimal p_s value exist for each frequency and, as expected, this optimal value increases with the frequencies.



	$P_{S_{exp}}^{Opt}$ (Nm)	$P_{S_{calc}}^{Opt}$ (Nm)
1.50 Hz	12	12.6
1.75 Hz	19	21.4
2.00 Hz	30	31.7

Fig. 8. (Left) Total output exergy vs stiffness parameter p_s for different frequencies and amplitudes; (Right) Optimal p_s values of the graph compared with calculated ones.

On the right of Fig. 8, both the experimentally determined ($P_{S_{exp}}^{Opt}$) and calculated ($P_{S_{calc}}^{Opt}$; equation (15)) optimal values of p_s for the three different swing frequencies of the graph on the left of Fig. 8, are given. The calculated optimal p_s gives a good approximation of the compliance parameter that is required for minimal energy consumption in the experiment. The graph of Fig. 8 additionally shows the total exergy as a function of the parameter p_s for two amplitudes of 5° and 10° at 1.75Hz. The optimal stiffness stays nearly the same as we would expect for a pendulum, but energy consumption is higher for larger amplitudes. The actual passive trajectory of the pendulum deviates from a pure sine-wave. This deviation increases for larger amplitudes. Consequently, more valve switching is needed for larger movements, which induces higher energy consumption.

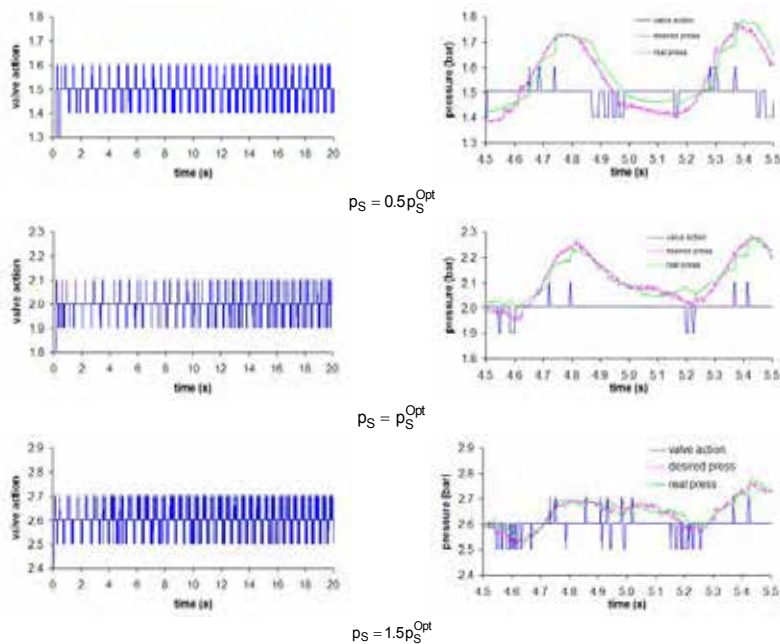


Fig. 9. Experimentally determined valve action and detail of the pressure changes for 3 different settings of p_s .

To test the effectiveness of the proposed compliance controller the same experiment with the chirp function trajectory going from 1.5Hz to 2.0Hz, as in simulation (section 4.2.2), is performed on the real pendulum. Graphs depicting valve action and detailed pressure courses are given in Fig. 9. One can see that the results are comparable to the simulations shown in Fig. 6. The fact that there are more valve actions in reality than in simulation can be explained by some leakages in the muscles and deviations due to parameter estimation errors. The desired pressure slopes are not as smooth as in simulation, this is mainly due to the noise on the input signal, most of it coming from the velocity part of the PID controller. For more complex trajectories than a sine function the consumed energy will increase, because the imposed

trajectory differs more with respect to the natural movement of the pendulum, which is a sine function for small angles. The study of the sine wave is however of great importance for bipedal locomotion. The energy transfer mechanism used in walking is often referred to as an "inverted pendulum mechanism". This observation has led to the development of the so called "passive walkers" where walking is a succession of swing motions.

5. Bipedal Robot "Lucy"

5.1 Robot Hardware

The Multibody Mechanics Research Group of the Vrije Universiteit Brussel has built the planar walking biped "Lucy". The goal of the project is to achieve a lightweight bipedal robot able to walk in a dynamically stable way, while exploiting the passive behaviour of the PPAM's in order to reduce energy consumption and control efforts. In primary instance the compressor or supply tank and PC are placed outside the robot for simplicity reasons. Building an autonomous robot is not the major issue, the main focus of the research is to investigate joint compliance variation for bipedal walking mechanisms. The muscles are however strong enough to carry an additional payload.

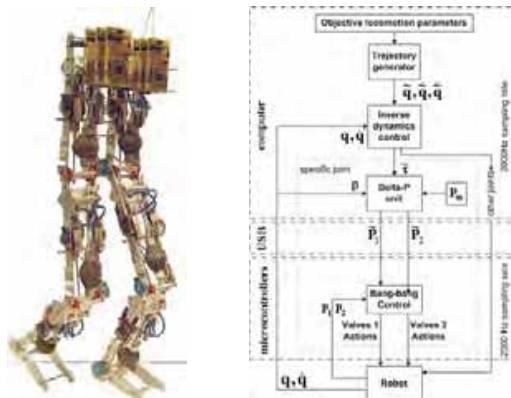


Fig. 10. (left) The robot "Lucy" ; (right) Control scheme for the complete robot.

The left of Fig. 10 shows the robot "Lucy" which weighs about 30kg and is 150cm tall. The motion of "Lucy" is restricted to the sagittal plane in order to avoid unnecessary complexity regarding control and design. Moreover, it has been shown that for bipedal walking the dynamical effects in the lateral plane have a marginal influence on the dynamics in the sagittal plane. Key elements in the design phase were modularity and flexibility to be able to make changes to the robot configuration during the experimental process. This resulted in nearly the same mechanical and electronic configuration for each structural element such as lower leg, upper leg and body. These modular units are the same as the ones used for the pendulum structure. Six of these units are connected as depicted in Fig. 10. The same sensors, valve system and local microcontroller are provided for each unit. An additional micro-controller is used to read ground reaction force sensors, absolute position of the body and compressed air consumption. The high-level control is implemented on a PC. All the micro-controllers communicate with this central, Windows operated PC by a USB 2.0 serial

bus. As such, the complete biped is controlled at a sample rate of 2000Hz. The timing of the communication refresh rate is controlled by the USB Cypress micro-controller. One should also remark in the context of this refresh rate, that the delay time of the valves is about 1ms, which suggests that the communication frequency of 2000Hz is high enough. More information on the hardware of "Lucy" can be found in (Verrelst et al., 2005c).

5.2 Robot Control

The considered controller is given in the schematic overview on the right of Fig. 10 and is a combination of a global trajectory generator and a joint trajectory tracking controller. The trajectory generation is designed to generate walking movements based on objective locomotion parameters chosen for a specific robot step, which are average forward speed, step length, step height and intermediate foot lift. These parameters are calculated by a high level path planning control unit, which is beyond the scope of the current research. The trajectories of the leg links are represented by polynomials and are calculated by a simplified version of the method developed by Vermeulen (Vermeulen et al., 2005). The upper body and hip velocity is held constant in this case, which is valid for low walking speeds. In the future, the full method developed by Vermeulen will be used to achieve faster walking. The trajectories of the leg links, represented by polynomials, are planned in such a way that the upper body motion is naturally steered, meaning that in theory no ankle torque would be required. To overcome possible external disturbances, a polynomial reference trajectory is established for the upper body motion, which mimics a natural trajectory. Consequently the required ankle torque is low, meaning that it does not cause the ZMP (Zero Moment Point) to move out of the stability region. Thus, taking the motion of the upper body into account and not keeping it at a fixed angle as is the case in this paper. The effectiveness of this method applied to the robot "Lucy" has been proven in simulation (Verrelst et al., 2006).

The joint trajectory tracking controller is divided into three parts, analogously as for the pendulum structure: the computed torque module, the delta-p unit and the bang-bang pressure controller. The trajectory planner, computed torque module and delta-p unit are implemented on a central computer; the bang-bang controller on the micro-controllers.

After the trajectory generator the joint drive torques are calculated. This control unit is different for the single and double support phase. During single support the torques are calculated using the popular computed torque technique consisting of a feedforward part and a PID feedback loop. The same formulation is used as in equation (8), but the matrices D_e , C_e and G_e represent the complete robot. During single support the robot has 6 DoF and the 6 computed torques can be calculated straightforward. During the double support phase, immediately after the impact of the swing leg, however, three geometrical constraints are enforced on the motion of the system. They include the step length, step height and the angular position of the foot. Due to these constraints, the robot's DoF is reduced to 3. The dynamic model has to be reformulated in terms of the reduced independent variables. The dependant variables are related to the dependant variables through the kinematical Jacobian. An actuator redundancy arises because there are 6 actuators which need to control only 3 DoF. To be able to calculate the 6 joint drive torques, a control law as proposed by Shih (Shih et al., 1992) can be used, which is based on a matrix pseudo-inverse. The disadvantage is that these results are discontinuous when switching between single and double support phase. A discussion on this implementation has been done by means of simulations (Verrelst et al., 2006). An alternative way to distribute torques over the actuators is to make a linear transition of torques between the old and new single support phase, by

calculating the applied torque as if the robot is in single support phase. The advantage of this strategy is that there are no torque discontinuities when switching between single and double support phase, but the calculated torques are not dynamical correct. However, the double support phase is rather short and a correcting feedback loop is provided. Experimental results show that this is a good strategy for the regarded motions.

The computed torque module presents all 6 required torque values to each local delta-p unit for each muscle pair of "Lucy". From this level on, the different control blocks are analogous to the one of the pendulum control structure. Of course, for each joint the specific torque generation characteristics are provided for the delta-p unit, since a hip, knee and ankle have different attachment points for the muscles. But in the current implementation the compliance strategy as used for the pendulum is not yet implemented for "Lucy", the compliance parameter is fixed at an arbitrary value.

5.3 Walking Experiments

In this section the measurements of the walking biped are shown with the following chosen objective locomotion parameters: mean forward velocity: 0.03m/s, steplength: 0.10m, stepheight: 0.0m, footlift: 0.04m. The calculated single support and double support phase durations are respectively 2.67s and 0.67s.

The graphs of Fig. 11 to Fig. 13 depict 4 double support phases and 4 single support phases. The graph at the left of Fig. 11 show the desired and real joint angles for the ankle, the knee and the hip. Vertical lines on all graphs show the phase transition instants. Due to the nature of the bang-bang pneumatic drive units and the imperfections introduced in the control loops, tracking errors can be observed. But tracking errors are not that stringent as it is for e.g. welding robots. The most important thing is that the overall dynamic robot stability is guaranteed. The graph on the right of Fig. 11 depicts the actual applied torque for the knee of the left leg, which consists of a PID feedback part and a computed torque part. The computed torque controller is working well, but the robot parameters still have to be finetuned to lower the action of the PID controller.

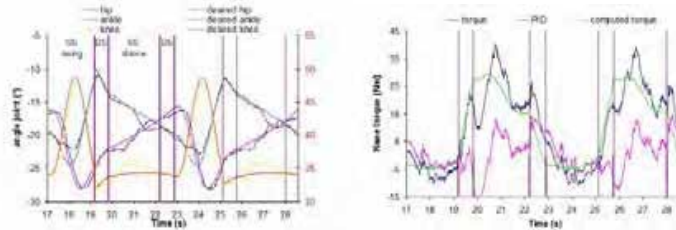


Fig. 11. (left) Desired and measured angle of left hip, knee and angle joint ; (right) Left knee torque.

The pneumatics are characterized by pressure courses in both muscles of each joint. The graphs in Fig. 12 show desired and measured gauge pressure for the front and rear muscle of the knee of the left leg. All these graphs additionally show the valve actions taken by the respective bang-bang pressure controller. Note that in these figures a muscle with closed valves is represented by a horizontal line depicted at the 2bar pressure level, while a small peak upwards represents one opened inlet valves, a small peak downwards one opened exhaust valves and the larger peaks represent two opened inlet or four opened outlet valves.

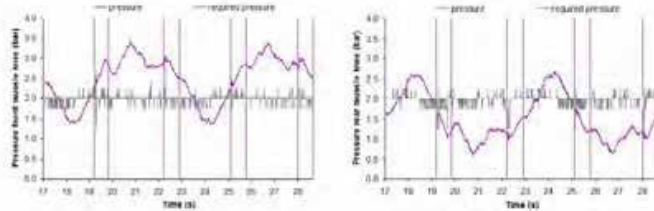


Fig. 12. Pressure and valve action of front (left) and rear (right) left knee muscle.

The desired pressures are calculated by the delta-p unit. For this experiment the mean pressure p_m (comparable to p_s) for all joints is taken at 2bar, consequently the sum of the pressures in each pair of graphs, drawing the front and rear muscle pressures, is always 4bar. It is observed that the bang-bang pressure controller is very adequate in tracking the desired pressure. Currently a lot of valve switching is required due to the fix compliance setting. By incorporating the natural dynamics of the system, as is described in the previous section, the switching will be reduced.

The two graphs in Fig. 13 depict the horizontal and vertical position of both feet as well as ZMP and CoG position.

Note that the swing foot moves twice the step length during the single support phase. Compared with the desired objective locomotion parameter, only small deviations can be observed. This proves the global performance of the proposed trajectory control strategy. The position of the ZMP is measured by four load cells of Transducer Techniques (THA-250-Q). They are attached in the front and end of the sole plate and measure vertical ground forces, allowing to calculate the actual ZMP position in combination with the estimated ankle torques. Currently, the ZMP measurement is not used in the control loop yet, but it will be in the future to achieve faster walking. The ZMP and COG are close alike because the velocity of the robot is still rather slow. So the inertial influence is rather small compared to the gravitational component of the controller. The controller has however already performed well under dynamic walking situation in simulation (Verrelst et al., 2006). Several movies of the walking motion of the biped "lucy" can be seen at the following website: <http://lucy.vub.ac.be>.

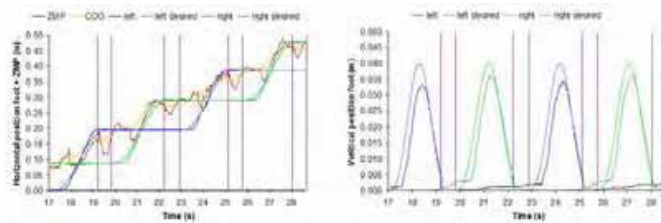


Fig. 13. (left) Horizontal position foot, Zero Moment Point (ZMP) and Centre of Gravity (COG) ; (right) Vertical position foot.

6. Conclusions

This publication discusses the importance of variable compliance for legged robots. Use of passive compliance in the actuation reduces reflected inertia which primarily is interesting

vis-à-vis safe human/robot interaction. But the variable compliance can also be used to change the natural dynamics of a system, which in turn can be exploited to reduce energy consumption of legged robots. Doing so, an interesting combination can be created between very energy efficient “passive walkers” and fully (stiffly) actuated versatile robots.

In the current state of our research the compliant actuation is provided with an antagonistic setup of pleated pneumatic artificial muscles. These specific muscles have been created to overcome some shortcomings of the more popular McKibben Artificial muscles. By regulating the both pressure in an antagonistic setup of two artificial muscles, as well as compliance as generated torque can be regulated. A description of the complete low-level tracking control is given, which includes a computed torque controller, a delta-p pressure calculation module, a bang-bang pressure controller and additionally a compliance controller. Most of these control blocks uses feedforward model based calculations.

The main purpose of the publication is to show that proper compliance control decreases energy consumption, which has been proven by simulation and experiments on a one DoF pendulum structure. The continuous compliance setting is applied for tracking a sine wave with varying frequency (shirp function), ranging from 1.5 to 2.0Hz. A comparison between proper and wrong compliance setting has been made with respect to estimated exergy consumption, which showed energy saving of more than six times, while the tracking output of the system is preserved. This clearly indicates the importance of variable compliance used for repetitive motion.

Finally, this repetitive motion is found in the movement of legged robots, which can be seen as an inverted pendulum motion for a stance leg and regular pendulum motion for a swing leg. As such the idea is to use the same compliance control of the pendulum structure for a bipedal walking robot. For this purpose the planar walking biped “Lucy”, actuated by six pairs of artificial muscles, has been built. A short overview of the hardware and overall control system is given together with some basic walking experimental results. These results only discuss the tracking performance of the compliant actuation system and do not yet incorporate the proposed compliance control, which will be implemented in the near future. It is expected that the latter control structure substantially reduces energy consumption.

However, it has to be mentioned that the use of pneumatic energy in general is not energy efficient in an absolute sense. Thus, one should keep in mind that pneumatic energy for a mobile robot might not be the best solution, even if exploitation of natural dynamics is incorporated. Our research currently focuses on showing the importance of variable compliance for legged robots, in fact to show the relative decrease of energy consumption by proper compliance control. And, the current choice with the use of pneumatic artificial muscles provides a convenient way to implement this variable compliance. Parallel research is being conducted to develop alternative variable compliant actuation principles, by means of combining electrical actuation with standard passive spring elements. However, the developed trajectory tracking strategies, controlling the antagonistic setup with two artificial muscles, will be used for a compliant robotic exoskeleton to assist in the rehabilitation of paraplegic patients at the lower extremities. For such application energy consumption is not that stringent since the power source is used externally, but the compliant interface and the high power to weight ratio of the artificial muscles are extremely beneficial.

7. Bibliography

- Asano, F., Yamkita, M. and Furuta, K. (2000). Virtual passive dynamic walking and energy-based control laws, *Proceedings of the IEEE/RSJ Int Conf on Intelligent Robots and Systems*, pp. 1149-1154.

- Davis, S. T. and Caldwell, D. G. (2001). The bio-mimetic design of a robot primate using pneumatic artificial muscle actuators, *Proc 4th Int Conf on Climbing and Walking Robots*, Karlsruhe, Germany, pp. 197-204.
- Bicchi, A. and Tonietti, G. (2004). Fast and soft arm tactics: Dealing with the safety-performance trade-off in robot arms design and control, in *IEEE Robotics and Automation Magazine*, vol. 11, pp. 22-33.
- Daerden, F. (1999). *Conception and Realization of Pleated Pneumatic Artificial Muscles and their Use as Compliant Actuation Elements*, PhD thesis, Vrije Universiteit Brussel.
- Daerden, F. and Lefeber, D. (2001). The concept and design of pleated pneumatic artificial muscles., *International Journal of Fluid Power*, vol. 2, no. 3, pp. 41-50.
- Garcia, M., Chatterjee, A., Ruina, A. and Coleman, M. (1998). The simplest walking model: Stability, complexity, and scaling, *ASME Journal of Biomechanical Engineering* 120: 281-288.
- Gregorio, P., Ahmadi, M. and Buehler, M. (1997). Design, control, and energetics of an electrically actuated legged robot, *IEEE Transactions on Systems, Man and Cybernetics* 27B(4): 626-634.
- Hollander, K.W. and Sugar, T.G. (2004). *Concepts for compliant actuation in wearable robotic systems*, US-Korea Conference, North Carolina, August.
- Hurst, J. W., Chestnutt, J. and Rizzi, A. (2004). An actuator with physically variable stiffness for highly dynamic legged locomotion, new orleans, usa, pp. .," *Proc of the IEEE Int Conf on Robotics and Automation*, pp. 4662-4667.
- International Standard ISO6358 (1989). Pneumatic fluid power - method of test - determination of flow rate characteristics of components using compressible fluids.
- Lee, T.-T. and Liao, J.-H. (1988). Trajectory planning and control of a 3-link biped robot, *Proc. IEEE Int. Conf. on Robotics and Automation*, New York, USA, pp. 820-823.
- Matsuoka, K. (1980). A mechanical model of repetitive hopping movements, *Biomechanisms* 5: 251-258.
- McGeer, T. (1990). Passive dynamic walking, *Int. Journal of Robotics Research* 9(2): 62-82.
- Migliore, S. A., Brown, E. A. and DeWeerth, S. P. (2005). Biologically inspired joint stiffness control, *Proc IEEE Int Conf on Robotics and Automation, Barcelona, Spain, April*, pp. 4519-4524.
- Morita, T. and Sugano, S. (1995). Design and Development of a new Robot Joint using Mechanical Impedance Adjuster, *Proc IEEE Int Conf on Robotics and Automation, Nagoya, Japan, April*, pp. 2469-2475.
- Okada, M., Nakamura, Y. and Ban, S (2001). Design of Programmable Passive Compliance Shoulder Mechanism, *Proc IEEE Int Conf on Robotics and Automation, Seoul, Korea, May*, pp. 348-353.
- Osuka, K. and Saruta, Y. (2000). Development and control of new legged robot Quartet III - from active walking to passive walking, *Proc IEEE/RSJ Intl Conf on Intelligent Robots and Systems*, pp. 991-995.
- Pratt, G. A. and Williamson, M. M. (1995). Series elastic actuators, *Proc IEEE-IROS Conference, Pittsburg, USA*, pp. 399-406.
- Pratt, J. E. and Pratt, G. (1998). Exploiting natural dynamics in the control of a planar bipedal walking robot, *Proc of the 36th Annual Allerton Conf on Communication, Control, and Computing*, Monticello, Illinois.
- Raibert, M. (1986). *Legged Robots That Balance*, MIT Press, Cambridge, Massachusetts.
- Rogers, G. and Mayhew, Y. (1992). *Engineering Thermodynamics, Work and Heat Transfer*, John Wiley & Sons, New York, USA.

- Sulzer, J., Peshkin, M. and Patton, J. (2005). Marionet: An exotendon-driven, rotary series elastic actuator for exerting joint torque, *Proc Int Conf on Robotics for Rehabilitation, Chicago, USA*.
- Schulte, H. F. (1961). The characteristics of the McKibben artificial muscle, *The Application of External Power in Prosthetics and Orthotics*, number Publication 874, National Academy of Sciences National Research Council, Lake Arrowhead, pp. 94-115.
- Shih, C.-L. and Gruver, W. (1992). Control of a biped robot in the double-support phase, *IEEE Transactions on Systems, Man and Cybernetics* 22(4): 729-735.
- Takanishi, A., Ishida, M., Yamazaki, Y. and Kato, I. (1985). The realization of dynamic walking by the biped walking robot WL-10RD, *Proc. Int. Conf. on Advanced Robotics*, pp. 459-466.
- Thompson, C. and Raibert, M. (1989). Passive dynamic running, *Proceedings of the International Symposium of Experimental Robotics*, pp. 74-83.
- M. Van Damme, R. Van Ham, B. Vanderborght, F. Daerden and D. Lefeber (2005). Design and Control of a "Soft" 2-DOF Planar Pneumatic Manipulator, *Proc. Int. Symp. on Measurement and Control in Robotics, Brussels, Belgium*.
- van der Linde, R. Q. (1999). Design, analysis and control of a low power joint for walking robots, by phasic activation of mckibben muscles., *IEEE Transactions on Robotics and Automation* 15(4): 599-604.
- Van Ham R., Vanderborght B., Van Damme M., Verrelst B., and Lefeber D (2005). MACCEPA: the Actuator with adaptable compliance for dynamic walking bipeds. *Proc 8th Int Conf on Climbing and Walking Robots, London, U.K., September*, pp. 759-766.
- Van Ham, R., Verrelst, B., Daerden, F., Vanderborght, B., and Lefeber, D. (2005). Fast and accurate pressure control using on-off valves, *International Journal of Fluid Power*, vol. 6, March, pp. 53-58.
- Vermeulen J., Verrelst B., Lefeber D., Kool P. & Vanderborght B. (2005). A real-time joint trajectory planner for dynamic walking bipeds in the sagittal plane, *Robotica*, Vol. 23, Issue 06, november, pp 669-680.
- Verrelst, B., Van Ham, R., Vanderborght, B., Vermeulen, J., Lefeber, D. and Daerden, F. (2005a). Exploiting adaptable passive behaviour to influence natural dynamics applied to legged robots, *Robotica*, vol. 23, no. 2, pp. 149-158.
- Verrelst B., Vanderborght B., Vermeulen J., Van Ham R., Naudet J. & Lefeber D. (2005b). Control Architecture for the Pneumatically Actuated Dynamic Walking Biped "Lucy", *Mechatronics*, Volume 15, Issue 6, July, pp. 703-729.
- Verrelst B., Van Ham R., Vanderborght B., Daerden F. & Lefeber D. (2005c). The Pneumatic Biped "LUCY" Actuated with Pleated Pneumatic Artificial Muscles, *Autonomous Robots*, vol.18, pp.201-213.
- Verrelst B., Vermeulen J., Vanderborght B., Van Ham R., Naudet J., Lefeber D., Daerden F. & Van Damme M. (2006), Motion Generation and Control for the Pneumatic Biped "Lucy", *International Journal of Humanoid Robotics*, Vol. 3, No. 1, pp. 1-35.
- Wisse, M. (2001). Biologically inspired biped design, *Symposium of the Int. Soc. for Postural and Gait Research*, Maastricht, The Netherlands, pp. 900-903.
- Wisse, M., Schwab, A. L. and vd. Linde, R. Q. (2001). A 3d passive dynamic biped with yaw and roll compensation, *Robotica* 19: 275-284.
- Yamgusch, J., Nishino, D. and Takanishi, A. (1998). Realization of dynamic biped walking varying joint stiffness using antagonistic driven joints, *Proc IEEE Int Conf on Robotics and Automation*, Leuven, Belgium, pp. 2022-2029.

Acquisition of Obstacle Avoidance Actions with Free-Gait for Quadruped Robots

Tomohiro Yamaguchi*, Keigo Watanabe** and Kiyotaka Izumi**

** Department of Electrical, Electronics and Information Engineering,
Kanagawa University*

*** Department of Advanced Systems Control Engineering, Saga University
Japan*

1. Introduction

Although legged mobile robots are inferior to wheeled or crawler types in mobility efficiency on the flat ground, they demonstrate high motion performance and adaptation capability to the ground by utilizing their high degrees of freedom (DOF). Since such robots choose stable leg-placement, stable movements can be performed on irregular terrains (Şafak & Adams, 2002). Moreover, they demonstrate some unique functionalities, e.g., the scaffold of a stable activity at the time of rest can be formulated from taking the posture of legs which are hard to topple (Hirose & Yoneda, 1993).

However, it is very difficult to decide robot gait due to its high DOF. When the legs of the robot are simply controlled by a fixed command, adaptation capability to the terrain is remarkably restricted and sometimes it is impossible to maintain a stable walk. Moreover, when a leg is unable to be placed properly an optimum leg placement must be efficiently found from among other candidates.

Therefore, it needs for a legged mobile robot to sequentially decide the progression of legs. For that purpose, the robot predictively perceives and recognizes geographical features of the terrain, and it consequently gets over any obstacle by using an adaptive ability acquired in advance. From this fact, legged robots are not necessary to avoid all the obstacles by altering their path, unlike wheeled or crawler types, because they can avoid an obstacle by crawling-over or striding, according to the obstacle's nature and the current state of the robot. Thus, it can be found that the mobility efficiency to reach a destination is improved by such action. Moreover, when robots have many legs like 4-legged or 6-legged types, the movement range is affected by the order of swing leg.

We studied path to a destination and obstacle avoidance of a quadruped robot considering free-gait. In general, quadruped robots can realize two types of walk: one is the static walk which retains the stability statically, keeping the center of gravity (COG) in the polygon constructed by the support legs, and the other is the dynamic walk which retains dynamic stability, though it is statically unstable. In the static walk, one feature is that an irregular-terrain walk is easily realized, whereas excelling in walking-speed or consumption energy is a feature of the dynamic walk (Kimura et al., 1990, Kimura, 1993). Thus, static walk is suitable for action acquisition of quadruped robots in geographical environments where obstacles, such as a level difference, exist (Chen et al., 2002, Chen et al., 2002).

In this research, since obstacle avoidance is taken into consideration, the static walk is adopted as a basic walk and the order of swing leg is determined. A free gait planning in the static walk was already formulated as a conditional optimization problem and the solution method by the Monte Carlo method was proposed by Nakamura et al. (Nakamura et al., 1999). However, assumed, fixed environment specializes the result obtained from this method; it has no flexibilities to the outside of search environment.

Dimensions of the obstacle and the current state of the robot can be perceived accurately, because of the presence of force sensors, ultrasonic sensors and potentiometers on the present quadruped robot. A simulator is built based on the robot's structure and the path to the destination is acquired from the simulator. When avoiding an obstacle by quadruped robot, there are many combinatorial solutions, such as combinations of turn and forward movements, combinations of sideway and forward movements, etc. The robot's body height must be regulated because leg movement is restricted by height. When the leg is placed on a corner of an obstacle, the robot may fall, so robot action must be determined from the relative position between the obstacle and the robot. We propose a method for determining the action of a quadruped robot using neural network (NN) from the position of the destination, the obstacle configuration, and the robot's self-state. Note, however, that no any free-gait motion is taken into consideration at the first research. The order of swing leg in free-gait is determined in the second research using the amount of movements and the robot's self-state. The static walk of the quadruped robot has 24 kinds of the order of swing leg. Since the static walk always needs to set the COG of the robot in the polygon constructed by the supporting legs, the amount of movements of the body is different, depending on the order of swing leg. Therefore, the order of swing leg is determined by another NN. Furthermore, an NN for determining the robot action is acquired by re-learning the NN that was built in the case when the order of swing leg was fixed. To reach a destination with a minimum number of walking cycles (Furusho, 1993), NN design parameters are optimized by a genetic algorithm (GA) using data from several environments, in which each environment has different destinations and obstacle dimensions.

2. Quadruped Robot

Fig. 1 shows the experimental setup. TITAN-VIII (Hirose & Arikawa 1999) (see Fig. 2) is the quadruped robot. TITAN-VIII has four legs, one with three DOF, and each joint has a potentiometer. Force sensing resistors (Interlink Electronics, FSR Part # 402) are used on the leg sole to measure force exerted on each leg. Ultrasonic sensors (NiceRa, T/R40-16) are used on the forelegs to detect an obstacle; each foreleg has three ultrasonic sensors.

Potentiometer measurement and force sensing resistor are transferred to a personal computer through a robot interface board (Fujitsu, RIF-01) and an A/D converter board (Interface Corporation, PCI-3133). The ultrasonic sensor measures the time difference between emittance and reception of ultrasonic waves, which is reflected on an obstacle by universal pulse processor (UPP), part of RIF-01. The computer sends joint angle commands to a motor driver (Okazaki Sangyo Co. Ltd., Titech Motor Driver) on the robot through the robot interface board. Since sensor information in feedback control must be processed in real time, RT-Linux is used as the computer OS.

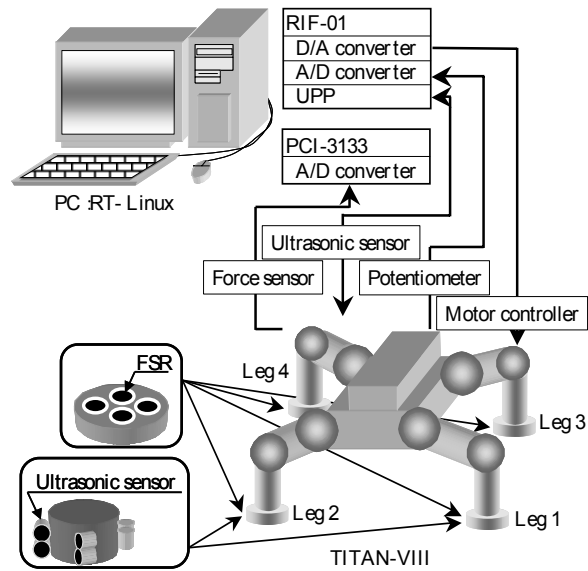


Fig. 1. Robot control system.

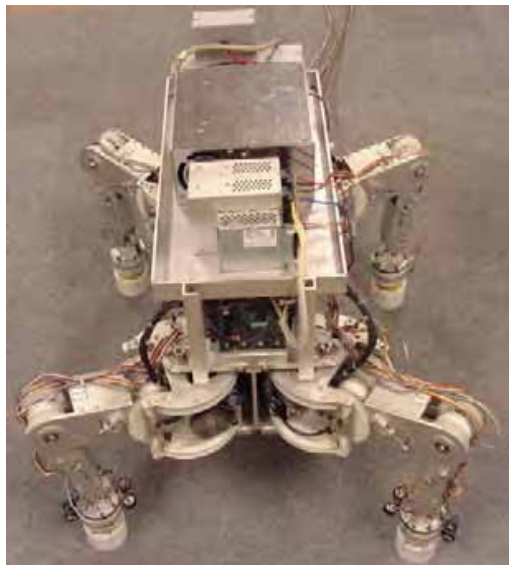


Fig. 2. TITAN-VIII.

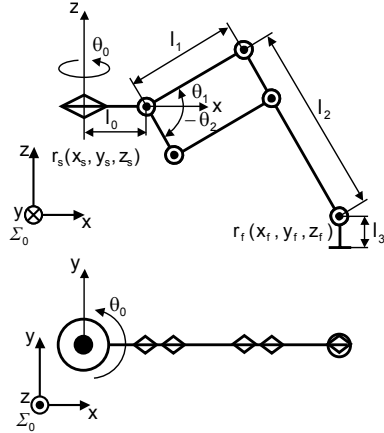


Fig. 3. Coordinate system of TITAN-VIII's leg.

The coordinate system of one leg of the robot is defined in Fig. 3. For any leg, assuming position coordinates of a shoulder are $r_{si}(x_{si}, y_{si}, z_{si})$ and position coordinates of a sole are $r_{fi}(x_{fi}, y_{fi}, z_{fi})$, joint angles, $\theta(\theta_{0i}, \theta_{1i}, \theta_{2i})$ are obtained by

$$\begin{bmatrix} \theta_{0i} \\ \theta_{1i} \\ \theta_{2i} \end{bmatrix} = \begin{bmatrix} \tan^{-1} \left(\frac{y_{fi} - y_{si}}{x_{fi} - x_{si}} \right) \\ \tan^{-1} \left(\frac{d_{1i}}{\pm \sqrt{r_1^2 - d_{1i}^2}} \right) - \phi_i \\ \tan^{-1} \left(\frac{d_{2i}}{\pm \sqrt{r_1^2 - d_{2i}^2}} \right) - \phi_i \end{bmatrix} \quad (1)$$

where

$$r_i = \sqrt{\left(\frac{x_{fi} - x_{si}}{\cos \theta_{0i}} - l_0 \right)^2 + z_i^2} \quad (2)$$

$$\phi_i = \tan^{-1} \frac{1}{z_i} \left(\frac{x_{fi} - x_{si}}{\cos \theta_{0i}} - l_0 \right) \quad (3)$$

$$d_{1i} = \frac{1}{2l_1} \left\{ \left(\frac{x_{fi} - x_{si}}{\cos \theta_{0i}} - l_0 \right)^2 + z_i^2 + l_1^2 - l_2^2 \right\} \quad (4)$$

$$d_{2i} = \frac{1}{2l_2} \left\{ \left(\frac{x_{fi} - x_{si}}{\cos \theta_{0i}} - l_0 \right)^2 + z_i^2 + l_1^2 + l_2^2 \right\} \quad (5)$$

$$Z_i = Z_{fi} - Z_{si} \quad (6)$$

Here, i denotes the leg number ($i=1, 2, 3, 4$). With reference to, l_0 , l_1 , and l_2 are lengths of links. The ultrasonic sensor and the force sensor are on the foot at l_3 , where $l_3 = 130$ [mm].

Given the initial positions for soles and shoulders, joint angles of legs including the swing leg are obtained from Eq. (1). Note, however, that the operation of each joint is $90.0 \leq \theta_{01} \leq 245.0$, $-65.0 \leq \theta_{02} \leq 90.0$, $115.0 \leq \theta_{03} \leq 270.0$, $65.0 \leq \theta_{04} \leq -90.0$, $-65.0 \leq \theta_{1i} \leq 65.0$ and $-65.0 \leq \theta_{2i} \leq 90.0$ in degrees. Actions should be determined to not exceeding operational range.

3. Obstacle Detection and Recognition

To avoid an obstacle, its existence and height must be recognized using sensory information. For the robot, obstacles are detected and recognized by ultrasonic sensors. Ultrasonic sensors detect obstacles perpendicular to signals emitted by them. If the ultrasonic sensor was on the robot, the measurement of sensors would be shortened, so the ultrasonic sensors are on legs so that the emission of ultrasonic waves is changeable. As a result, obstacles not perpendicular to the robot's forward direction can be detected. When a leg is swinging, it can move up to a position where the ultrasonic sensor does not detect anything. Using this concept, the height of the obstacle is measured roughly.

Ultrasonic sensors on forelegs detect obstacles in the robot's forward direction. A downward sensor detects the unevenness of terrain. When the center of the robot's body is set as the origin, the position of an obstacle, (x_{ob}, y_{ob}) , detected by the robot's front sensors is given by

$$x_{ob} = x_{si} + (l_0 + l_1 \cos \theta_{1i} + l_2 \sin \theta_{2i}) \cos \theta_{0i} - L_{so} \sin \theta_{0i} \quad (7)$$

$$y_{ob} = y_{si} + (l_0 + l_1 \cos \theta_{1i} + l_2 \sin \theta_{2i}) \sin \theta_{0i} + L_{so} \cos \theta_{0i} \quad (8)$$

The position of an obstacle detected by the robot's left and right sensors is given by

$$x_{ob} = x_{si} + (l_0 + l_1 \cos \theta_{1i} + l_2 \sin \theta_{2i}) \cos \theta_{0i} + L_{so} \cos \theta_{0i} \quad (9)$$

$$y_{ob} = y_{si} + (l_0 + l_1 \cos \theta_{1i} + l_2 \sin \theta_{2i}) \sin \theta_{0i} + L_{so} \sin \theta_{0i} \quad (10)$$

Here, i denotes the front leg numbers ($i=1, 2$). The position of the detected obstacle is calculated with the distance L_{so} measured by ultrasonic sensor, together with the body position and the joint angles of the foreleg.

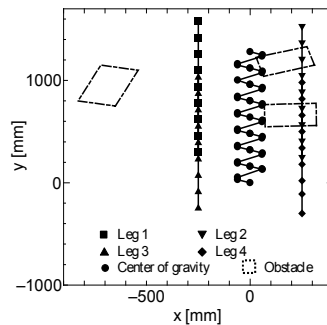


Fig. 4. Movement path of quadruped robot.

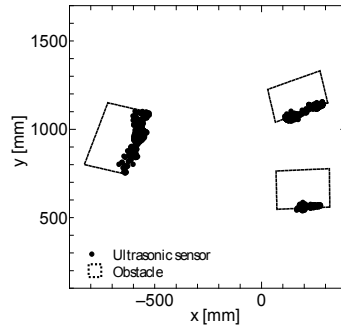


Fig. 5. Recognition result of the obstacles in the forward direction.

Positions of obstacles, detected where the robot moves (Fig. 4) are given in Figs. 5 and 6. Fig. 5 shows information regarding obstacles from foreleg sensors. We found that one side of each obstacle is detected. Fig. 6 shows positions and heights of obstacles when a leg passed over them, detected by the downward sensor. Robot action must be determined using such information.

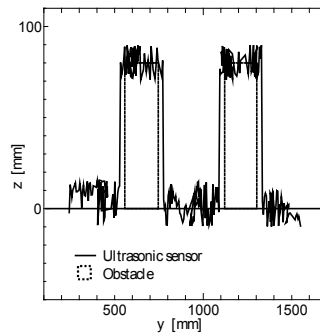


Fig. 6. Recognition result of the ground.

4. Action Determination of Quadruped Robot Using an NN

For an actual robot, information obtained from sensors includes leg joint angles and dimensions and heights of obstacles. We assumed that destination information is given in advance. Actions of the quadruped robot are determined by the above information. For each action, there are several combinatorial solutions such as the combination of forward- and turning-motions, one of forward- and sideway-motions, etc.

The robot action is decided by a three-layered NN (Fig. 7). This NN is trained offline so as to achieve an action using a minimum number of walking cycles.

Inputs to the NN are assumed to be the position of each sole $\{x_{j1}(k), y_{j1}(k), \dots, \{x_{j4}(k), y_{j4}(k)\}$, the robot's body height $Zr(k)$, x-directional maximum and minimum distances to an obstacle at right $\{x_{or\max}(k), x_{or\min}(k)\}$, the y-directional maximum and minimum distances to the obstacle at right $\{y_{or\max}(k), y_{or\min}(k)\}$, and the height of the obstacle at right $\{z_{or\max}(k), z_{or\min}(k)\}$, the x-directional maximum and minimum distances to the obstacle at left $\{x_{ol\max}(k), x_{ol\min}(k)\}$, the y-

directional maximum and minimum distances to the obstacle at left $\{y_{olmax}(k), y_{olmin}(k)\}$, and the height of the obstacle at left $\{z_{olmax}(k), z_{olmin}(k)\}$; x-directional distance error, i.e., distance between the COG of the robot and destination $x_{de}(k)$, y-directional distance error $y_{de}(k)$, and direction error, i.e., the direction between the destination and forward direction of the robot $\theta_{de}(k)$. Positions of each sole and obstacles are defined in the frame whose origin is fixed to the body center. Outputs of the NN are the amount of x- and y-directional movement of the robot $\{\Delta Xr(k), \Delta Yr(k)\}$ and the turning angle of the robot $\Delta\theta r(k)$. If no obstacles exist in front of the robot, measured values of obstacles are set to $(x_{ormax}, y_{ormax}, z_{ormax})=(1.0, -0.99, 0.0)$, $(x_{ormin}, y_{ormin}, z_{ormin})=(0.99, -1.0, 0.0)$, $(x_{olmax}, y_{olmax}, z_{olmax})=(-0.99, -0.99, 0.0)$, and $(x_{olmin}, y_{olmin}, z_{olmin})=(-1.0, -1.0, 0.0)$ [m], assuming the obstacle is behind the robot.

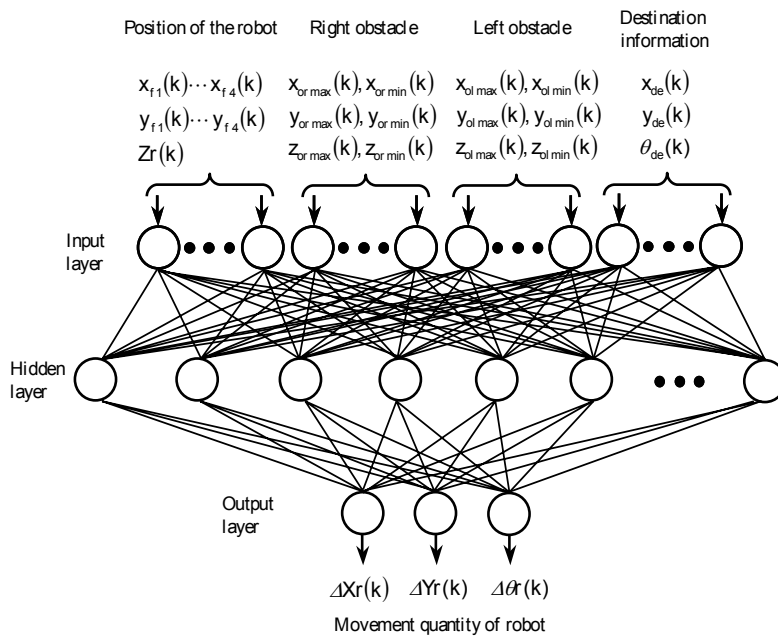


Fig. 7. Three layered neural network for determining the action of the quadruped robot.

A radial basis function neural network (RBFNN) (Elanayar & Shin, 1994, Sakawa & Tanaka, 1999), known as an NN that can realize various approximation functions, is used in the control system. With an RBFNN, a nonlinear function is expanded by any basis function having a circular contour, and is used as function approximation or pattern recognition. Unit functions at the hidden (or intermediate) layer of RBFNNs are given by

$$\phi_i(x) = \exp\left\{-\frac{\|\mathbf{X}(k) - \mathbf{c}_i\|^2}{\sigma_i^2}\right\} \quad (11)$$

where ϕ_i denotes i th unit output at the hidden layer, and design parameters of RBF are center \mathbf{c}_i and standard deviation σ_i for each input. j th unit output at output layer o_j is given by

$$o_j(k) = \sum_{i=1}^m \omega_{ij} \phi_i(\mathbf{X}) \quad (12)$$

which is calculated by a linear combination of outputs of the hidden layer. Here, ω_{ij} denotes the connection weight between the i th hidden unit and the j th output unit, and m denotes the number of units at the hidden layer, where the number of hidden units is determined by trial and error. The number of hidden units was set to $m=100$, because a good result was obtained when m was four times the number of inputs.

Using this RBFNN, an action of the quadruped robot is determined from the information of the obstacles, the current state of the robot, and the destination information.

4.1 Acquisition of Obstacle Avoidance by Simulation

A block diagram of obstacle avoidance control system is shown in Fig. 8. To avoid obstacles, the system responds to the environment by altering the path and getting over or striding obstacles. To do so, the robot changes its height to that of an obstacle. Such actions constrain how much the robot can adjust the height. The RBFNN determines the movement of the robot using obstacle dimensions, the position of each leg, and robot height, collectively considered during walking. Positions of shoulders and legs are computed from the amount of movement. Each joint angle is calculated by Eq. (1), and the output is communicated to the robot.

Distance error is updated by change in the current COG position of the robot, after computing the COG position by considering positions of shoulders and legs. Although the position of an obstacle is measured by ultrasonic sensors of the robot, position information of obstacles in simulation is updated by change of the COG position of the robot. Each leg is changed based on the amount of robot movement. The range of leg placement is assumed as follows: when placing the leg on the ground, its range should be 50 [mm] away from a corner of an obstacle, whereas, when placing the leg on an obstacle, its range should be 30 [mm] away from a corner. If a leg cannot be placed in a position, the robot places it at the nearest position from the scheduled position.

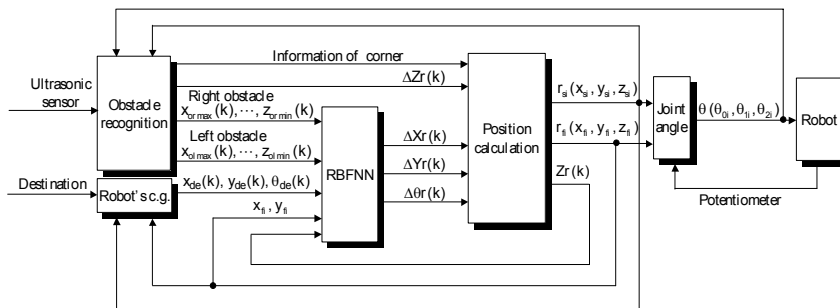


Fig. 8. Obstacle avoidance control system with consideration to the destination.

4.1.1 Simulation condition

The simulation environment is shown in Fig. 9. The y-axis is set to the forward direction of the robot. The robot is assumed to start from point (0.0, 0.0, 0.3) [m] and approach the goal, a circle having radius 0.2 [m], centered on the destination point.

Simulation is performed three times by setting distance error to $(x_{de}, y_{de})=(0.0, 1.8), (0.15, 2.2), (-0.15, 2.2)$ [m]. We assume that one obstacle exists at the robot's right and the other at the left. Coordinates of legs and the obstacle are shown in Fig. 10. Initial positions for legs are set to $(x_{f1}, y_{f1})=(-0.25, 0.3), (x_{f2}, y_{f2})=(0.25, 0.25), (x_{f3}, y_{f3})=(-0.25, -0.25)$, and $(x_{f4}, y_{f4})=(0.25, -0.3)$ [m]. Positions of x- and y- coordinates of obstacles are shown in Table 1, where each row represents coordinate data combined to produce coordinates of any two obstacles. Height coordinates z_{or} and z_{ol} of obstacles are set to 0.06, 0.12, 0.3 [m], where the data can be combined to produce a combination of heights. Coordinate $z_{or}(z_{ol})=0.3$ [m] implies that it is an obstacle the robot cannot get over, so that the combination of $z_{or}=0.3$ [m] and $z_{ol}=0.3$ [m] is not considered in simulation. One of the two obstacles is assumed to be the obstacle that the robot cannot get over, so that simulation is conducted for 96 obstacles.

Walking is a crawl in which the body is supported by three or more legs. The order of swing leg selection is the right hind-leg, right foreleg, left hind-leg, and left foreleg. Realization of stable static walking is similar to the crawl. The COG of the robot should be in the polygon constructed by the supportive legs; therefore the body is moved so that the COG of the robot is always in the supportive leg polygon. We assume that the robot is parallel to the ground.

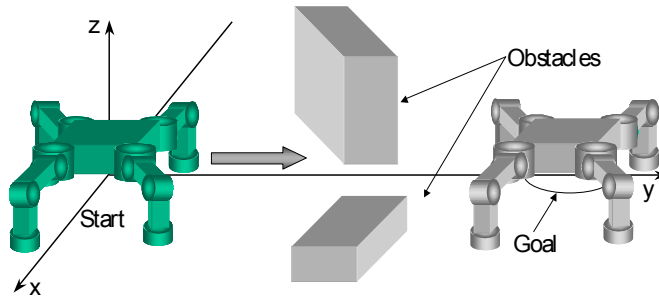


Fig. 9. The environmental setup for the acquisition of quadruped robot's action.

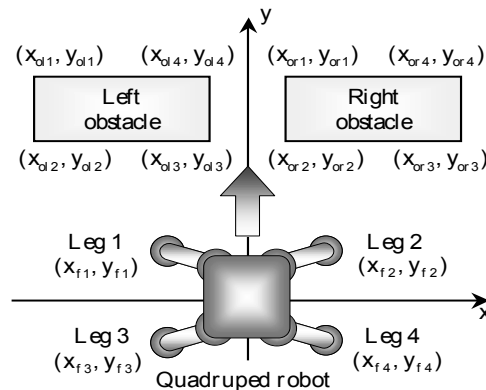


Fig. 10. Coordinates of the robot's legs and the obstacles.

No	x_{or1}	x_{or2}	x_{or3}	x_{or4}	x_{ol1}	x_{ol2}	x_{ol3}	x_{ol4}
1	0.5	0.1	0.1	0.5	- 0.5	- 0.1	- 0.1	- 0.5
2	0.5	0.1	0.1	0.5	- 0.5	- 0.2	- 0.2	- 0.5
3	0.5	0.1	0.1	0.5	-0.5	- 0.1	- 0.1	- 0.5
4	0.5	0.1	0.1	0.5	- 0.5	- 0.2	- 0.2	- 0.5
No	y_{or1}	y_{or2}	y_{or3}	y_{or4}	y_{ol1}	y_{ol2}	y_{ol3}	y_{ol4}
1	0.85	0.65	0.55	0.75	0.86	0.66	0.65	0.85
2	0.85	0.65	0.66	0.86	0.75	0.55	0.65	0.85

Table 1. x- and y- directional coordinates of obstacles.

4.1.2 Setup of fitness function

In simulation, connection weights of the NN and parameters (center and standard deviations) of RBFs are optimized by a GA (Michalewicz, 1996) so that the robot avoids obstacles and reaches the destination with a minimum number of walk cycles. Table 2 shows design parameters for the GA used in simulation. The associated fitness function of an individual is defined by

$$\text{fitness} = \sum_{i=1}^{ob_n} (\text{fitness}_o + \text{fitness}_a + \text{fitness}_c) \quad (13)$$

whose solution is searched for as a minimization problem. ob_n is the number of environments considered in optimization. fitness_o is an evaluation function associated with penalty for collision with an obstacle. fitness_o is given by

$$\text{fitness}_o = \begin{cases} 0.0, & \text{if there is no collision} \\ [x_{de}^2(k) + y_{de}^2(k)] \times 10, & \text{otherwise} \end{cases} \quad (14)$$

Walking stops if the robot collides with an obstacle. fitness_a is an evaluation function related to joint constraints, i.e., whether each joint angle is in an admissible range or not. fitness_a is given by

$$\text{fitness}_a = \begin{cases} 0.0, & \text{if there is no outside of the range} \\ [x_{de}^2(k) + y_{de}^2(k)] \times 10, & \text{otherwise} \end{cases} \quad (15)$$

Walking stops if the joint exceeds joint constraints. fitness_c is an evaluation function related to walk cycles required to reach the destination, and given by

$$\text{fitness}_c = [x_{de}^2(k) + y_{de}^2(k)] \times \frac{T}{50} \quad (16)$$

T denotes the walk cycles required to move from the starting point to the destination while avoiding obstacles by stable walking. The maximum number of walk cycles T_{\max} in one environment is set to 50 and walking stops if the walk cycles exceed T_{\max} .

The number of individuals	100
Crossover rate	0.6 (uniform crossover)
Selection strategy	Tournament selection (3 individuals)
Elitist preserving strategy	10

Table 2. Design parameters for GA.

4.2 Simulation Result

4.2.1 Performance for training data

A typical control result after training the RBFNN using the above procedure is shown in Fig. 11. Environmental conditions for one of 96 combinations were as follows: x- and y-directional initial distance error of the destination were $(x_{de}, y_{de})=(-0.15, 2.2)$ [m]; x- and y-directional coordinates of two obstacles were $(x_{or1}, y_{or1})=(0.2, 0.85)$, $(x_{or2}, y_{or2})=(0.2, 0.65)$, $(x_{or3}, y_{or3})=(0.5, 0.55)$, $(x_{or4}, y_{or4})=(0.5, 0.75)$, $(x_{ol1}, y_{ol1})=(-0.5, 0.85)$, $(x_{ol2}, y_{ol2})=(-0.5, 0.65)$, $(x_{ol3}, y_{ol3})=(-0.1, 0.66)$, and $(x_{ol4}, y_{ol4})=(-0.1, 0.86)$ [m]; and z-directional coordinates were $z_{or}=0.3$ and $z_{ol}=0.12$ [m].

The robot turned to the left to avoid an obstacle at right, in which it could not get over but conquered the obstacle at left. Legs 1 and 3 were used to get over the obstacle at left and the robot reached the destination. The number of walk cycles to the goal was 9 and the final distance error was $(x_{de}, y_{de})=(-0.037, 0.046)$ [m].

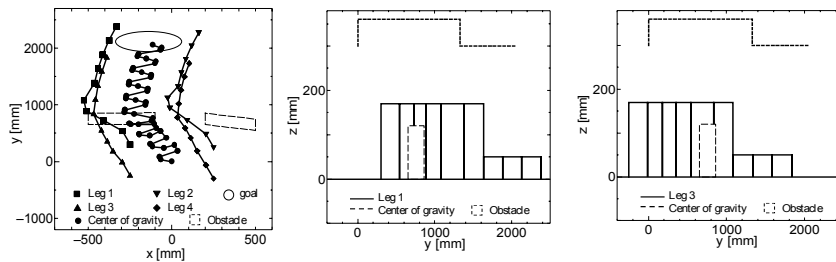


Fig. 11. Obstacle avoidance action for the trained environment.

4.2.2 Performance for untrained environment

In this section, we examine RBFNN performance in an untrained environment. We set the initial distance errors at $(x_{de}, y_{de})=(0.35, 2.1)$ [m], and x- and y-directional coordinates of two obstacles at $(x_{or1}, y_{or1})=(0.11, 0.87)$, $(x_{or2}, y_{or2})=(0.11, 0.56)$, $(x_{or3}, y_{or3})=(0.57, 0.55)$, $(x_{or4}, y_{or4})=(0.57, 0.86)$, $(x_{ol1}, y_{ol1})=(-0.56, 0.92)$, $(x_{ol2}, y_{ol2})=(-0.56, 0.59)$, $(x_{ol3}, y_{ol3})=(-0.14, 0.66)$, and $(x_{ol4}, y_{ol4})=(-0.14, 0.99)$ [m], together with z-directional coordinates at $z_{or}=0.07$ and $z_{ol}=0.3$ [m].

Fig. 12 shows the results. In simulation, since the obstacle at left could not be gotten over by the robot, it moved to the right and got over it. We found that legs 2 and 4 were used for getting over the obstacle at right so that the robot reached the destination. The number of walk cycles to the goal was 9 and the final distance error was $(x_{de}, y_{de})=(0.036, 0.104)$ [m].

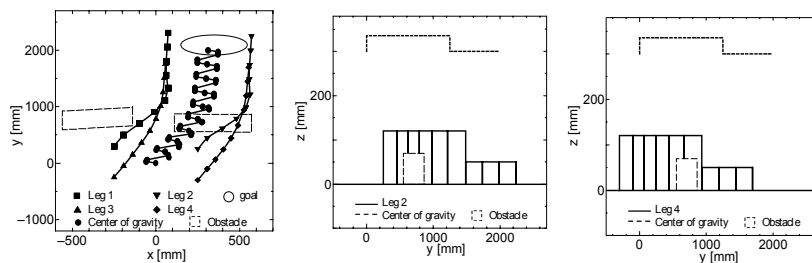


Fig. 12. Obstacle avoidance action for an untrained environment.

4.3 Experiments

Experiments were conducted by quadruped robot TITAN-VIII. Walk was at a crawl and the robot recognized an obstacle by ultrasonic sensors on the forelegs. x - and y -directional movement and the turning angle of the robot were determined by the NN from simulation. We assumed that the robot could not get over an obstacle taller than 0.15 [m]. Placement of a swing leg is decided by information retrieved by the downward ultrasonic sensor, depending on whether the placement is a corner of the obstacle. Whether the leg is a swing leg or not is decided by the force sensor.

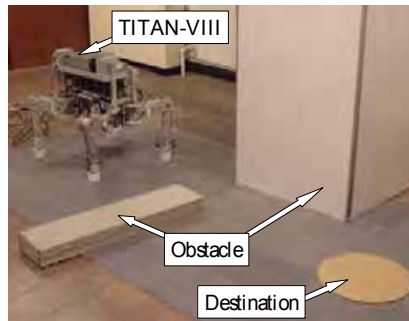


Fig. 13. Environment of an experiment.

The experimental environment is shown in Fig. 13. One obstacle is on the right and the other on the left. The obstacle at left can not be gotten over by the robot. Initial distance error was set to $(x_{de}, y_{de})=(0.0, 1.9)$ [m].

Positions of obstacles are detected by the robot (Figs. 14 and 15). Fig. 14 shows information on obstacles gathered by sensors on the forelegs, whose results show that only one side of each obstacle could be detected. Fig. 15 shows positions and heights of obstacles when the leg has passed over them, which were detected by downward sensors. The path to the destination and the presence of obstacles is shown in Fig. 16. We found that the robot turned to the right to avoid the obstacle at left, which could not be gotten over, but got over the obstacle at right. Legs 2 and 4 were used for getting over the obstacle at right and the robot reached the destination. The number of walk cycles to the goal was 9 and the final distance error was $(x_{de}, y_{de})=(0.137, 0.075)$ [m].

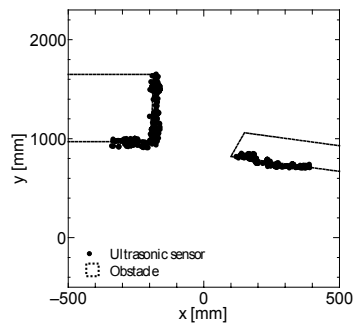


Fig. 14. Recognition result of the obstacles gathered by sensors on the forelegs.

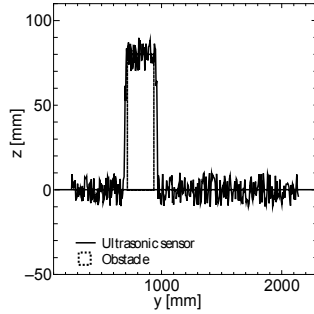


Fig. 15. Recognition result of the obstacles detected by downward sensors.

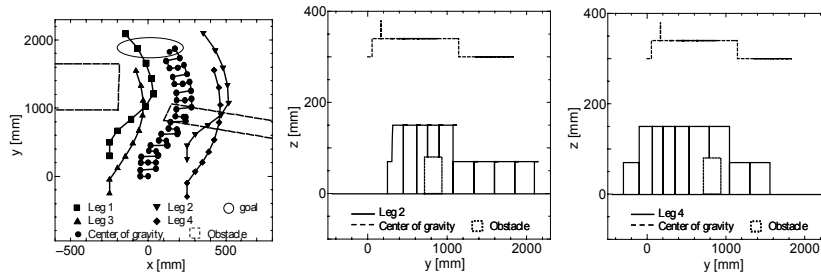


Fig. 16. Obstacle avoidance action in an actual experiment.

5. Determination of the Order of Swing Leg for Free-Gait

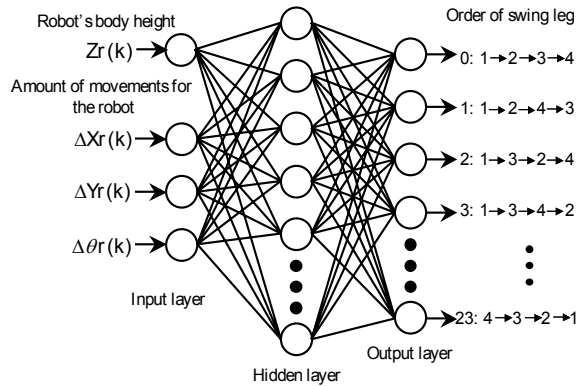


Fig. 17. Three layered neural network for determining the order of swing leg.

Since obstacle avoidance is taken into consideration, it is assumed that the static walk is adopted as a basic walk and the order of swing leg is determined. 24 kinds of the order exist in a static walk of the quadruped robot. Although 24 kinds of the order can be tried to implement whenever the

quadruped robot walks, in this research, the order of swing leg is determined by a three-layered NN shown in Fig. 17. Inputs to the NN are assumed to be the robot's body height $Zr(k)$, the amount of x- and y-directional movements of the robot $\{\Delta Xr(k), \Delta Yr(k)\}$ and the turning angle of the robot $\Delta\theta(k)$. Moreover, we prepare 24 units at the output, corresponding to 24 kinds of the order (Table 3). The order of swing leg fed to the quadruped robot uses the order associated to the unit whose output value is closest to one among output units. RBFNN is also used for the NN, where the number of units in the hidden layer is set to 50.

Output number	Order of swing leg	Output number	Order of swing leg
0	1→2→3→4	12	3→1→2→4
1	1→2→4→3	13	3→1→4→2
2	1→3→2→4	14	3→2→1→4
3	1→3→4→2	15	3→2→4→1
4	1→4→2→3	16	3→4→1→2
5	1→4→3→2	17	3→4→2→1
6	2→1→3→4	18	4→1→2→3
7	2→1→4→3	19	4→1→3→2
8	2→3→1→4	20	4→2→1→3
9	2→3→4→1	21	4→2→3→1
10	2→4→1→3	22	4→3→1→2
11	2→4→3→1	23	4→3→2→1

Table 3. Order of swing leg to each output of NN.

5.1 Acquisition of the Order of Swing Leg

In a static walk of quadruped robot, since the amount of movements of the body changes with the order of swing leg, the robot produces different movable range for each order of swing leg. For this reason, if the stability of static walk is maintained by less movement of the body, then the movable range of each leg becomes large; it can insuquently enlarge the movable range of the robot.

For teacher signal used for this research, when the robot's body height $Zr(k)$ was changed from 300 [mm] to 370 [mm], the amount of x-directional movement of the robot $\Delta Xr(k)$ is changed from -100 [mm] to 100 [mm], the amount of y-directional movement of the robot $\Delta Yr(k)$ is changed from 150 [mm] to 350 [mm] and the turning angle of the robot $\Delta\theta(k)$ is changed from -15 [degree] to 15 [degree], respectively, the order of swing leg that the movement of the robot's body is a minimum and the stability of static walk is satisfied is set as one, and the other order is set to zero. Here, there were 19 kinds of the order of swing leg that the amount of movements of the robot's body became the minimum. Note however that when the amount of changes of $Zr(k)$, $\Delta Xr(k)$, $\Delta Yr(k)$, and $\Delta\theta(k)$ is fixed, the number of selections is changed, depending on the order of swing leg. Therefore, the amount of change of $Zr(k)$, $\Delta Xr(k)$, $\Delta Yr(k)$, and $\Delta\theta(k)$ is enlarged for the case of high number of selections, whereas the amount of change of movement is made small for the case of low number of selections, and 20 data are prepared for each order of swing leg. Moreover, it is assumed that initial leg positions of quadruped robot are set to $(x_{f1}, y_{f1})=(-0.25, 0.3)$, $(x_{f2}, y_{f2})=(0.25, 0.25)$, $(x_{f3}, y_{f3})=(-0.25, -0.25)$, and $(x_{f4}, y_{f4})=(0.25, -0.3)$ [m]. Here, the subscript number denotes the leg number.

In this research, the order of swing leg is determined using RBFNN. Connection weights of the NN and parameters (center and standard deviations) of RBFs are optimized by a GA so that the relation between an input and an output is satisfied. There is 19 kinds of the order of swing leg chosen when changing $Zr(k)$, $\Delta Xr(k)$, $\Delta Yr(k)$, and $\Delta\theta r(k)$, respectively, and 20 data are prepared for each one. Here, there are a total of 380 kinds of combination optimized by using GA. The associated fitness function of an individual is defined by

$$fitness = \sum_{i=1}^n (fitness_A + fitness_B) \tag{17}$$

whose solution is searched for as a minimization problem. n is the total number of combinations in optimization. $fitness_A$ is an evaluation function only applied when a teacher signal ts_j is one, which is given by

$$fitness_A = \sqrt{(o_j - ts_j)^2} \Big|_{j=j1} \tag{18}$$

where j denotes any unit number of output layer and $j1$ denotes the unit number of $ts_j \equiv 1$. Contrarily, $fitness_B$ is an evaluation function applied when a ts_j is zero, which is given by

$$fitness_B = \sqrt{\max\{(o_1 - ts_1)^2, \dots, (o_j - ts_j)^2, \dots, (o_{24} - ts_{24})^2\}} \tag{19}$$

$fitness_B$ denotes the largest value in the difference of o_j and ts_j . j denotes the unit number 1 to 24 except for the case of $j1$ that denotes the unit number of $ts_j \equiv 1$.

5.2 Obstacle Avoidance with Consideration to the Free-Gait

A block diagram of obstacle avoidance control system considered here is shown in Fig. 18. The avoidance action of quadruped robot is determined by the upper NN. Furthermore, the order of swing leg is determined by the lower NN from the amount of movements of the robot.

The simulation environment is the same as section 4.1. The y-axis is set to the forward direction of the robot. The robot is assumed to start from point (0.0, 0.0, 0.3) [m] and approach the goal, a circle having radius 0.2 [m], centered at the destination point.

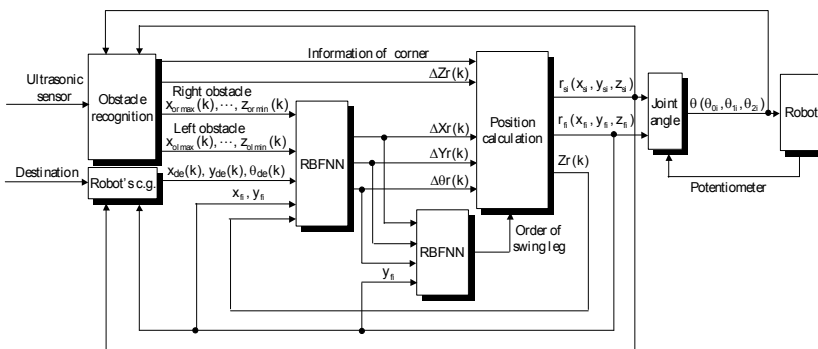


Fig. 18. Obstacle avoidance control system with consideration to the order of swing leg.

We set the initial distance errors at $(x_{de}, y_{de})=(-0.15, 2.2)$ [m], and x- and y-directional coordinates of two obstacles at $(x_{or1}, y_{or1})=(0.1, 0.85)$, $(x_{or2}, y_{or2})=(0.1, 0.65)$, $(x_{or3}, y_{or3})=(0.5, 0.66)$, $(x_{or4}, y_{or4})=(0.5, 0.86)$, $(x_{ol1}, y_{ol1})=(-0.5, 0.75)$, $(x_{ol2}, y_{ol2})=(-0.5, 0.55)$, $(x_{ol3}, y_{ol3})=(-0.2, 0.65)$, and $(x_{ol4}, y_{ol4})=(-0.2, 0.85)$ [m], together with z-directional coordinates at $z_{or}=0.12$ and $z_{ol}=0.3$ [m]. Here, although a robot can get over an obstacle at right, an obstacle at left shall not be get over. The robot's body height $Z_r(k)$ is adjusted and changed to the height of the obstacle which can be get over.

5.2.1 When the order of swing leg is fixed

The movement path of the quadruped robot when fixing the order of swing leg and avoiding an obstacle is shown in Fig. 19, and the corresponding amount of movements of the robot's body Br is shown in Table 4. Here, the leg number used as swing leg is assigned as the left foreleg to 1, right foreleg to 2, left hind-leg to 3 and right hind-leg to 4, as shown in Fig. 2. The robot's body height changes as shown in Fig. 20. Furthermore, it can be checked from Fig. 20 that both the leg 2 and 4 were used for a getting over to the obstacle at right.

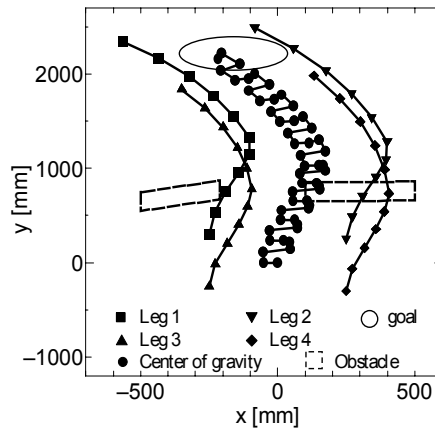


Fig. 19. Movement path of the quadruped robot when fixing the order of swing leg.

$\Delta X_r(k)$ [mm]	$\Delta Y_r(k)$ [mm]	$\Delta \theta(k)$ [deg]	$Z_r(k)$ [mm]	Br [mm]	Order of swing leg
22.026	237.002	0.809	300.0	418.090	4→2→3→1
43.958	217.174	-0.818	360.0	372.904	4→2→3→1
45.347	200.484	-0.639	360.0	354.896	4→2→3→1
32.227	189.676	1.413	360.0	362.871	4→2→3→1
12.261	186.691	4.182	360.0	390.217	4→2→3→1
-15.944	241.234	3.894	360.0	520.653	4→2→3→1
-17.443	240.270	4.289	360.0	528.034	4→2→3→1
-18.958	239.279	4.695	300.0	535.676	4→2→3→1
-20.522	238.245	5.122	300.0	543.786	4→2→3→1
-22.187	237.142	5.586	300.0	552.667	4→2→3→1

Table 4. The amount of movements of the robot's body when fixing the order of swing leg.

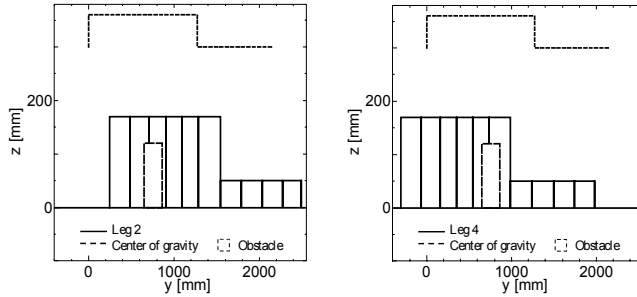


Fig. 20. Movement path of the leg 2 and 4.

5.2.2 When all 24 kinds of the order are tried

The movement path of the quadruped robot, when trying all 24 kinds of the order and avoiding an obstacle, is shown in Fig. 21, and the amount of movements of the robot's body Br is shown in Table 5. Here, the robot's body height changes with obstacles. For this reason, the robot's body height varies as shown in Fig. 20. Compared to the case where the order of swing leg is fixed, it is found that the amount of movements of the robot's body is relatively small.

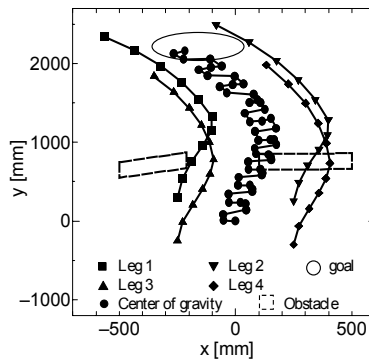


Fig. 21. Movement path of the quadruped robot when trying all 24 kinds of the order.

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta \theta r(k)$ [deg]	$Zr(k)$ [mm]	Br [mm]	Order of swing leg
22.026	237.002	0.809	300.0	417.078	2→4→3→1
43.958	217.174	-0.818	360.0	372.904	4→2→3→1
45.347	200.484	-0.639	360.0	354.896	4→2→3→1
32.227	189.676	1.413	360.0	362.871	4→2→3→1
12.261	186.691	4.182	360.0	389.931	2→4→3→1
-15.944	241.234	3.894	360.0	520.653	4→2→3→1
-17.443	240.270	4.289	360.0	528.034	4→2→3→1
-18.958	239.279	4.695	300.0	448.019	3→1→4→2
-20.522	238.245	5.122	300.0	446.065	3→1→4→2
-22.187	237.142	5.586	300.0	443.984	3→1→4→2

Table 5. The amount of movements of the robot's body when trying all 24 kinds of the order.

5.2.3 When the order of swing leg is determined by RBFNN

After training the lower RBFNN, the movement path of the quadruped robot, determining the order of swing leg and avoiding an obstacle, is shown in Fig. 22. The corresponding amount of movements of the robot's body Br is shown in Table 6. Note here that the robot's body height is the same as that shown in Fig. 20. Compared to the case where the order of swing leg is fixed, it is observed that the amount of movements of the robot's body is small. However, compared with the case where all 24 kinds of the order are tried, the amount of movements of the body was slightly large.

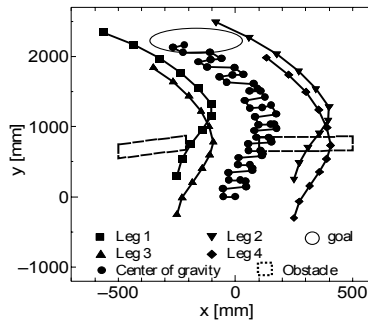


Fig. 22. Movement path of the quadruped robot when determining the order of swing leg using RBFNN.

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta \theta r(k)$ [deg]	$Zr(k)$ [mm]	Br [mm]	Order of swing leg
22.026	237.002	0.809	300.0	418.090	4→2→3→1
43.958	217.174	-0.818	360.0	372.904	4→2→3→1
45.347	200.484	-0.639	360.0	354.896	4→2→3→1
32.227	189.676	1.413	360.0	362.871	4→2→3→1
12.261	186.691	4.182	360.0	390.217	4→2→3→1
-15.944	241.234	3.894	360.0	520.653	4→2→3→1
-17.443	240.270	4.289	360.0	528.034	4→2→3→1
-18.958	239.279	4.695	300.0	448.019	3→1→4→2
-20.522	238.245	5.122	300.0	446.065	3→1→4→2
-22.187	237.142	5.586	300.0	443.984	3→1→4→2

Table 6. The amount of movements of the robot's body when determining the order of swing leg using RBFNN.

6. Re-learning of the NN for Determining the Robot Action

NN for determining the robot action is acquired by re-learning the NN that was built in the case when the order of swing leg was fixed.

A block diagram of obstacle avoidance control system is the same as section 5.2. Moreover, the simulation condition and fitness function are the same as section 4.1.

6.1 When the Obstacle at Right is a Wall

We set the initial distance errors at $(x_{de}, y_{de}) = (-0.15, 2.2)$ [m], and x- and y-directional coordinates of two obstacles at $(x_{or1}, y_{or1}) = (0.2, 0.85)$, $(x_{or2}, y_{or2}) = (0.2, 0.65)$, $(x_{or3}, y_{or3}) = (0.5, 0.55)$, $(x_{or4}, y_{or4}) = (0.5,$

0.75), $(x_{ol1}, y_{ol1})=(-0.5, 0.85)$, $(x_{ol2}, y_{ol2})=(-0.5, 0.65)$, $(x_{ol3}, y_{ol3})=(-0.1, 0.66)$, and $(x_{ol4}, y_{ol4})=(-0.1, 0.86)$ [m], together with z-directional coordinates at $z_{or}=0.3$ and $z_{ol}=0.12$ [m]. Here, although a robot can get over an obstacle at left, an obstacle at right shall not be get over.

The simulation result using the learned RBFNN is shown in Fig. 23. The number of walk cycles to the goal was 10, and the distance from the COG of the robot to the destination point was $(x_{de}, y_{de})=(0.017, -0.027)$ [m]. Moreover, the amount of movements of the robot and the order of swing leg are shown in Table 7. It was found that the order of the swing leg changes with the amount of movements of the robot.

In the simulation result of the case where unlearned RBFNN is used, the number of walk cycles to the goal was 9, and the distance from the COG of the robot to the destination point was $(x_{de}, y_{de})=(-0.037, -0.046)$ [m].

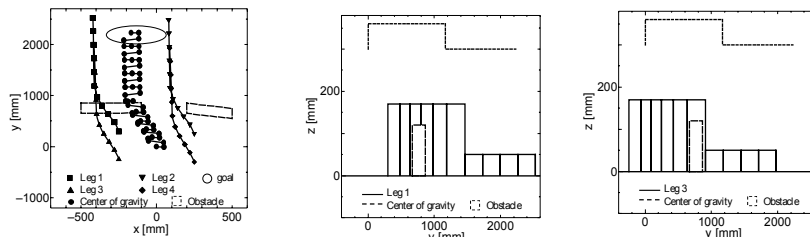


Fig. 23. Movement path of quadruped robot when changing the order of swing leg using RBFNN (in the case where an obstacle at right is a wall).

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta \theta r(k)$ [deg]	$Zr(k)$ [mm]	Order of swing leg
-26.645	183.469	0.751	300.0	3→1→4→2
-35.059	157.334	0.497	360.0	3→1→4→2
-32.240	162.725	0.311	360.0	3→1→4→2
-24.792	181.329	0.089	360.0	3→1→4→2
-14.077	211.723	-0.166	360.0	3→1→4→2
2.710	274.449	-0.255	360.0	4→2→3→1
2.834	266.891	-0.172	300.0	4→2→3→1
1.787	265.774	-0.159	300.0	4→2→3→1
0.661	264.499	-0.143	300.0	4→2→3→1
-0.516	263.101	-0.126	300.0	4→2→3→1

Table 7. The amount of movements of the robot in the case where an obstacle at right is a wall.

6.2 When the Obstacle at Left is a Wall

We set the initial distance errors at $(x_{de}, y_{de})=(-0.15, 2.2)$ [m], and x- and y-directional coordinates of two obstacles at $(x_{or1}, y_{or1})=(0.1, 0.85)$, $(x_{or2}, y_{or2})=(0.1, 0.65)$, $(x_{or3}, y_{or3})=(0.5, 0.66)$, $(x_{or4}, y_{or4})=(0.5, 0.86)$, $(x_{ol1}, y_{ol1})=(-0.5, 0.75)$, $(x_{ol2}, y_{ol2})=(-0.5, 0.55)$, $(x_{ol3}, y_{ol3})=(-0.2, 0.65)$, and $(x_{ol4}, y_{ol4})=(-0.2, 0.85)$ [m], together with z-directional coordinates at $z_{or}=0.12$ and $z_{ol}=0.3$ [m]. Here, although a robot can get over an obstacle at right, an obstacle at left shall not be get over.

The simulation result is shown in Fig. 24. The number of walk cycles to the goal was 11, and the distance from the COG of the robot to the destination point was $(x_{de}, y_{de})=(0.018, -0.014)$ [m].

Moreover, the amount of movements of the robot and the order of swing leg are shown in Table 8. In the simulation result when unlearned RBFNN is used, the number of walk cycles to the goal was 10, and the distance from the COG of the robot to the destination point was $(x_{de}, y_{de})=(0.068, 0.027)$ [m].

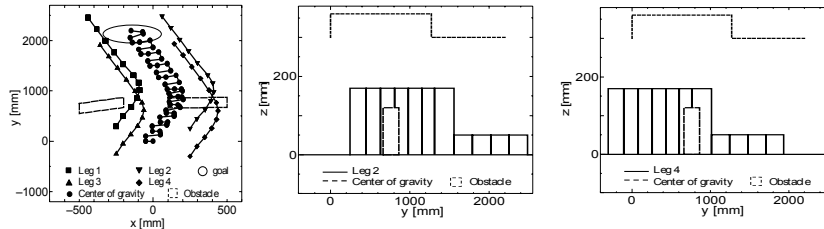


Fig. 24. Movement path of quadruped robot when changing the order of swing leg using RBFNN (in the case where an obstacle at left is a wall).

$\Delta Xr(k)$ [mm]	$\Delta Yr(k)$ [mm]	$\Delta \theta(k)$ [deg]	$Zr(k)$ [mm]	Order of swing leg
45.379	198.899	-0.298	300.0	4→2→3→1
55.253	181.795	0.837	360.0	4→2→3→1
49.297	172.875	1.419	360.0	4→2→3→1
36.077	165.824	1.896	360.0	2→4→3→1
18.925	159.794	2.220	360.0	2→4→3→1
-0.386	155.208	2.319	360.0	2→4→1→3
-23.440	241.344	0.167	360.0	4→2→3→1
-23.620	239.286	0.087	300.0	4→2→3→1
-24.022	237.751	0.013	300.0	4→2→3→1
-24.397	236.259	-0.060	300.0	4→2→3→1
-24.746	234.816	-0.131	300.0	4→2→3→1

Table 8. The amount of movements of the robot in the case where an obstacle at left is a wall.

7. Conclusions

We experimentally have proved a method for acquiring a path to a destination and obstacle avoidance of a quadruped robot. Robot actions were determined through an RBFNN, whose input consisted of destination information, obstacle configuration, and current robot status. Using training data on environmental conditions, focusing on x -, y -, and z -coordinates of different obstacles and certain destinations, RBFNN design parameters were optimized using a GA so that the robot reached the destination with a minimum number of walking cycles. For an untrained (unknown) environment, we found that the RBFNN was useful for acquiring an obstacle avoidance path to the destination. Effectiveness of this approach was examined by actual experiments. However, free-gait motion was not taken into consideration in the first research.

A method of determining the order of swing leg in free gait by an RBFNN, whose inputs are the amount of movements for the quadruped robot and the height of the body, has been proposed for the second research. In the tuning of design parameters of the RBFNN, 20 data to which the amount of movements for the robot was changed are prepared for each order of swing leg. Such design parameters were optimized using GA so that the relation between an input and an output is satisfied.

As a result, compared to the case where the order of swing leg is fixed, the amount of movements for the robot body was small. However, compared to the case where all 24 kinds of the order are tried, that for the robot body was slightly large. It seems to be attributed to the fact that there was few data used for the study of the RBFNN. In order to make the amount of movements for the robot body much smaller, more data need to be used for the study of RBFNN.

In order to acquire the obstacle avoidance action of quadruped robots with considering the order of swing leg, the action of a quadruped robot has been determined through an RBFNN, whose inputs were the destination information, the obstacle configurations, and the robot's self-state. The NN for determining the robot's action is acquired by re-learning the NN that was built in the case where the order of swing leg was fixed. Compared to the case where the unlearned RBFNN is used, the final distance error to the destination of the present approach was small; however the walk cycle was comparable to each other. It is attributed to the fact that a priority was assigned to the error distance in the evaluation of GA. For this reason, in order to make a walk cycle smaller, further fitness function of GA needs to be re-examined. Moreover, the effectiveness of the proposed system needs to be verified by using the actual system.

Obstacles are recognized by an ultrasonic sensor that detects reflected ultrasonic waves on a flat surface. Obstacles were assumed to be flat, i.e., rectangular blocks and oblique obstacles were not considered. Since the order of the swing leg was assumed to be constant, this assumption appears to have slightly restricted the robot action. We will improve the mobility efficiency of the robot by constructing a system that changes the sequence of the swing leg based on environmental conditions and improve recognition system by adding a vision sensor (Chow & Chung, 2002).

8. References

- Chen, X.; Watanabe, K.; Kiguchi, K. & Izumi, K. (2002). An ART-based fuzzy controller for the adaptive navigation of a human-coexistent quadruped robot, *IEEE/ASME Transactions on Mechatronics*, Vol. 7, No 3, pp. 318-328
- Chen, X.; Watanabe, K.; Kiguchi, K. & Izumi, K. (2002). Path tracking based on closed-loop control for a quadruped robot in a cluttered environment, *ASME, Journal of Dynamic Systems, Measurement, and Control*, Vol. 124, pp. 272-280
- Chow, Y.H. and Chung, R. (2002), VisionBug: A hexapod robot controlled by stereo cameras, *Autonomous Robots*, Vol. 13, No. 3, pp. 259-276.
- Elanayar, V. T. S. & Shin, Y. C. (1994). Radial basis function neural network for approximation and estimation of nonlinear stochastic, *IEEE Transactions on Neural Networks*, Vol. 5, No. 4, pp. 594-603
- Furusho, J. (1993). Research deployment of the walking robot, *Journal of Robotics Society of Japan*, Vol. 11, No 3, pp. 306-313
- Hirose, S. & Arikawa, K. (1999). Development of quadruped walking robot TITAN-VIII for commercially available research platform, *Journal of Robotics Society of Japan*, Vol. 17, No 8, pp. 1191-1197
- Hirose, S. & Yoneda, K. (1993). Toward development of practical quadruped walking vehicle, *Journal of Robotics Society of Japan*, Vol. 11, No 3, pp. 360-365
- Kimura, H. (1993). Dynamic walk of the quadruped robot, *Journal of Robotics Society of Japan*, Vol. 11, No 3, pp. 372-378

- Kimura, H.; Shimoyama, I. & Miura, H. (1990). Dynamics in the dynamic walk of a quadruped robot, *Advanced Robotics*, Vol. 4, No. 3, pp. 283-301
- Michalewicz, Z. (1996). *Genetic Algorithms + Data Structure = Evolution Programs*, 3rd, revised and extended edition ed., Springer, Germany
- Nakamura, T.; Seki, M.; Mori, Y. & Adachi, H. (1999). Free gait planning using Monte Carlo method for locomotion on rugged terrain, *1999 JSME Conference on Robotics and Mechatronics*, 1A1-42-061 (CD-ROM)
- Şafak, K. K. & Adams, G. G. (2002). Dynamic modeling and hydrodynamic performance of biomimetic underwater robot locomotion, *Autonomous Robots*, Vol. 13, No. 3, pp. 223-240
- Sakawa, M. & Tanaka, M. (1999). *Introduction to Neurocomputing*, Tokyo, Morikita Shuppan Co., Ltd.

Fault-Tolerant Gait Planning of Multi-Legged Robots

Jung-Min Yang*, Yong-Kuk Park** and Jin-Gon Kim**

*Department of Electrical Engineering, Catholic University of Daegu

**School of Mechanical and Automotive Engineering, Catholic University of Daegu
Republic of Korea

1. Introduction

A fault-tolerant gait of multi-legged systems is defined as a gait which can maintain the gait stability and continue its walking against the occurrence of a leg failure (Yang & Kim, 1998). The notion of the fault-tolerant gait comes from the fact that legged robots with static walking have inherent fault tolerance capability against a failure in a leg, since a failed leg for itself does not cause fatal breakdown or instability to walking motions (Nagy et al., 1994). This means that for a given type of failure, the problem of finding fault-tolerant gaits can be formulated with which legged robots can continue their walking after an occurrence of a failure, maintaining static stability. As a novel field of gait study, fault-tolerant gaits are worth researching since the feature of leg failure can be involved into the frame of gait study and its adverse effect on gait planning can be analyzed with performance criteria such as stability margin, stride length, etc.

Fault-tolerant gaits are classified by the kind of leg failure to be tolerated and the point of time that fault tolerance is carried out, that is, before or after a failure occurs. Several algorithms of fault-tolerant gaits developed in the past can be sorted out by these categories (Chu & Pang, 2002; Lee & Hirose, 2002, Nagy et al, 1994; Yang, 2002). Among them, Yang (Yang, 2002) proposed fault-tolerant gaits for post-failure walking of quadruped robots with a locked joint failure, in which a joint of a leg is locked in a known place (Lewis & Maciejewski, 1997). As one of common failures that can be frequently observed in dynamics of robot manipulators, the locked joint failure reduces the number of degrees of freedom of the robot manipulator by one and consequently its workspace to a certain limit. To establish the fault-tolerance scheme, the reduction of the workspace of a failed leg should be interpreted and reflected based on gait study.

The objective of this article is to develop the fault-tolerant gait algorithm for a locked joint failure when the model of legged robots is *hexapod*. Compared with the previous results on quadruped robots (Yang, 2002, Yang, 2003), the following aspects will be considered in this article for the motion of hexapod robots. First, quadruped robots can have only one gait pattern in static walking, (4→3→4→3···), i.e., one leg is lifted off and swung while other three legs are in the support phase. But hexapod robots can have variable gait pattern, i.e., tripod, quadruped and pentaped gaits. In this article, we show that fault tolerance can be realized for walking with any gait and, especially, periodic gaits can be generated using tripod and quadruped gaits for straight-line walking on even terrain. We also present fault-tolerant gaits with non-zero crab angle, or a walking motion with the direction of

locomotion different from the longitudinal axis of the robot body. Second, the previous result did not take into account kinematic constraints of the failed leg in developing fault-tolerant gait planning. But, there exist singularities in the configuration of a failed leg where legged robots cannot change the present gait due to the failure and are fallen into dead-lock state. We verify that for the existence of fault-tolerant gaits, the configuration of the failed leg must be within a prescribed range of kinematic constraints.

This article is organized as follows: In Section 2, a hexapod prototype is described with conditions on walking mechanism. The configuration of the locked joint failure is also addressed. The kinematic constraints for the existence of fault-tolerant gaits are provided in Section 3. In Section 4, a general scheme for fault-tolerant gait planning is proposed for straight-line walking of a hexapod robot over even terrain. In particular, periodic gaits are derived from the proposed scheme with formulations of the stride length and stability margin. In Section 5, based on the principles of fault-tolerant gait planning for a locked joint failure, periodic crab gaits are proposed in which a hexapod robot has tripod walking after a joint of a leg is locked from failure. In Section 6, a post-failure walking example is addressed in which a hexapod robot walking with the wave gait before a failure could realize fault tolerance by the proposed scheme. Finally, some concluding remarks are given in Section 7.

2. Preliminaries

2.1 Modeling of Hexapod

A two-dimensional model of a hexapod robot is shown in Figure 1. The six legs are placed symmetrically about the longitudinal axis and have rectangular working areas with the length R_x and the width R_y . C_i is the center point of leg i 's working area. The robot body is also in the shape of a rectangle with $2U$ width and distant from working areas by W . C is the center of gravity of the body and the origin of the robot coordinate system X - Y . The crab angle α ($-180^\circ < \alpha \leq 180^\circ$) is defined as the angle between the longitudinal axis of the robot body and the direction of crab walking. α_d , a robot parameter that will be used later in this paper, is defined as the angle between the off-diagonal and the base of the working area.

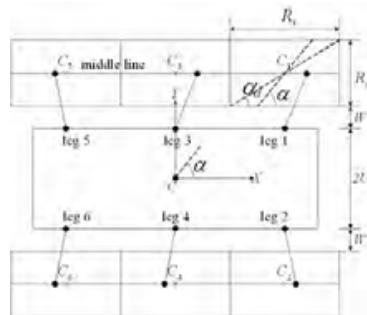


Fig. 1. Two-dimensional model of a hexapod robot.

It is supposed that a leg attached to the hexapod robot has the geometry of the articulated arm (Lewis et al., 1993) shown in Figure 2. This model has two rigid links and three revolute joints; the lower link is connected to the upper link via an active revolute joint and the upper

link is connected to the body via two active revolute joints, one parallel with the knee joint and the other parallel with the body longitudinal axis. Hence the foot point has three degrees of freedom with respect to the body and the overall walking can be driven in any direction. We denote the joint at the main actuator as *joint one*, the joint at the lifting actuator as *joint two*, and the joint at the knee actuator as *joint three*. θ_1 , θ_2 and θ_3 are values of each joint angle, and l_1 and l_2 are lengths of the upper and lower links, respectively.

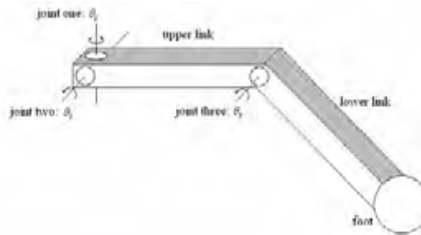


Fig. 2. Three-joint leg model.

By adopting the major premise of gait study (Song & Waldron, 1987), we assume that the robot body remains parallel to the ground, and that the ground is flat over the region affecting the robot workspace. The dimension of the working area of each leg is thus kept the same throughout walking. Moreover, it is assumed that the hexapod robot has static walking in which dynamic effects of legs and the body are negligible. All the mass of the legs is supposed to be lumped into the body and the orientation of the support surface with respect to the gravity vector is irrelevant. Because walking on uneven terrain is precluded in the study, we properly suppose that the hexapod robot has straight-line walking with a regular gait, that is, each leg tracks on the middle line or crab line with angle α of its working area (see Figure 1), paralleled with the trajectory of the center of gravity.

2.2 Review of Locked Joint Failure

A single locked joint failure reduces the number of degrees of freedom in the leg by one and hence inflicts serious damage on mobility of the failed leg (Lewis & Maciejewski, 1997). However, unlike free-swinging failure (Shin & Lee, 1999) and mutilation failure, the locked joint failure does not take away supporting ability from the failed leg. For employing the failed leg in post-failure walking, we should examine the configuration of the failed leg determined by the position of a locked joint and the resulting change of the reachable area.

Figure 3 illustrates the behaviour of a failed leg with the geometry of Figure 2. After joint one of a leg is locked from failure, the kinematics of the failed leg is the same as a two-link revolute joint manipulator. Its workspace is reduced to the plane made of the two links and the reachable region of the foothold position in the working area is projected onto a straight-line of which the lateral view is shown in Figure 3(a). $\hat{\theta}_1$ denotes the locked angle of joint one, and the values of θ_2 and θ_3 are determined by the foothold position. The locked failure at joint two or three yields almost identical post-failure behavior. When joint two or three is locked, the failed leg is tantamount to a one-link manipulator with two revolute joints. Since the altitude of the robot body is assumed to be constant, if one of joints two and three is

locked, the angle of another joint is fixed too for a given foothold position. For instance, joint two is assumed to be locked at $\hat{\theta}_2$ and the failed leg is placed onto the point P in Figure 3(b). Though the leg might have another foothold position P' in kinematics, it is impossible to place onto the point because the altitude of the robot body does not change. Therefore, joint three is also "locked" at $\hat{\theta}_3$ in the configuration of Figure 3(b), and the failed leg moves by swinging joint one as shown in Figure 3(c), making the track of foothold positions in the shape of an arc.

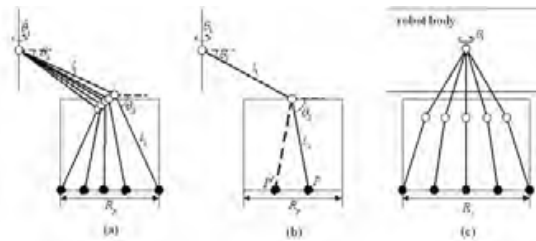


Fig. 3. Locked joint failure: (a) lateral view of the locked failure at joint 1, (b) lateral view of the locked failure at joint 2 (or joint 3) and (c) front view of the locked failure at joint 2 (or joint 3).

3. Kinematic Constraints

While the previous study investigated constrained motions of a failed leg in detail, it did not take into account kinematic constraints necessary for the existence of fault-tolerant gaits. In this section, we show that there is a range of kinematic constraints which the configuration of a failed leg should satisfy for guaranteeing the existence of fault-tolerant gaits.

3.1 Straight-line motion

3.1.1 Failure of joint one

In the first, let us consider the case where joint one is locked from failure. The failed leg in this case cannot take longitudinal swing with respect to the body and can take only lateral swing using the remaining normal joints as shown in Figure 4(a), where $\hat{\theta}_1$ is the locked angle of joint one. The motion of such a leg becomes that of a two-link revolute-joint manipulator. Its workspace is reduced to the plane made of the links and the reachable region of the foothold position in the working area is projected onto a straight-line shown in Figure 4(b). The failed leg can place its foot on the one and only foothold position, denoted by P , the intersection point of the foot trajectory (middle line of the working area) and the reachable line. For such an intersection point to exist, locked angle $\hat{\theta}_1$ must be in the range of

$$\hat{\theta}_{1,\min} \leq \hat{\theta}_1 \leq \hat{\theta}_{1,\max}$$

as shown in Figure 4(b), where all the angles are measured from the bisecting line of the working area in the clockwise direction. $\hat{\theta}_{1,\min}$ and $\hat{\theta}_{1,\max}$ are calculated as

$$\hat{\theta}_{1,\min} = -\arctan\left(\frac{R_x}{R_y + 2W}\right)$$

$$\hat{\theta}_{1,\max} = \arctan\left(\frac{R_x}{R_y + 2W}\right)$$

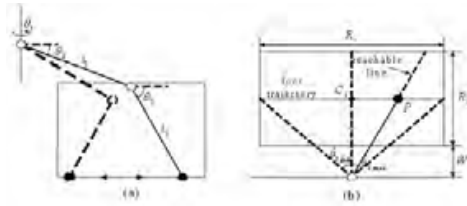


Fig. 4. Locked failure at joint one in straight-line walking: (a) lateral view and (b) kinematic constraint.

3.1.2 Failure of joint two

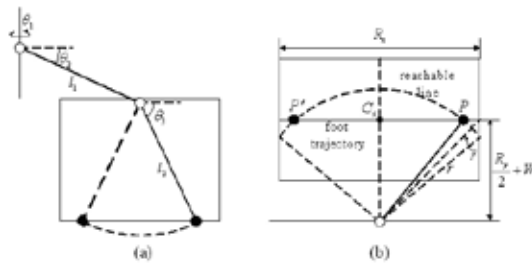


Fig. 5. Locked failure at joint two in straight-line walking: (a) lateral view and (b) kinematic constraint.

When joint two is locked from failure, the failed leg can move only the lower link by joint three for vertical swing. Depending on the configuration at the moment of failure, the failed leg can be placed on the inner foothold position or the outer position as shown in Figure 5(a). The resulting reachable region on the working area is thus projected onto a line of arc shape shown in Figure 5(b). Unlike the case of joint one where there is only one foothold position, the failed leg in this case can place its foot on one of two possible foothold positions, P and P' , in the foot trajectory. The kinematic constraint for guaranteeing such foothold positions can be described as

$$\frac{R_y}{2} + W \leq r \leq \bar{r} \tag{1}$$

where r is the radius of the arc and \bar{r} is the distance between the leg attachment point and the front-end(or rear-end) position projected onto the X - Y plane. If the radius of the arc r is in the above range, there exists at least one intersection point of the arc and the foot trajectory. For describing (1) in terms of joint angles and robot parameters, let us rewrite \bar{r} and r as

$$\begin{aligned} \bar{r} &= \frac{1}{2} \sqrt{R_x^2 + (R_y + 2W)^2} \\ r &= l_1 \cos \hat{\theta}_2 + l_2 \cos \theta_3 \end{aligned} \tag{2}$$

where $\hat{\theta}_2$ is the locked angle of joint two. Note that r is identical to the length of the leg projection onto the working area. Since the robot body is supposed to have a constant altitude, the angle of joint three θ_3 should also be fixed in failure mode (see Figure 5(a)). Substituting (2) into (1) leads to

$$\frac{R_y}{2} + W \leq l_1 \cos \hat{\theta}_2 + l_2 \cos \theta_3 \leq \frac{1}{2} \sqrt{R_x^2 + (R_y + 2W)^2}$$

The above result prescribes the kinematic constraint of locked angle $\hat{\theta}_2$ which guarantees the existence of the fault-tolerant gait for straight-line walking.

Consider the case of a locked failure at joint three as a remark. The constrained motion of the failed leg is almost identical to the case of joint two. If joint three is locked from failure, the leg is reduced to a one-link manipulator with two revolute joints (Craig, 2003), and the restricted reachable region projected onto the working area is of arc shape too. Therefore, the kinematic constraint for the existence of the fault-tolerant gait is the same as the case of joint two:

$$\frac{R_y}{2} + W \leq l_1 \cos \theta_2 + l_2 \cos \hat{\theta}_3 \leq \frac{1}{2} \sqrt{R_x^2 + (R_y + 2W)^2}$$

where $\hat{\theta}_3$ is the locked angle of joint three.

3.2 Crab Walking

In crab walking, the damage caused by a locked joint failure turns out to be the reduction of the range of the crab angle that the hexapod can have after a failure (Yang, 2003). We investigate such kinematic constraints for a given configuration of a locked failure.

3.2.1 Failure of joint one

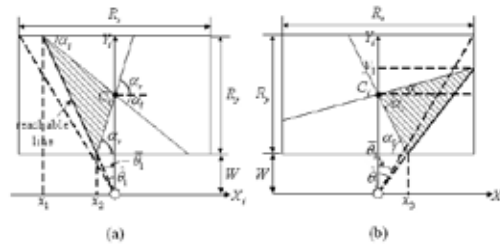


Fig. 6. Kinematic constraint for the locked failure in crab walking at joint one: (a) $|\hat{\alpha}_1| < |\bar{\alpha}_1|$ and (b) $|\hat{\alpha}_1| \geq |\bar{\alpha}_1|$.

Figure 6 illustrates the kinematic constraint for fault-tolerant crab gaits when joint one of a leg is locked at $\hat{\alpha}_1$. For clarity's sake, we assume that leg i 's coordinate system $x_i - y_i$ is

located at the leg attachment point and the angle of joint one is measured from the bisecting line of the working area (the Y_1 axis) in the clockwise direction, whereas the crab angle is measured from the X axis of the robot coordinate system in the counterclockwise direction (see Figure 1). As was illustrated in Figure 3, the reachable region of the foothold position is projected onto a straight-line in the working area. For the hexapod robot to be able to have crab walking after a failure, the foot trajectory must intersect the reachable line, in which the intersection point is to be the foothold position of the failed leg throughout post-failure walking. Therefore, the crab angle α should be within the shaded areas depicted in Figure 6 and should satisfy the following kinematic constraint:

$$\alpha_l \leq \alpha \leq \alpha_r \quad (3)$$

where α_l and α_r are the leftmost and rightmost limits, respectively. Since the working area is of rectangular shape, α_l and α_r are expressed differently according to the relative values of $\bar{\theta}_1$ and $\hat{\theta}_1$, where $\bar{\theta}_1$ is the locked angle of joint one that makes the reachable line cross the upper corner of the working area.

$$i) \quad |\hat{\theta}_1| < |\bar{\theta}_1|$$

If the locked angle is in the above range, the reachable line ends at the upper boundary of the working area. Referring to Figure 6(a), α_l is described as

$$\alpha_l = -\arctan\left(\frac{R_y}{2x_1}\right)$$

x_1 , the end position of the reachable line on the upper boundary of the working area, is calculated as $x_1 = (R_y + W) \tan \hat{\theta}_1$. Hence α_l can be represented by the robot parameters as

$$\alpha_l = -\arctan\left(\frac{R_y}{2(R_y + W) \tan \hat{\theta}_1}\right) \quad (4)$$

Likewise, α_r is calculated as

$$\alpha_r = \arctan\left(\frac{R_y}{2x_2}\right) = \arctan\left(\frac{R_y}{2W \tan \hat{\theta}_1}\right) \quad (5)$$

It should be noted from Figure 6(a) that both equations (4) and (5) are obtained for the case of $-\bar{\theta}_1 < \hat{\theta}_1 < 0$. But, the case of $0 < \hat{\theta}_1 < \bar{\theta}_1$ is symmetric with respect to the former case and its kinematic constraint can be easily resolved as below:

$$\alpha_l = -\arctan\left(\frac{R_y}{2W \tan \hat{\theta}_1}\right) \quad (0 < \hat{\theta}_1 < \bar{\theta}_1)$$

$$\alpha_r = \arctan\left(\frac{R_y}{2(R_y + W) \tan \hat{\theta}_1}\right)$$

$$ii) \quad |\hat{\theta}_1| \geq |\bar{\theta}_1|$$

The reachable line of the failed leg with a locked angle in the above range ends at the side boundary of the working area. From Figure 6(b), the values of α_1 and α_r are described as

$$\alpha_1 = -\arctan\left(\frac{R_y}{2x_3}\right)$$

$$\alpha_r = \arctan\left(\frac{y_1 - (R_y/2 + W)}{R_x/2}\right)$$

Since x_3 and y_1 are obtained from trigonometry as $x_3 = W \tan \hat{\theta}_1$ and $y_1 = (R_x/2) \tan \hat{\theta}_1$, α_1 and α_r are reduced to

$$\alpha_1 = -\arctan\left(\frac{R_y}{2W \tan \hat{\theta}_1}\right) \quad (\hat{\theta}_1 \geq \bar{\theta}_1)$$

$$\alpha_r = \arctan\left(\frac{R_x - (R_y + 2W) \tan \hat{\theta}_1}{R_x \tan \hat{\theta}_1}\right)$$

Like the former case, the values of α_1 and α_r for the case of $\hat{\theta}_1 \leq -\bar{\theta}_1$ can be obtained by symmetry:

$$\alpha_1 = -\arctan\left(\frac{R_x - (R_y + 2W) \tan \hat{\theta}_1}{R_x \tan \hat{\theta}_1}\right) \quad (\hat{\theta}_1 \leq -\bar{\theta}_1)$$

$$\alpha_r = \arctan\left(\frac{R_y}{2W \tan \hat{\theta}_1}\right)$$

3.2.2 Failure of joint two

Figure 7 illustrates the kinematic constraint for fault-tolerant crab gaits when joint two of a leg is locked. Because a locked failure at joint three yields the same kinematic characteristics as that of joint two, its analysis is omitted in this paper. As shown in the figure, the reachable region of the foothold position becomes an arc with the radius r . The kinematic constraint is determined according to the length of r .

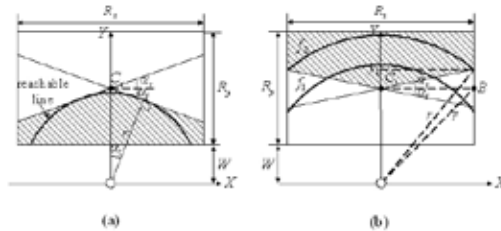


Fig. 7. Kinematic constraint for the locked failure in crab walking at joint two: (a) $r < R_y/2 + W$ and (b) $r \geq R_y/2 + W$.

i) $r < R_y/2 + W$

If r is in the above range, the reachable arc cannot enclose the center point C_i of the working area. Hence, for the foot trajectory to intersect the arc, the crab angle should be within the shaded area marked in Figure 7(a). The kinematic constraint for the crab angle α is thus

$$\alpha_r \leq \alpha \leq \alpha_l \quad (6)$$

From Figure 7(a), α_l is obtained as

$$\alpha_l = -\arccos\left(\frac{r}{R_y/2+W}\right)$$

Since r is the length of the leg projection onto the working area,

$$r = l_1 \cos \hat{\theta}_2 + l_2 \cos \hat{\theta}_3$$

where $\hat{\theta}_2$ is the locked angle of joint two and $\hat{\theta}_3$ is the corresponding fixed angle of joint three (refer to Figure 3). Applying the above equation, α_l is re-written as

$$\alpha_l = -\arccos\left(\frac{l_1 \cos \hat{\theta}_2 + l_2 \cos \hat{\theta}_3}{R_y/2+W}\right) \quad (7)$$

As the reachable arc is symmetric with respect to the Y_i axis of the working area, α_r is equal to $-\alpha_l$:

$$\alpha_r = \arccos\left(\frac{l_1 \cos \hat{\theta}_2 + l_2 \cos \hat{\theta}_3}{R_y/2+W}\right) \quad (8)$$

Substituting (7) and (8) into (6) completes the kinematic condition for the crab angle in the case of $r < R_y/2+W$.

$$ii) R_y/2+W \leq r < \bar{r}$$

In Figure 7(b), \bar{r} denotes the distance between the leg attachment point and the point B , the middle point of the right boundary of the working area. The case of $R_y/2+W \leq r < \bar{r}$ is observed most commonly in the failure situation, where the hexapod robot can walk with any crab angle since there exists always the intersection point between the reachable arc and the foot trajectory. f_1 in Figure 7(b) is an example of such a reachable arc which lies in this range.

$$iii) \bar{r} \leq r$$

If r is greater than or equal to \bar{r} , the reachable arc is drawn, for example, like f_2 in Figure 7(b). Thus the foot trajectory guaranteeing the existence of the fault-tolerant crab gait should be in the shaded area in Figure 7(b) and its kinematic constraint is expressed the same as (6). α_r is found from Figure 7() to be

$$\alpha_r = \arctan\left(\frac{y_1 - (R_y/2+W)}{R_x/2}\right) \quad (9)$$

where is reduced to

$$y_1 = \sqrt{r^2 - (R_x/2)^2} = \sqrt{(l_1 \cos \hat{\theta}_2 + l_2 \cos \hat{\theta}_3)^2 - (R_x/2)^2} \quad (10)$$

Like the former case, α_l is equal to α_r except the minus sign. Hence, from (9) and (10), we have

$$\alpha_l = -\arctan\left(\frac{\sqrt{4(l_1 \cos \hat{\theta}_2 + l_2 \cos \hat{\theta}_3)^2 - R_x^2} - (R_y / 2 + W)}{R_x}\right) \quad (\bar{r} \leq r)$$

$$\alpha_r = \arctan\left(\frac{\sqrt{4(l_1 \cos \hat{\theta}_2 + l_2 \cos \hat{\theta}_3)^2 - R_x^2} - (R_y / 2 + W)}{R_x}\right)$$

4. Fault-Tolerant Gaits: Straight-Line Motion

4.1 General Scheme

In the previous research, we have described constrained motions of a leg when a locked joint failure occurs to each of the three joints, respectively, and proposed the following principles of fault-tolerant gait planning:

Principles of Fault-Tolerant Gait Planning for a Locked Joint Failure (Yang, 2002)

- a) When the failed leg is in the support phase, the robot body does not translate because the failed leg cannot maintain the current foothold position.
- b) When the failed leg is in the transfer phase, it does not have active swing with respect to the body motion and is moved only passively by the translation of the body.

Let us examine the meaning of the above principles. When the robot body translates in legged locomotion, the configurations of all the supporting legs should be simultaneously changed to maintain the current foothold positions. If a locked failure occurs to a joint of a leg, however, the rank of Jacobian of the failed leg is reduced by one (Roberts & Maciejewski, 1996) and there exists no solution space of the inverse kinematics with which the foothold position of the failed leg remains the same against the translation of the body. Thus, the robot body should not translate when the failed leg is in the support phase (Principle a). Furthermore, the forward movement of a leg with respect to the robot body is inexecutable with a locked joint. For instance, if joint one of a leg is locked from failure, the failed leg cannot take active swing with respect to the robot body (see Figure 3(a)). On the other hand, if joint two or three is locked from failure, the accessible area where the failed leg could place its foot is reduced to only a partial region of the working area (see Figure 3(b) and (c)), which forbids normal straight-line walking with continuous leg swing. Therefore, it is prescribed that the failed leg is moved only passively by the translation of the body (Principle b).

4.2 Periodic Walking

In terms of mobility, it would be more advantageous that the robot walking with a periodic gait before a failure could preserve its periodicity (with a different gait pattern) even after a fault event. Based on the principles of fault-tolerant gait planning, we propose fault-tolerant periodic gaits for hexapod robots. Among the periodic gaits hexapod robots can have in static walking, quadruped gaits and tripod gaits are proposed in this section. The generation procedure of pentaped gaits, in which five legs are always in the support phase during walking, is similar to the other cases and omitted.

4.2.1 Quadruped gait

In quadruped gaits, two legs are always in the transfer phase. Figure 8 shows the algorithm of the fault-tolerant quadruped gait. Since a pair of legs is simultaneously lifted off and placed, the quadruped gait resulting from the algorithm belongs to a singular gait. Provided

that there is no timing gap between movements of two pairs of legs, the number of supporting legs always remains four. There are three movements of a pair of legs during the cycle time and the robot body advances only after a failed leg is lifted off. According to the principles of fault-tolerant gait planning, the final pair of leg in the transfer phase, including a failed leg, does not take active swing while the robot body moves.

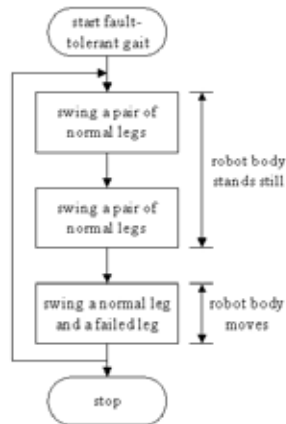


Fig. 8. Algorithm of the fault-tolerant quadruped gait.

Let us determine the leg sequence and the stride length formula based on the algorithm in Figure 8. The leg sequence with which a quadruped gait can have the maximum longitudinal stability margin (Song & Waldron, 1987) in straight-line walking is known to be (2,5)-(3,4)-(1,6) when the hexapod robot has the prototype shown in Figure 1 (Yang, 1999). In other words, lifting two legs in the symmetric positions with respect to the center of gravity guarantees the maximum stability to the resulting support pattern. The leg stroke, the distance through which the foot is translated relative to the body during the support phase (Song & Waldron, 1987), should be equal to the stride length if the gait is periodical. Therefore, if the stride length of the robot body is chosen, the leg stroke is determined equally.

Figure 9 illustrates the proposed quadruped gait with stride length λ for a locked joint failure at leg 1. Quadruped gaits for a failure at other legs can be derived by symmetry of the hexapod. Black circles denote foothold positions of supporting legs and white circles denote the previous locations of foothold positions. Dash-dotted rectangles are support patterns obtained from lifting two legs. Without loss of generality, the periodic gait sequence is supposed to start with all the six legs set to be in the support phase on the foothold positions in Figure 9(a), where normal legs are placed $(R_x - \lambda)/2$ in front of their rear-end positions and the failed leg lands on a point with distance x_1 ($0 \leq x_1 \leq R_x$) from its rear-end position. Leg 2 and leg 5 are lifted off in the first and moved as much as λ in the +X direction (Figure 9(b)). Next, leg 3 and leg 4 are lifted off and moved the same distance (Figure 9(c)). Because leg 1 is in the support phase, the robot body should not move in these states. The fault tolerance is realized in Figure 9(d) where leg 1 is lifted off with leg 6 and is transferred forward by the movement of the robot body. It is noted that working areas of the initial state are drawn fixed in Figure 9(d) for the convenience of illustration. In fact, the real locations of

the areas change as the body moves. By closing the behavior in Figure 9(d), one period of the leg sequence is accomplished. The longitudinal stability margin of this gait sequence is determined by the shortest of S_{25} and S_{16} , which are the longitudinal stability margins of the states obtained from lifting (leg 2, leg 5) and (leg 1, leg 6), respectively (see Figure 9(b) and (d)). S_{25} and S_{16} are calculated as

$$S_{25} = \frac{1}{4}(2x_1 + R_x - \lambda)$$

$$S_{16} = \frac{1}{2}(R_x - \lambda)$$

Therefore, longitudinal stability margin $S(\lambda)$ of the quadruped gait with stride length λ is derived as

$$S(\lambda) = \min\left[\frac{1}{4}(2x_1 + R_x - \lambda), \frac{1}{2}(R_x - \lambda)\right]$$

Restating the above equation according to the range of x_1 leads to

$$S(\lambda) = \begin{cases} (2x_1 + R_x - \lambda)/4, & x_1 < (R_x - \lambda)/2 \\ (R_x - \lambda)/2, & x_1 \geq (R_x - \lambda)/2 \end{cases} \quad (11)$$

This result implies that the stability of the fault-tolerant quadruped gait increases as the foothold position of the failed leg is more distant from the rear-end position in its working area. Because the failed leg cannot have active swing, the longest stroke which can be obtained in fault-tolerant periodic gaits is equal to R_x , the length of the working area (Yang, 2002). Thus the range of the stride length should be $0 < \lambda \leq R_x$. If λ is equal to R_x , $S(\lambda)$ becomes zero from (11). This complies with a corollary in gait study that a periodic gait with its maximum stride length has the marginal stability margin.

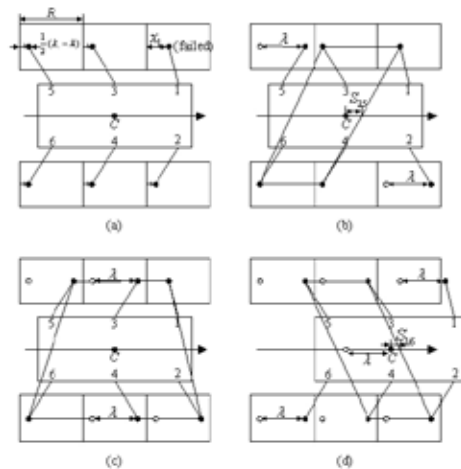


Fig. 9. The periodic quadruped gait for a locked failure at leg 1: (a) initial state, (b) swing leg 2 and leg 5, (c) swing leg 3 and leg 4, and (d) swing leg 1 and leg 6 with the body movement.

4.2.2 Tripod gait

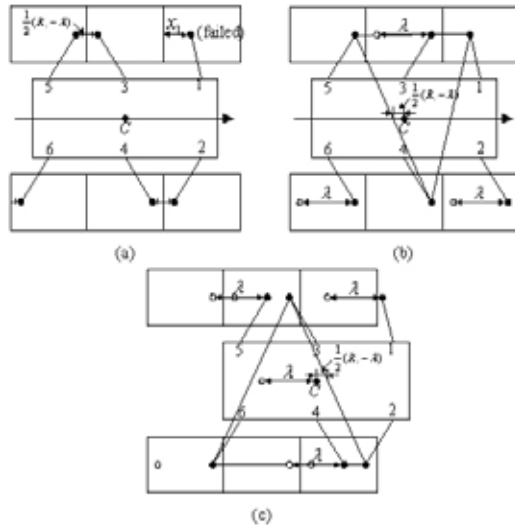


Fig. 10. The periodic tripod gait for a locked failure at leg 1: (a) initial state, (b) swing legs 2, 3 and 6, and (c) swing legs 1, 4 and 5 with the body movement.

The tripod gait has been adopted as the standard gait for hexapod robots (Miao & Howard, 2000) and is regarded as the fastest gait with the minimum stability. Figure 10 shows the tripod gait with stride length λ for a locked joint failure at leg 1. Like the quadruped gait, all the legs are supposed to be in the support phase with the foothold positions given in Figure 10(a). The leg sequence is (2, 3, 6)-(1, 4, 5), the same as that of the standard tripod gait on even terrain (Lee et. al, 1988), but, from the principles of fault-tolerant gait planning, the robot body has discontinuous movement.

The stability margin of the tripod gait is determined only by the stride length, regardless of the foothold position of the failed leg. This is because when the failed leg (leg 1) is in the support phase, the boundary of the support pattern made of the two normal legs (leg 4 and leg 5) has always the shortest distance from the vertical projection of the center of gravity (Figure 10(b)). Referring to Figure 10(b) and (c), the longitudinal stability margin is derived as

$$S(\lambda) = \frac{1}{2}(R_x - \lambda) \tag{12}$$

5. Fault-Tolerant Gaits: Crab Walking

Based on the general algorithm of fault tolerance for straight-line motion, the fault-tolerant periodic crab gait is proposed in this section. For the sake of simplicity, only the procedure of tripod crab gaits is addressed while the quadruped crab gait is omitted.

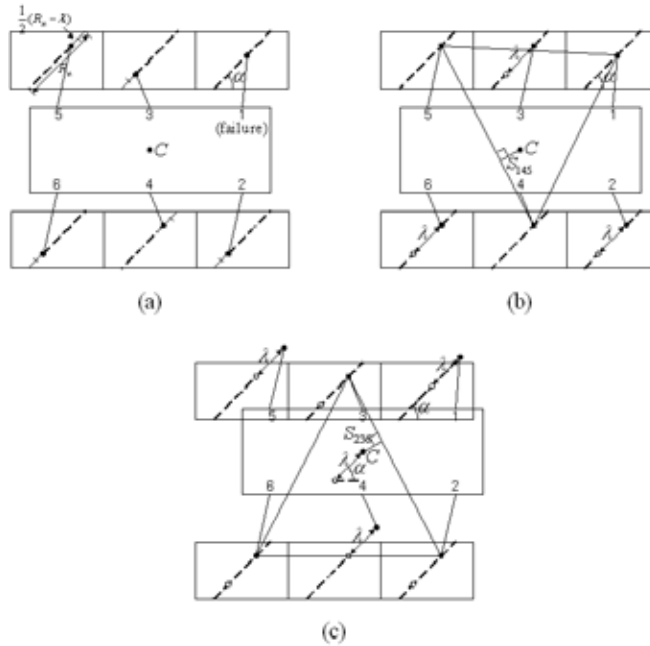


Fig. 11. Fault-tolerant periodic crab gait: (a) initial state, (b) swing legs 2, 3 and 6 and (c) swing legs 1, 4 and 5 with body moving.

Figure 11 shows the proposed fault-tolerant tripod crab gait where a locked joint failure occurs to leg 1. Tripod crab gaits for a failure at other legs can be derived by symmetry of the hexapod. It is noted that the crab angle α should satisfy the kinematic constraints derived in the previous section. Figure 11(a) is the initial state. Legs 2, 3 and 6, the tripod consisting of normal legs, are placed $(R_\alpha - \lambda)/2$ in front of their rear boundaries along the foot trajectory, where R_α is the length of the foot trajectory with the crab angle α with the following value:

$$R_\alpha = \begin{cases} R_y / \sin \alpha & \alpha_d < \alpha \leq 90^\circ \\ R_y / \cos \alpha & 0 \leq \alpha \leq \alpha_d \end{cases}$$

Because the failed leg does not have active swing, the stride length λ cannot be greater than R_α and lies in the range of $0 < \lambda \leq R_\alpha$. Leg 4 and leg 5, the two normal legs of another tripod, are placed $(R_\alpha - \lambda)/2$ behind their front boundaries along the foot trajectory in the initial state, and leg 1, the failed leg, is on the foothold position given by the failure configuration (see Figures 6 and 7). According to the leg sequence of the standard tripod gait, the tripod (2, 3, 6) is lifted off first in Figure 11(b) and moved as much as λ , while the robot body stands still. Then, another tripod (1, 4, 5) is lifted off and moved passively

with the translation of the body by λ in Figure 11(c), which completes one cycle of the gait. Dash-dotted triangles in Figure 11(b) and (c) are support patterns obtained when a tripod is in the transfer phase. Since the gait stability of a periodic gait is defined as the minimum value of the stability margin in one cycle, the gait stability of the crab gait with the crab angle α and stride length λ , denoted by $S(\alpha, \lambda)$, is determined as the shortest of S_{145} (Figure 11(b)) and S_{236} (Figure 11(c)). But S_{145} and S_{236} are found to be the same value and are derived as the following:

$$S(\alpha, \lambda) = \left(\frac{R_x}{2} - \frac{\lambda}{2} \left(\cos\alpha + \frac{R_x}{R_x + 2W + 2U} \sin\alpha \right) \right) \frac{R_y + 2W + 2U}{\sqrt{(R_y + 2W + 2U)^2 + R_x^2}} \quad (13)$$

If $\alpha = 0^\circ$ and the gait has the longest stride length, i.e., $\lambda = R_0 = R_x$, then $S(0^\circ, R_x) = 0$ in the above equation. This result coincides with that of the fault-tolerant tripod gait with straight-line motion (refer to Eq. (12)). Also, note that $S(\alpha, \lambda)$ is always a positive value if $\alpha > 0^\circ$. This means that the fault-tolerant crab gait is advantageous over the straight-line motion in terms of the gait stability.

If there is no time lag between the states of Figure 11(a) ~ (c), the fault-tolerant crab gait has duty factor $1/2$, the same as the gait with straight-line motion. This implies that the hexapod robot maintains its mobility as much as the straight-line motion against a locked failure even though it walks with a non-zero crab angle. In addition, the gait stability (13) is irrelevant to the foothold position of the failed leg. Hence the proposed crab gait promises successful post-failure walking for any locked failure.

6. Post-Failure Walking Example

In most cases, a gait of the hexapod robot at the moment of a locked failure may not belong to any state of the proposed periodic gaits. To change the present gait into a state of the proposed periodic gaits, some pre-adjustment of the leg position and the body movement are necessary. This issue will be discussed as a case study in this section.

6.1 Normal Walking: Wave Gait

In this section, the proposed gait planning is applied to a post-failure walking problem of a hexapod robot. We assume that the hexapod robot has been moving with the wave gait and duty factor $\beta = 2/3$ before a locked joint failure occurs to leg 1. The wave gait is a standard periodic gait of straight-line motion and mostly well known in the area of level-going (Song & Choi, 1990). Its behavior can be analyzed using the gait diagram and the stationary gait pattern. Figures 12 and 13 are the gait diagram and the stationary gait pattern of the wave gait with $\beta = 2/3$, respectively. A darkened solid line in Figure 12 represents the support phase of a leg and is cast on the leg stroke in the stationary gait pattern in Figure 13. Each leg stroke is then divided into equal segments labeled according to the times of the corresponding argument in the gait diagram. The support pattern at any instant of time can be easily constructed by just connecting all the segments with the same label. For example, the support pattern immediately after leg 1 is lifted off can be obtained by connecting all the segments with label 4, except that of leg 2, since leg 2 has been already lifted. The support pattern is thus the dash-dotted rectangle in Figure 13.

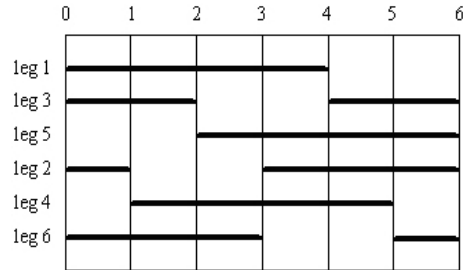


Fig. 12. Gait diagram of the wave gait with $\beta=2/3$.

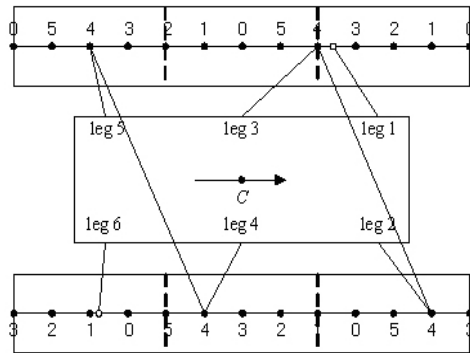


Fig. 13. The stationary gait pattern.

6.2 Post-Failure Walking

We investigate the adjustment procedure for the two cases, where the failed leg is in the support phase and the transfer phase at the moment of a locked joint failure, respectively. Let us denote T as the cycle time and t_f ($0 \leq t_f < T$) as the time by which the occurrence of a locked joint failure in leg 1 lags behind the contact of leg 1 on the ground. Since the wave gait with $\beta = 2/3$ is a type of quadruped gaits, we demand that post-failure walking should also be a quadruped gait. For the simplicity of analysis, it is assumed that the fault-tolerant gait in post-failure walking has the maximum stride length. Note that when $\lambda = R_x$, all the normal legs in the fault-tolerant quadruped gait are lifted off at the rear-end positions and placed on the front-end positions in one period (refer to Figure 9).

6.2.1 Failure in the support phase

If leg 1 fails in the support phase, t_f is in the range of $0 \leq t_f \leq 2T/3$. As two legs are always in the transfer phase, the gait transition procedure is made by the following steps:

- i) The robot body halts the movement at the instance of failure.

- ii) If leg 6 is in the transfer phase, it is placed on the rear-end position of its working area. Other legs in the transfer phase are placed on their front-end positions.
- iii) Find a state of the fault-tolerant quadruped gait to which the present state can be transformed with the minimum number of the change of the foothold positions, and complete the gait transition.

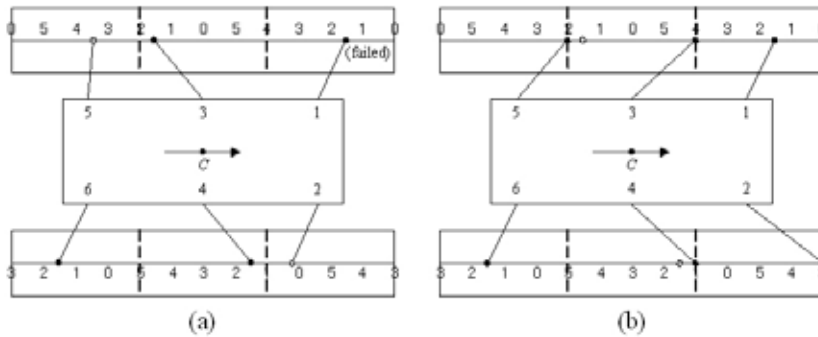


Fig. 14. Gait transition procedure when $\tau/6 < t_f < \tau/3$: (a) the state at the instance of failure, (b) place leg 2 and leg 5 and swing leg 3 and leg 4.

Recall that leg 1, the failed leg, is moved passively with leg 6 in the final state of the fault-tolerant quadruped gait in Figure 9. Thus if leg 6 is in the transfer phase, it should be placed on the rear-end position for the lift-off with leg 1. Other normal legs in the transfer phase should be placed on their front-end positions for minimizing required adjustments of the foothold position before starting fault-tolerant walking.

As a specific example of applying the above transition steps, let us assume that leg 1 fails in the range of $\tau/6 < t_f < \tau/3$. Figure 14 shows the gait transition procedure for this case. Leg 2 and leg 5, the two transferring legs at the moment of failure, are placed on their front-end positions in the first, followed by the swing of leg 3 and leg 4. The resulting state in Figure 14(b) is tantamount to the state where the failed leg is about to be lifted in the fault-tolerant periodic gait, i.e., Figure 9(c). By lifting leg 1 and leg 6 and moving the robot body after the state of Figure 9(b), the fault-tolerant quadruped gait can be initiated.

6.2.2 Failure in the transfer phase

If leg 1 fails in the transfer phase, t_f is in the range of $2\tau/3 \leq t_f < \tau$. In a similar manner to the case of the support phase, the robot body halts its movement at the moment of failure and two legs in the transfer phase find competent foothold positions. For example, the gait transition procedure when t_f is in the range of $2\tau/3 < t_f < 5\tau/6$ is illustrated by Figure 15.

Since leg 1 and leg 6 are in the transfer phase, the initial state is like Figure 15(a). Figure 15(b) shows the placement of the transferring legs. Leg 1, the failed leg, should be placed on a restricted position and leg 6 is placed on its rear-end position to be lifted off with leg 1

after the transition procedure. The foothold positions of the remaining normal legs are adjusted in Figure 15(c) and the hexapod is ready to start the fault-tolerant periodic gait. Note that both the transition procedures in Figures 14 and 15 are dead-lock free and guaranteed for any value of λ . They are also efficiently made in that the number of the change of the foothold positions is minimized.

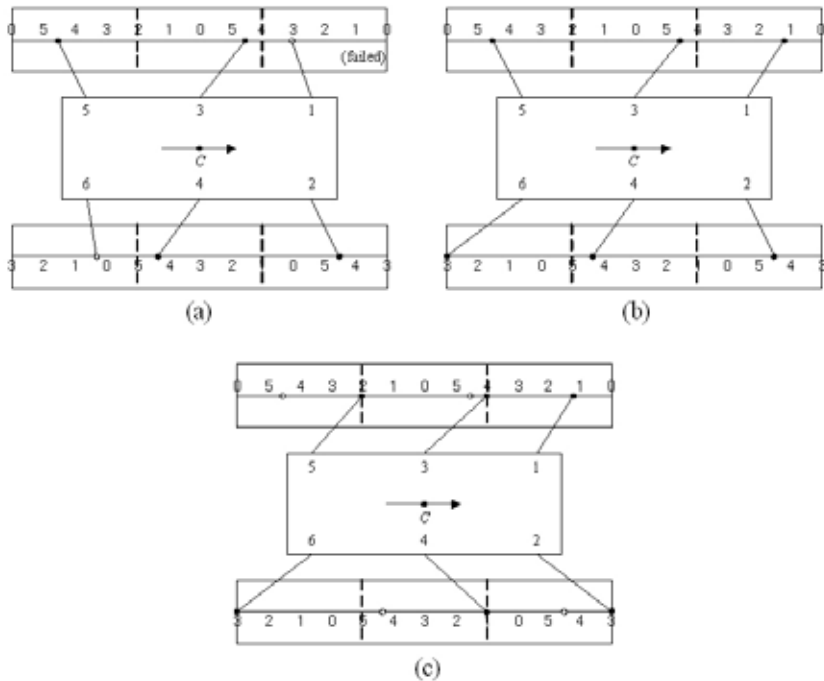


Fig. 15. Gait transition procedure when $2\pi/3 < t_f < 5\pi/6$: (a) the state at the instance of failure, (b) place leg 1 and leg 6, and (c) swing (leg 2, leg 5) and (leg 3, leg 4).

6.3 Comparison of Gait Performance

Let us examine the change of gait performance caused by the transition to the fault-tolerant gait. Figure 16 is the gait diagram of the proposed fault-tolerant quadruped gait in Figure 9. For a clear comparison, we assume that there is no resting stage between the placement of a pair of legs and the lift-off of another pair in the quadruped gait. From the gait diagram, the duty factor is obtained as $2/3$, the same as that of the wave gait. But the stride length of the fault-tolerant quadruped gait is R_x , less than $3R_x/2$ of the wave gait with $\beta = 2/3$ (Song & Choi, 1990). Therefore, it can be said that the proposed fault-tolerant gait has no loss of the duty factor against a locked joint failure and its performance is degenerated only in the stride length, which is an inevitable adverse effect of the locked joint failure.

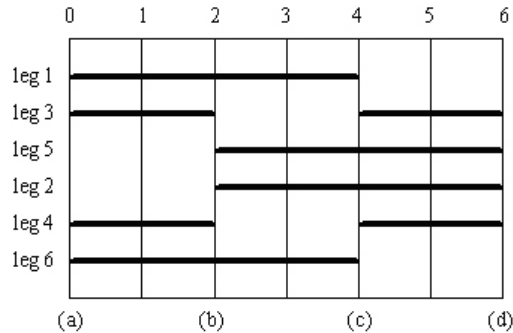


Fig. 16. Gait diagram of the fault-tolerant gait with straight-line motion.

7. Conclusion

In this article, gait planning for static walking of hexapod robots has been considered from a different point of view. The notion of a locked joint failure is introduced and its effect on robot walking is analyzed based on manipulator kinematics and gait study. A locked joint failure does not reduce stability of a gait but constrain the workspace of the failed leg to a restricted area. We have shown that there is a range of kinematic constraint on the configuration of the failed leg which guarantees the existence of post-failure walking on the straight-line and crab-walking trajectory, respectively. A strategy of fault tolerance for a locked joint failure has been proposed for the hexapod robot, in which the hexapod has discontinuous movement of the body with respect to leg swing and the failed leg is swung passively by the translation of the body. As a special form of the proposed strategy, periodic quadruped and tripod gaits have been proposed for straight-line motion and crab walking, respectively, and their behavior and efficiency have been investigated. By taking the proposed periodic gait, the hexapod can overcome any fault event caused by a locked joint failure and maintain static stability. The transition procedure from the standard wave gait to the proposed periodic gait has been shown as an example to demonstrate the applicability of the proposed scheme.

As further researches, there are fault-tolerant gait planning for irregular gaits on uneven terrain and fault-tolerant gaits considering dynamic effects.

8. References

- Chu, K. K. & Pang, K. H. (2002). Comparison between different model of hexapod robot in fault tolerant gait, *IEEE Transactions on Systems, Man, and Cybernetics A*, Vol. 32, No. 6, pp. 752-756.
- Craig, J. J. (2003). *Introduction to Robotics: Mechanics and Control*(3rd Ed.), Prentice Hall, ISBN 0201543613, Upper Saddle River, NJ.
- Lee, T. T.; Liao, C. M. & Chen, T. K. (1988). On the stability properties of hexapod tripod gait, *IEEE Transactions on Robotics and Automation*, Vol. 4, No. 4, pp. 427-434.
- Lee, Y. J. & Hirose, S. (2002). Three-legged walking for fault-tolerant locomotion of demining quadruped robots, *Advanced Robotics*, Vol. 16, No. 5, pp. 415-426.

- Lewis, C. L. & Maciejewski, A. A. (1997). Fault tolerant operation of kinematically redundant manipulators for locked joint failures, *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 4, pp. 622-629.
- Lewis, F. L.; Abdallah, C. T. & Dawson, D. M. (1993). *Control of Robot Manipulators*, Macmillan, ISBN 0-02-370501-9, New York.
- Miao, S. & Howard, D. (2000). Optimal tripod gait generation for hexapod walking machines, *Robotica*, Vol. 18, No. 6, pp. 639-649.
- Nagy, P. V.; Desa, S. & Whittaker, W. L. (1994). Energy based stability measures for reliable locomotion of statically stable walkers: theory and application, *International Journal of Robotics Research*, Vol. 3, No. 3, pp. 272-287.
- Roberts, R. G. & Maciejewski, A. A. (1996). A local measure of fault tolerance for kinematically redundant manipulators, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, pp. 543-552.
- Shin, J. H. & Lee, J. J. (1999). Fault detection and robust fault recovery control for robot manipulators with actuator failures, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 861-866, 1999.
- Song, S. M. & Choi, B. S. (1999). The optimally stable ranges of $2n$ -legged wave gaits, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 4, pp. 888-902.
- Song, S. M. & Waldron, K. J. (1987). An analytical approach for gait study and its application on wave gaits, *International Journal of Robotics Research*, Vol. 6, No. 2, pp. 60-70.
- Yang, J. M. & Kim, J. H. (1998). Fault-tolerant locomotion of the hexapod robot, *IEEE Transactions on Systems, Man, and Cybernetics B*, Vol. 28, No. 1, pp. 109-116.
- Yang, J. M. (1999). *Fault Tolerant Gaits of Legged Robots*, Ph.D. Dissertation, Korea Advance Institute of Science and Technology, Daejeon, Korea.
- Yang, J. M. (2002). Fault tolerant gaits of quadruped robots for locked joint failures, *IEEE Transactions on Systems, Man, and Cybernetics C*, Vol. 32, No. 4, pp. 507-516.
- Yang, J. M. (2003). Crab walking of quadruped robots with a locked joint failure, *Advanced Robotics*, Vol. 17, No. 9, pp. 863-878.



Edited by Jonas Buchli

This book covers many aspects of the exciting research in mobile robotics. It deals with different aspects of the control problem, especially also under uncertainty and faults. Mechanical design issues are discussed along with new sensor and actuator concepts. Games like soccer are a good example which comprise many of the aforementioned challenges in a single comprehensive and in the same time entertaining framework. Thus, the book comprises contributions dealing with aspects of the Robotcup competition.

Photo by Ociacia / iStock

IntechOpen

