IntechOpen

IntechOpen Book Series
Mobile Robotics, Volume 1

Applications of Mobile Robots

Edited by Efren Gorrostieta Hurtado

APPLICATIONS OF MOBILE ROBOTS

Edited by **Efren Gorrostieta Hurtado**

Applications of Mobile Robots
http://dx.doi.org/10.5772/intechopen.74181
Edited by Efren Gorrostieta Hurtado

Part of IntechOpen Book Series: Mobile Robotics, Volume 1
Book Series Editor: Efren Gorrostieta Hurtado

Contributors

Anthony Tzes, Sotiris Papatheodorou, Cristiano Premebida, Rares Ambrus, Zoltan-Csaba Marton, Mohammed A. H Ali, Musa Mailah, Zool H Ismail, Nohaidda Binti Sariff Sariff, Tao Song, Fengfeng Xi, Guo Shuai, Edgar Alonso Martínez García, Luz Abril Torres-Méndez, Felipe Nascimento Martins, Alexandre Brandão, Augusto Loureiro Da Costa, Diego Stéfano Fonseca Ferreira, Wagner Luiz Alves De Oliveira, Ebrahim A. Mattar, Ramon Barber, Jonathan Crespo, Clara Gomez, Alejandra C. Hernandez, Marina Galli

Notice

First published in London, United Kingdom, 2019 by IntechOpen
eBook (PDF) Published by IntechOpen, 2019
IntechOpen is the global imprint of INTECHOPEN LIMITED, registered in England and Wales, registration number: 11086078, The Shard, 25th floor, 32 London Bridge Street
London, SE19SG – United Kingdom
Printed in Croatia

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

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

Applications of Mobile Robots
Edited by Efren Gorrostieta Hurtado

p. cm.

Print ISBN 978-1-78985-755-9

Online ISBN 978-1-78985-756-6

eBook (PDF) ISBN 978-1-83962-086-7

ISSN 2632-5195

IntechOpen Book Series

Mobile Robotics

Volume 1

Efren Gorrostieta Hurtado has a Doctorate in Engineering with specialization in Mechatronics, and a Master of Science and Technology with specialization in Automation and Control. He is an electronics engineer with a specialty in control and biomedical engineering, and graduated in 1992 from the Technological Institute of Higher Studies of the West (ITESO), Guadalajara Jalisco, Mexico. He has worked as a professor at ITESO in the area of control, and at the National Technological Institute in the Morelia and Queretaro units, where he worked as a professor, coordinator, and head of the postgraduate unit. He also worked as a professor at La Salle Bajío University. He is currently a research professor at the Autonomous University of Querétaro in the Faculty of Engineering. He is the author of several articles and editor of books on robotics and automation. His current research interests are in the fields of control system mechatronics, robotics, and machine learning. He is a founding member of the Mexican Mechatronics Society.

Book Series Editor and Editor of Volume 1:
Efren Gorrostieta Hurtado
Engineering Faculty of the Autonomous University of Queretaro, Mexico

Scope of the Series

Mobile Robotics is a series devoted to applications of mobile robots and interaction of intelligent systems in different parts of the robot. The topics which this series is exploring are robot control, navigation system, intelligent systems, learning systems, deep learning systems, robot design, and applications in different environments. The series also welcomes research work in the area of mobile robotics development.

Contents

Preface

This book contains a selection of research work that focuses on the development of mobile robots. In this way we can discover the evolution of multi-agent robots in recent years. The investigations present alternatives to various problems such as a dynamic model based on speed for mobile robots with a differential drive, collaborative control schemes of a system for mobile ground robots, learning systems applied to mobile robots and the behavior of navigation related to interior environments, and the application of artificial intelligence algorithms for learning sensory data based on learned models. This book also presents a road mapping system and the extraction of features for navigation with mobile robots in roundabout environments and road monitoring, tracking routes of a mobile wheeled manipulator designed for manufacturing processes, systems where a Kalman filter estimator is extended for visual odometry, and finally the design of an on-chip system for the execution of cognitive agents.

We hope that the present research work will enrich and contribute to ideas and elements of interest for each of our readers.

I appreciate the professional work of each of the researchers who have contributed by sharing their work, developments, and ideas, which allowed us to present this book and share all this information with readers and the rest of the scientific community.

Efren Gorrostieta Hurtado, Dr Eng
Engineering Faculty of the Autonomous University of Queretaro, Mexico

Chapter 1

A Survey and Analysis of Cooperative Multi-Agent Robot Systems: Challenges and Directions

Zool Hilmi Ismail and Nohaidda Sariff

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79337

Abstract

Research in the area of cooperative multi-agent robot systems has received wide attention among researchers in recent years. The main concern is to find the effective coordination among autonomous agents to perform the task in order to achieve a high quality of overall performance. Therefore, this paper reviewed various selected literatures primarily from recent conference proceedings and journals related to cooperation and coordination of multi-agent robot systems (MARS). The problems, issues, and directions of MARS research have been investigated in the literature reviews. Three main elements of MARS which are the type of agents, control architectures, and communications were discussed thoroughly in the beginning of this paper. A series of problems together with the issues were analyzed and reviewed, which included centralized and decentralized control, consensus, containment, formation, task allocation, intelligences, optimization and communications of multi-agent robots. Since the research in the field of multi-agent robot research is expanding, some issues and future challenges in MARS are recalled, discussed and clarified with future directions. Finally, the paper is concluded with some recommendations with respect to multi-agent systems.

Keywords: cooperative mobile robots, multi-agent robot systems, coordination, control, communication

1. Introduction

Research on the multi-agent robot systems has been conducted since late 80s as it provides a more efficient and robust system compared to a single robot. ALLIANCE [1] and ACTRESS [2] robot are among of the earliest heterogeneous multi-agent robots developed by previous

researchers. The benefits received from information sharing among agents, data fusion, distribution of task, time and energy consumption have made the multi-agents research still relevant until present.

There were many researchers who focused on cooperative multi-agent research. The most challenging part was to provide a robust and intelligent control system so that the agents can communicate and coordinate among them to complete the task. Hence, it has been found that designing the control architecture, communication, and planning system were the major issues discussed and solved among researchers. Other than that, improvement to the existing coordination techniques, optimal control architectures, and communication were also the main highlights in the previous research. A few examples of cooperative multi-agent robots applications are soccer robot [3], unmanned guided vehicles (UGV's) and unmanned aerial vehicles (UAV's) [4], micro chain [5], and paralyzed robot [6].

There were two main reviewed papers proposed by Cao and Zhi Yan which were related to cooperative multi-agent research. Cao et al. [7] proposed a paper that represents the antecedents and direction of the cooperative mobile robot in the mid-1990s (most of the reviewed papers were published from 1990 to 1995). There were several issues discussed such as group architecture, resource conflict, the origin of cooperation, learning, and geometric problem. The applications and critical survey of the issues and direction of cooperative robots based on existing motivation have been indicated. Besides that, there were also a survey and an analysis of multi-robot coordination proposed by Yan et al. [8] in 2013 (most of the reviewed papers were published from 2000 to 2013). They presented a systematic survey and analysis of multiple mobile robot systems coordination. Related problems such as communication mechanism, a planning strategy, and a decision-making structure have been reviewed. In addition, various additional issues of cooperative MARS have been highlighted in these reviewed papers. Most of the papers were published from 2010 to 2015 which the recent research papers on cooperative multi-agent systems have been reviewed.

The main contributions of this paper are (i) the most reflected and affected key elements and current issues in cooperative mobile robots and (ii) directions and future challenges for the multi-agents robot, with recommendations and related suggestions. The remain sections of the paper are structured as follows: the first section discusses three main categories of multi-agent robot systems, the second section focuses on discussion of problems and some current issues of multi-agent systems and the final section is the conclusions with some challenges and recommendations for future research direction in the field of cooperative multi-agent systems.

2. Key elements of cooperative multi-agent robot systems

A wide means of research in cooperative multi-agent robots systems have focused on the three main elements which are (1) types of agents; homogeneous and heterogeneous, (2) control architectures; reactive, deliberative and hybrid, and (3) communication; implicit and explicit. In order to provide efficient coordination among multi-agent robots, the selections and designs of the control architecture and communication must possess a coherent behavior with

the agents. Therefore, this paper thoroughly explains each of the key elements with related examples from previous research and followed by the issues and directions of the multi-agent robot systems.

2.1. Types of agents: homogeneous and heterogeneous

Multi-agent robots can be divided into two categories which are homogeneous and heterogeneous. The agents become homogeneous when the physical structures or capabilities of the agents/individuals are identical (**Figure 1**). The capabilities for heterogeneous agents are not identical and they are different among robots, where each robot has its own specialization or specific task to complete [8]. Besides that, the physical structures of heterogeneous agents are also not identical among them (**Figures 2** and **3**).

Research carried out by Sugawara and Sano [9] and Hackwood and Beni [10] have proven that their homogeneous agents that have identical structures and identical capabilities can perform the task efficiently. However, for Li and Li [11], the heterogeneous agents are more applicable than homogeneous agents in the real world. Therefore, instead of focusing on homogeneous agents, current researchers are also concerned about heterogeneous agent's issues [1–6, 11–15]. The agent's physical structures and capabilities which are not identical have made the agents fall into these heterogeneous agents categories [16, 17].

There are two researchers known as Parker [18] and Goldberg [19] who compared the task coverage and interference between homogeneous and heterogeneous agents. Parker discovered that the task coverage for homogeneous agents is maximum compared to heterogeneous. This is because the homogeneous agents execute the same task at one time, while the heterogeneous agents need to distribute their task to another agent during the execution. Due to the task distributions among heterogeneous agents, the interference becomes higher compared to homogeneous agents, as proven in Goldberg's research [19]. As a result, we can summarize that the selection of a homogeneous or heterogeneous agent depends on the research application. Since the capability of heterogeneous agents is not identical, it becomes a challenging issue especially in finding consensus among agent during execution of the task. **Table 1** shows the research conducted by previous researchers using their heterogeneous agents.

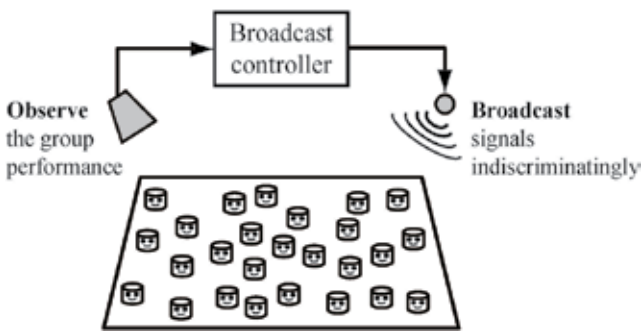


Figure 1. Multi-agent (homogeneous agents) control with broadcast [33].

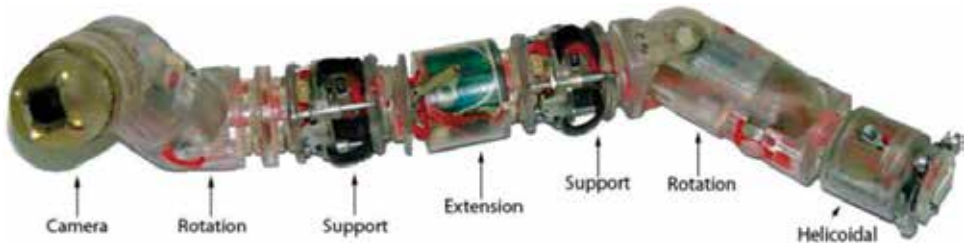


Figure 2. Examples of Heterogeneous agent (chained micro robots) [5].

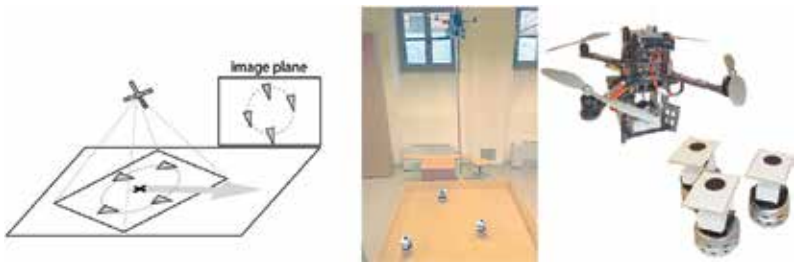


Figure 3. The heterogeneous team includes a single UAV (Pelican Quadrotor) controlling many UGV's (Khepera II robots) [4].

2.2. Control architectures: reactive, deliberative, and hybrid

The selection of control architectures for multi-agent robots is based on the capabilities of each agent to work in the groups and it also depends on how the overall systems work. The control architectures can be classified into three categories which are (i) reactive, (ii) deliberative and (iii) hybrid (reactive and deliberative).

Reactive control is also known as decentralized control. Reactive control relies on the concept of perception-reaction where the agents will cooperate between agents based on direct perception, signal broadcast or indirect communication via environmental changes. It does not require a high-level of communication to interact with agents. There are a few approaches which are related to reactive control for multi-agent robots. Glorennec [20] coordinated multi-agent robots by using fuzzy logic techniques to avoid obstacles and robots in the environment, whereas, research done by Lope et al. [12] coordinated their multi-agent robots by using the reinforcement learning algorithm based on the learning automata and ant colony optimization theory. Their multi-agent robots can organize the task by themselves to choose any task to be executed. It was proven that without interference from the central controller, the robots are capable of selecting their own task independently.

Despite these approaches, Chen and Sun [21] proposed a new optimal control law as distributed control for multi-agent robots in finding consensus to avoid obstacles in the environment. Local information from neighbors is required in this research. It is proven that this approach is capable of solving consensus problem under obstacle avoidance scenarios. In terms of local information context, Vatankhah et al. [22] developed a unique adaptive controller to move

Robot task	Type of robots	Reason of heterogeneous
The pusher robots work among them and push the paralyzed robot to a certain point. The paralyzed robot is driven by the global system [6]	One paralyzed robot with multiple numbers of pusher robots	Different robot, different task
The ROBOCUP robot team plays a soccer ball [3]	Group of agents (robots) acts as a goal keeper, middle field player, striker, and defender	Same robot, different task for each group of agents since each agent has different capabilities and characteristics
Unmanned aerial vehicles (UAV) acts as a supervisor to control unmanned ground vehicles (UGV) robots from any danger and collide with obstacles [4]	Single UAV flies to control and allocate several UGV'S	Different robot, different task
Coordination of heterogeneous multi-agents systems. Second order dynamics is the state of the leader while first order dynamics is the state of the followers [11]	Consists of leader and few followers	Same agent, different state dimension among the leader and follower (not identical)
Coordination of the micro robots chain [5]	Consists of different modules (active and passive) such as rotation, support, extension and helicoidally modules	Different modules, different task/ function
Multi-agent robots construct four different blocks [12]	Multiple agents	Same agent, different task
ACTRESS robot pushes the objects [2]	3 different robotors act as interface human operator, image processor, and global environment manager	Different agent, different task
ALLIANCE robot executes few tasks. The tasks are box pushing, puck gathering, marching, information, marching, hazardous and waste cleanup [1]	Small to a medium size of heterogeneous teams	Different agent, different task

Table 1. Examples of heterogeneous agent's research.

the leader and follower to a specific path. Finally, the decentralized control for stabilizing nonlinear multi-agent systems by using neural inverse optimal paper is carried out by Franco et al. [23].

Deliberative approach relied on the high-level communication, rich sensor and complete representation of the environment which allow the planning action. This approach is also known as centralized approach. The input data (usually from the static environment) that represents the global map can be planned to drive the agents efficiently to the target point [6, 24, 25]. The hybrid approach represents the integration control between reactive and deliberative control. Both controls complement each other to find the robust control system in controlling multi-agents robot. In deliberative control, all of the planning processes are involved with the calculation of a global target. As for reactive control, it is more towards a local plan for the robot to avoid the obstacles. There are examples of hybrid approaches related to multi-agents research studies as shown in **Table 2** [5, 6, 24, 25].

Task	Deliberative (D)	Reactive (R)	Communication of D and R
Pusher robots push the paralyzed robot to a specified target point [6]	Emit an attractive signal to move paralyzed robot to a specific target and to recruit another pusher robot to push the paralyzed robots (broadcast simple signal). It has a vision of the environment to determine the path	A force field approach used to define the pushers robots motion	D broadcast emitted signal to R controller
Solving dynamic problem for multi-agents by proposing a novel control scheme [25]	Introduce supervisor that assists a group of agents with centralized coverage control law and global trajectory tracking control law	Introduce control laws for coverage agents to avoid a collision and maintain proximity to a supervisor	Using a control law. Each law is active at a given time
Movement of chain micro-robots [5]	High layer for central control	Low-level embedded layer based on behavior function	D and R communicate using command exchange protocol

Table 2. Hybrid control architectures of multi-agent robots.

Every researcher has used different types of control architecture that are suitable for their system. They have come out with their own idea about the control architectures. Based on [26], hybrid architectures offer the most widespread solution in controlling intelligent mobile robots. Besides that, in a real world, agents also require acting in a dynamic and uncertain environment [6]. Subsequently, the hybrid approach allows the robot to navigate the target as well as avoiding the obstacles successfully within that environment [24].

The researchers who have focused on reactive architectures or known as decentralized approach [27] have claimed that decentralization will provide flexibility and robustness. However, Franco et al. [23] have different views where they agreed with the deliberative approach (centralized) is obviously good for their system although it is hard to control in a complex and large system due to technical and economic reasons. Sometimes, centralized control design totally depends on the system structure and it cannot handle structural changes. Once removed, it needs to be designed all over again. It is also costly and complex in terms of online computation and its control design.

2.3. Communications: implicit and explicit

Cooperation is usually based on some forms of communication. Communication is a mode of interactions between multi-agent robots. With an efficient communication system, the robot is capable of interacting, sharing and exchanging information. Communication also determines the success in mobile robots cooperation [28, 29]. Based on research by Cao et al. [7], there are three types of communication structures which are (i) interaction via the environment, (ii) interaction via sensing, and (iii) interaction via communications. However, this section will only focus on the two main types of interaction (ii) and (iii) which are important in the communication of mobile robots.

Implicit communication or also known as interaction via sensing refers to the local interactions between agents (agent to agent) as shown in **Table 3**. The agents will sense other agents

Task	Sensors
Multi-robots work together to avoid other robots, remove obstacles and pass objects [30]	Real mobile robots equipped with CCD cameras
Follower robots follow the leader while avoiding the obstacles [31]	4 robots equipped with ultrasound sensors
Robot teams will track the target and push the box cooperatively [32]	4 robots are equipped with side sensors. Different signals are emitted to differentiate between the robots (3 robots) and target robot (1 robot)
UAV allocate UGV's [4]	UAV equipped with a video camera with onboard Inertial Measurement Unit (IMU). UGV equipped with onboard laser range finder sensor
Multi-robot cooperatively collects the pucks in the field [9]	Robots are equipped with a pair of photo sensors and a pair of IR sensors

Table 3. Implicit communication researches.

by embedding a different kind of sensors among them. They will react to avoid obstacles among themselves if they sense signals from other agents [4, 10, 30–32]. However, due to limitation of hardware parts, the interaction via sensing has been replaced by using a radio or infrared communication.

Explicit communication refers to the direct exchange of information between agents or via broadcast messages. This often requires onboard communication modules. Issues on designing the network topologies and communication protocol arise because these types of communication are similar to the communication network [3, 5–6, 33–39]. **Table 4** shows an example of explicit communications being used in the robot systems.

Task	Communication devices/network	Interaction
Coordination of the ROBOCUP teams (middle size league) [3]	Robots equipped with communicating devices (off-the-shelf) radio modems and wireless Ethernet cards. Communication is based on underlying IP protocol either TCP-IP or UDP-IP	Agent to agent
Developing the control architectures for chain micro robots [5]	Command exchange protocol is used for communication between modules and PC by sending a message. The name of the protocol is I^2C protocol	One to many agents (modules) for global. Agent to agents for local
The pusher robots work cooperatively to push the paralyzed robot to a specific point [6]	PC will send messages to the Mindstorm robot (paralyzed) to control Mirosot robots (pusher) by using infrared serial communications interface/ transceiver	One way communication. One to all agents (broadcast) for global
Broadcast control framework for multi-agent coordination [33]	The broadcast signal sent from computer to all agents via Bluetooth	One way communication. One to all agents (broadcast)
Sign board based inter-robot communication in distributes robotic system [34]	Communication-based on conceptual mechanism of "sign-board" being used in inter-robot system	Agent to agent

Task	Communication devices/network	Interaction
Cooperative multi-robot system using Hello-Call Communication [35]	Each agent communicates together (chains) using "hello-call" protocol to extend their effective communication ranges	Agent to agent
Swarm robots control mobile robot using wireless sensor networks [36]	Using Wifi and three communication channels to interact between swarms for cooperation	One to many agents (broadcast)
Effect of grouping in local communication system of multiple mobile robots [37]	Information spread by the effect of random walk and local communication known as information diffusion (equation of acquisition probability)	Agent to groups of agents
A design method of local communication area in multiple mobile robots systems [38, 39]	Communication by information probability by infinite series	Agent to agent

Table 4. Explicit communication researches.

3. Problems and issues of cooperative multi-agent robot systems

Although researchers in recent years have addressed the issues of multi-agent robot systems (MARS), the current robot technology is still far from achieving many real world applications. Some real world MARS applications can be found in unmanned aerial vehicles (UAV's), unmanned ground vehicles (UGV's), unmanned underwater vehicles (UUV's), multi-robot surveillance, planetary exploration, search and rescue missions, service robots in smart homes and offices, warehouse management, as well as transportation. Therefore, in this paper, problems and issues related to cooperative multi-agent systems are discussed to improve the current approaches and to further expand the applications of MARS.

3.1. Centralized and distributed control

Based on Section 2.2, the differences between two types of control approaches have been highlighted. However, some problems and issues of both control system in coordinating multi agents will be discussed. By having centralized control, the global information of the environment has been used to calculate the path, trajectory or position of the agents before all [5, 6, 24–26, 40]. The information then can be sent directly to the agents by using a suitable communication medium. This is one advantage of this control where the agents can obtained the information directly from its central. Research by Azuma [33] shows that the central will sent the updated location directly to the agents by using a WIFI continuously until the agents reach the target point. The quadratic equation is used to calculate agent performances while Simultaneous Perturbation Stochastic Approximation is the algorithm used for the control design [41]. Besides that, A* algorithm, Dijkstra, Genetic Algorithm [42–44] and Ant Colony Optimization algorithm [45–47], are example of another algorithms have been used in multi agent centralized control. Oleiwi et al. [24] used a modified GA with A* algorithm for its global motion controller while Atinc et al. [25] proposed a novel control scheme that has a centralized coverage control law.

The main issue in centralized control exists when the number of agents is expanding. The computation will become high since there is only one centralized processor that control over all of

the system. Effect of this high computation, the time as well as the energy consumption will be effected at some point. Therefore, to solve this problem, hybrid control approach [5, 6, 24, 25] has been proposed with objective to balance between centralized control and distributed control [23, 26, 48–52]. Besides that, alternative towards optimizing or minimizing the trajectory length, time and energy consumption [24] as well as adding the intelligences [11, 20, 22, 27, 31, 32, 53] has taken into consideration to reduce the computation time. In terms of scalability, adaptability and flexibility of the controller can be claimed lesser as compared to distributed control. Any changes especially dealing with dynamics will cause the repetition in the computing and sometimes will effect overall of the system with only a limited number of controllers. Thus, centralized control sometimes does not fit with the dynamic environment.

Distributed control had proven scalable, adaptive, flexible and robust for multi agents system not only in static but also in a dynamic environment [54]. Many researchers had proven that their distributed controller can work efficiently for their multi agent robot systems [12, 26, 31, 32, 34, 48–50, 53, 55–58]. Innocenti et al. [27] have proven that their ActivMedia Pioneer 2DX mobile robots can reach its target by using their fuzzy logic controller. Same goes to Chen and Sun [21], Vatankhah et al. [22] and Glorennec [20] where they develop the distributed controller purposely for obstacles avoidance for their multi agents by using a fuzzy, neuro fuzzy, and a new optimal control protocol.

In distributed, the main issue is the task has to be distributed in a robust an efficient manner to ensure that every agent is able to perform its individual task cooperatively with another agents to achieve certain target. Distributing task among heterogeneous agents [11, 15] is more crucial and complex comparing with homogeneous agents which are identical [20–22, 59]. Limited sensing range and low bandwidth are also among physical constraints in distributed approach. With a limited local information, the agent cannot predict and cannot control the group behavior effectively in some sense. Another issues in distributed such as consensus, formation, containment, task allocation, optimization and intelligence will also discussed thoroughly in below section.

3.2. Consensus

Since multi-agent robots need to interact and communicate together to work cooperatively, issue on finding consensus for the homogeneous and heterogeneous robot has attracted researchers' attention over the past few years. Consensus refers to the degree of agreement among multi-agents to reach certain quantities of interest. The main problem of consensus control in multi-agent robots is to design a distributed protocol by using local information which can guarantee the agreements between robots to reach certain tasks or certain states. Therefore, a large number of interest concerning on developing the consensus control distributed protocol for homogeneous and heterogeneous robots which can be classified into a leader following consensus [60] and leaderless consensus [61–66], (to name a few), have been intensively studied by researchers recently [22, 67].

Each of heterogeneity agents is not identical and the states between agents are different which will cause difficulties in finding consensus. This is known as cooperative output consensus problem. This is a challenging issue for heterogeneous robots and there are a number of researchers who focused on the leaderless output consensus problem [13, 15, 68] and leader-follower output consensus problem [13, 15, 69–71]. Wieland et al. [68] proposed an internal

model principle to solve the leaderless output consensus problem for heterogeneous linear multi-agent systems. Wang et al. [69] discussed the classes of multi-agent system by switching topologies via static and dynamic feedback.

Research on finding consensus in the broadcasting area has also been carried out by few researchers. Li and Yan [72] solved the consensus in both fixing and switching type topology based on the spectrum radius of stochastic matrices. Azuma et al. [73] studied the consensus problem with a limited communication range and unlimited broadcast range by proposing its own controller. They introduced a concept of connected agent groups. This is to reduce consensus for "group to group" relation and for "agent to agent" relation in the groups by proposing two groups of consensus controller which are local and global. They proved that their controller can work efficiently in a mixed environment with communication and broadcast.

Besides that, research carried out by Das and Ghose [74, 75] solved the positional consensus problem for multi-agents. Das and Ghose [74] proposed a novel linear programming formulation and random perturbation input in the control command to achieve consensus at the prespecified location. The results showed that novel linear programming that is less intensive computation and perfect consensus can be obtained from random perturbation. They also proposed a novel linear programming formulation for their research [75]. Overall, it can be summarized that consensus problem is a vital issue which has been solved by many researchers. They have identified solutions to consensus problems for either homogeneous agents or heterogeneous agents which focus on finding an agreement among agents based on agent states (linear, nonlinear, static or dynamics topology) although there is an existence of leader in the environment or leaderless. Other than that, finding consensus in broadcasting topology/communication, broadcast mixed environment [73] and positioning agents [74] are also another recent issues focused by previous researchers.

3.3. Containment

Containment control is another problem investigated by many researchers. Containment problem refers to introducing more than one leader among the agents to ensure the groups are not ventured by the hazardous environment. If the agents are faced with this situation, they will move the robots to the safe region spanned by a group of leaders. The agents can either be homogeneous agents that have identical dynamics or heterogeneous agents that have different dynamics.

There are several issues investigated by previous researchers to solve the containment control problem for multi-agent robots such as (i) containment problem for a different dynamic level of the leaders and followers [70, 71, 14], (ii) containment problem for a linear and nonlinear systems [50, 76–79], (iii) containment problem for first order and second order systems [43–44, 72]. By assuming the follower and the leader of heterogeneous agents have different dynamics but the dynamics between each follower are similar, Youcheng and Yiguang [80] and Yuanshi and Long [81] had carried out their research studies. Besides that, Haghshenas et al. [14] solved the containment problem for two followers when the dynamic level is not identical.

A research on solving the containment problem for linear and nonlinear systems has been carried out by few researchers. Ping and Wen [76] investigated the linear first order systems for their multiple leaders. The distributed finite time containment problem for linear systems

was also being explored by the authors [77]. For nonlinear systems, Liu et al. [50] investigated the distributed containment control problem for second order nonlinear multi-agents with dynamic leaders. The issues of containment problem for the first order and second order systems have been investigated by Ping and Wen [76], Bo et al. [51] and Rong et al. [52]. Ping and Wen [76] studied the first order multi-agent systems while Bo et al. [51] proposed the control protocol for first order discrete-time systems with fixed time delays.

3.4. Formation

Formation control is an important issue to coordinate and control a group of multi-agent robots [49, 82–84]. The robots must be able to control their relative position and orientation among the robots in a group to move to a specific point. The motivations that drive the most attention among researchers to this problem are the biological inspirations, challenging control problems and the demand of multi-robot systems. There are many issues needed to be considered in designing a controller for mobile robot formation such as the stability of the formation, controllability of different formation patterns, safety and uncertainties in formations [82]. Other than that, issues of formation shape generation, formation reconfiguration and selection, formation tracking as well as role assignments in formation is discussed by Kiattisin.

There are three main control strategies for formation control proposed by previous researchers [82, 85, 86] such as (i) behavior based [87], (ii) virtual structure [88], (iii) leader-follower [89]. Each formation control method has its advantages and disadvantages. Balch and Arkin [87] proposed behavior-based formation control for their multi-robot teams. Behavior-based approach refers to several desired behavior of the agents such as goal seeking, obstacles avoidance, collision avoidance, etc. The final robot decision to choose which behavior comes first is based on the average weight of the behavior. The advantage of this approach is it can be used to guide the multi-agent robots in the unknown or dynamic environment by using the local information that the robot has. However, the drawback is where it cannot guarantee to converge easily during the process.

Virtual structure is a formation control that considers the entire formation as a rigid body which was pioneered by Lewis and Tan [88]. The main advantage of this approach is easy coordination of the group's behavior and the formation is well maintain during maneuvers. However, the limitation of the virtual structures is it has to maintain the same virtual structure at all times especially when the formation shape needs to be frequently reconfigured. If not, the possible applications are limited. Leader and followers approach is another formation control for multi-agent robots proposed by previous researchers [85, 86, 89]. In this strategy, some robots are considered as leaders while others will act as followers. The leaders will lead the followers to the target path while the followers will position and orientate by themselves while following the leaders. The main advantage of this approach is it can reduce the tracking error while the disadvantages are it will lead to a poor disturbance rejection property and the leader's motion will not depend on the followers. In addition, the formation does not tolerate to the leader's faults.

The networking system in formation control is another challenging issue highlighted by Chen and Wang [82] and Kiattisin in their reviewed papers. The communication delay in inter-robot information flow and communication loss problem will affect the performance of formation

control and can even make the formation control system unstable. Therefore, a suitable communication protocol and network control system need to be implemented correctly into the robot system. In order to get more realistic formation control design for multi-agent robots coordination, the formation control needs to come together with an effective communication system design (either for local or global information via sensing or wireless network). Lastly, an alternative of implementing a hybrid control framework for multi-agent robots formation control has also become an issue to let the robots work in real world applications.

3.5. Task allocation

The problem of task allocation among multi-agent robots has attracted researcher's attention. Once the computer assigns the task, the task needs to be sent to the robots for execution. Thus, a suitable approach needs to be applied in the system to ensure that the task is successfully allocated to the robots. Sarker et al. [90] used attractive field model to self-organize their robots while allocating its task. On the other side, Tolmidis and Petrou [91] proposed multi-objective optimization for their dynamic task allocation. The experiment results show a scalability, a generic solution and a better utilization of time as well as energy. Nagarajan and Thondiyath [92] also provided their own algorithm for task allocation which had proven better performances and better in minimizing the turnaround time, makespan and also cost.

3.6. Intelligences

The intelligence of multiagent robots to work cooperatively or coordinate its task is based on its controller. The design of the controller will determine agent's performances. The evolution of MARS shows that the level of intelligence is increasing in proportional with the technology. Since the beginning of artificial intelligences has been introduced, many researchers have started to design their controller by using this artificial intelligences approaches.

There are several approaches of artificial intelligences have been used by researchers in their multi agent controller development [11, 20, 22, 27, 31, 32, 53]. Fuzzy logic and neural network are approach have been used in multi agent robot control design which already proven its robustness and effectiveness [93]. Al-Jarrah et al. [31] used 2 fuzzy levels, which consisted of a fuzzy probabilistic control and adaptive neuro-fuzzy inference system, ANFIS. Vatankhah et al. [22] proposed a neuro-fuzzy structure with critic based learning structure and [11] proposed iterative learning control (ILC) scheme for their control system. Another researchers [20, 27, 32, 53, 94–97] were also using fuzzy control as one of the artificial intelligence approaches to develop their robots controller.

Other than artificial intelligence, there is another kind of intelligence proposed by previous researchers that had proven their multi-agent robots work effectively and successfully. Instead of focusing to a basic learning method, Tosic and Vilalta [98] proposed a unified framework for their multi-agent coordination by adopting the reinforcement learning, co-learning, and meta-learning in their system. Leader and follower concept also has been applied by few researchers to coordinate and plan their agent path [11, 22, 31]. Broadcast concept and framework for multi agent coordination can also be considered as an alternative towards intelligence [33, 61, 73, 99]. Azuma et al. [33, 73] developed the controller and broadcasted the signal from "agent to agent" or "agent to all agents". They also proposed integral-type broadcast

controllers and provide a sufficient condition for the controller gain to stabilize the broadcast for their group of Markov Chains [99]. Seyboth et al. [61] proposed the novel control strategy known as event-based broadcast control. They proved that their controller is more effective as compared to the time-based broadcast control. Finally, by having the intelligence, multi agent robot control is ready to be apply for an advance and complex multi-agents applications [3, 36, 49, 59]. As an example, Jolly et al. [53] and Candea et al. [3] have proposed their own controller to let the soccer robots coordinate and play successfully.

3.7. Optimization

Optimization is one of the important issue in designing a control system for multi agent robots. The objective is to find an optimal strategy under a given cost function either to find optimum trajectory/path, time, speed an as well as energy consumption. For example, by minimizing the path, less time is taken by the agent to move to its target point and the energy consumption will become less also.

Kumar and Kothare [100] have discovered the optimal strategy and optimal control architectures for their swarm agents. Their aim was to stabilize a swarm of stochastic agents by proposing the novel broadcast stochastic receding horizon controller. In order to search for an optimal path trajectory by minimizing the trajectory, time and energy consumption, Oleiwi et al. [24] had proposed the optimal motion planner. They combined the modified genetic algorithm with A* algorithm to find a path from the start point to the goal point, fuzzy to avoid obstacles and cubic spline interpolation curve to reduce energy consumption.

However, Chen and Sun [21] had a different approach, where they had proposed a new optimal control protocol to find an optimal control for their multi-agent consensus. Nagarajan and Thondiyath [92] proposed an algorithm that can minimize the turnaround time and cost during the agent's allocation task. The result showed that the algorithm performed better than the existing algorithm.

3.8. Communications

Issues on communications either implicit or explicit type of communication has been tackled since it will give effect to the multi agent controller performances. There are researchers who have focused on implicit communication where their robots interact based on sensor signal embedded to the robots [4, 30, 31]. However, there are also some drawbacks of implicit communication such as (1) limitations of the hardware and the sensors i.e. the hardware cannot support too many sensors, and the sensors can only work at certain conditions and distances, and (2) time delay if too many agents need to pass the information from one to another. Therefore, explicit communication come in to place where the information (messages) can be sent via broadcast (one to all) [5, 6, 33] or one to one agent [3, 5, 34].

However, other challenging parts of explicit communication are (1) to design a control framework to send the messages efficiently [33], (2) to design a suitable protocol that can guarantee all agents communicate effectively in the environment [3, 5, 34, 35], (3) to solve consensus problem which occurs during the interaction process either for homogeneous or heterogeneous agents [3], and (4) to design optimal controller that can optimize the speed and energy of the robots [33]. With the aim of providing an effective communication system for the robot

coordination, researchers have tried to fix these problems by designing a suitable communication control that is relevant to the systems. There are also researchers who complement both communications implicitly and explicitly for their cooperative multi-agents research.

4. Conclusion

This paper has provided a review of cooperative multi-agent robots system (MARS). It shows that this research is leading to the creation of a robust cooperation and coordination of multi-agent robots in various real applications. In order to produce high performance among agents, improvement in controller and communication part is the most crucial issues highlighted by researchers. Thus we strongly believe that this research has a potential to be expanded as the technology develops and the cooperative agents are foreseen to produce a big contribution towards the applications. Improvement on the controller design and communications either by adding intelligences or optimize certain cost function is in parallel with the technologies development which will then produce a multi agents which are mobile, scalable, flexible, global, dynamic and persistent connectivity. Regardingly, the following are other future challenges and recommendations that could be explored by our future researchers in expanding the area of MARS.

4.1. Future challenges

There are many challenges for future cooperative multi-agent systems and, among them, the most crucial challenge lies in controller design, which should be robust and intelligent enough to support overall system. Besides that, communication among agents is also important since it will determine the success of the system. Therefore, there are several future challenges that should be taken into consideration:

1. The need of more powerful coordination among homogeneous and heterogeneous agents. This is especially for advance and complex multi-agent robots application such as soccer robots [3], swarm robots [36], UGV's [4], UAV's [4] or any other robots.
2. Since the physical identity and capability among heterogeneous agents are not identical, issues in coordinating the agents will become more challenging compared to homogeneous agents [2–5, 6, 9, 12]. Attention should be given more to these agents.
3. Adapting various artificial intelligence approaches in solving control and communication problems of MARS either consensus [13, 15, 69–71], containment [70, 71, 14, 49, 82–84], position [45, 58] or any other problems should be considered as long as there is an improvement towards the robot performances.
4. Issues in reducing the energy consumption and time travel will produce an optimal controller for the agents. Thus, an appropriate design of controller should be applied together with the suitable communication system that can support the MARS [3, 5].
5. By broadcasting the information to agents, the information can be sent directly to agents, to avoid losses and time delay during transmission of the information [33, 45, 47–48, 51, 52]. Thus, this research should be expanded since the communication among agents can be improved from time to time.

4.2. Recommendations

Some recommendations for cooperative MARS are as follows:

1. The reactive and deliberative control architectures have their own strengths and weaknesses. In the future, an effective way is to implement hybrid approach into MARS which consists of both reactive and deliberative control that leads to a more efficient system.
2. An effective interaction between multi-agent robots can be achieved by integrating the implicit and explicit communications especially when the number of agents is increasing.
3. A suitable communication protocol and network control system should be implemented into MARS to avoid time delay during transmission of information among agents.

Acknowledgements

The financial support by Malaysian Government and University Teknologi Malaysia for this research is gratefully acknowledged.

Conflicts of interest

Some works in this paper are based on study review from selected journals and proceedings regarding the cooperative multi-agent robot systems. All works from other researchers have been cited carefully.

Author details

Zool Hilmi Ismail and Nohaidda Sariff*

*Address all correspondence to: nohaiddasariff@yahoo.com

Centre for Artificial Intelligence & Robotics (CAIRO), Malaysia-Japan International Institute of Technology (MJIT), Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, Kuala Lumpur, Malaysia

References

[1] Parker LE. ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. IEEE Transactions on Robotics and Automation. 1998;**14**:220-240

[2] Asama H, Matsumoto A, Ishida Y. Design of an autonomous and distributed robot system: ACTRESS. In: IEEE/RSJ International Workshop on Intelligent Robots and Systems; 4-6 September 1989. 1989. pp. 283-290

[3] Candea C, Hu H, Iocchi L, Nardi D, Piaggio M. Coordination in multi agent RoboCup teams. Robotics and Autonomous Systems. 2001;**36**:67-86

[4] Rosa L, Cognetti M, Nicastro A, Alvarez P, Oriolo G. Multi task cooperative control in a heterogeneous ground air robot team. In: 3rd IFAC Workshop on Multivehicle Systems. Vol. 48. 2015. pp. 53-58

[5] Brunete A, Hernando M, Gambao E, Torres JE. A behavior based control architecture for heterogeneous modular, multi configurable, chained micro robots. Robotics and Autonomous Systems. 2012;**60**:1607-1624

[6] Simonin O, Grunder O. A cooperative multi robot architecture for moving a paralyzed robot. Mechatronics. 2009;**19**:463-470

[7] Cao YU, Fukunagu AS, Kahng AB. Cooperative mobile robotics: Antecedents and directions. Journal of Autonomous Robots. 1997;**4**:1-23

[8] Yan Z, Jouandeau N, Cherif AA. A survey and analysis of multi robot coordination. International Journal of Advanced Robotics Systems. 2013;**10**:1-18

[9] Sugawara K, Sano M. Cooperative acceleration of task performances: Foraging behavior of interacting multi robots system. Physica D. 1997;**100**:343-354

[10] Hackwood S, Beni G. Self organization of sensors for swarm intelligence. In: IEEE International Conference on Robotics and Automation. 1992. pp. 819-829

[11] Li J. Iterative learning control approach for a kind of Heterogeneous multi agent systems with distributed initial state learning. Applied Mathematics and Computation. 2015; **265**:1044-1057

[12] Lope JD, Maravall D, Quinonez Y. Self-organizing techniques to improve the decentralized multi task distribution in multi robot systems. Neurocomputing. 2015;**163**:47-55

[13] Ma Q, Miao G. Output consensus for heterogeneous multi agent systems with linear dynamics. Applied Mathematics and Computation. 2015;**271**:548-555

[14] Haghshenas H, Badamchizadeh MA, Baradarannia M. Containment control of heterogeneneous linear multi agent systems. Automatica. 2015;**54**:210-216

[15] Li Z, Duan Z, Lewis FL. Distributed robust consensus control of multi agent systems with heterogeneous matching uncertainties. Automatica. 2014;**50**:883-889

[16] Vlacic L, Engwirda A, Kajitani M. Cooperative behavior of intelligent agents: Theory and practice. In: Sinha NK, Gupta MM, Zadeh LA, editors. Soft Computing & Intelligent Systems. UK: Academic Press; 2000. pp. 279-307

[17] Oliveira E, Fischer K, Stepankova O. Multi Agent Systems: Which Research for Which Applications. Robotics and Autonomous Systems. 1999;**27**:91-106

[18] Parker LE. Heterogeneous Multi Robot Cooperation. MA, USA: Massachusetts Institute of Techology Cambridge; 1994

[19] Goldberg D. Heterogeneous and homogeneous robot group behavior. In: Proceedings AAAI-96. 1996. p. 1390

[20] Glorennec PY. Coordination between autonomous robots. International Journal of Approximate Reasoning. 1997;**17**:433-446

[21] Chen Y, Sun J. Distributed optimal control for multi agent systems with obstacles avoidance. Neurocomputing. 15 January 2016;**173**(Part 3):2014-2021

[22] Vatankhah R, Etemadi S, Alasty A, Vossoughi G. Adaptive critic based neuro fuzzy controller in multi agents: Distributed behavioral control and path tracking. Neurocomputing. 2012;**88**:24-35

[23] Franco ML, Sanchez EN, Alanis AY, Franco CL, Daniel NA. Decentralized control for stabilization of nonlinear multi agent systems using neural inverse optimal control. Neurocomputing. 2015;**168**:81-91

[24] Oleiwi BK, Al-Jarrah R, Roth H, Kazem BI. Integrated motion planing and control for multi objectives optimization and multi robots navigation. In: 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics CESCIT 2015. Vol. 48. 2015. pp. 99-104

[25] Atinc GM, Stipanovic DM, Voulgaris PG. Supervised coverage control of multi agent systems. Automatica. 2014;**50**:2936-2942

[26] Posadas JL, Poza JL, Simo JE, Benet G, Blanes F. Agent based distributed architecture for mobile robot control. Engineering Applications of Artificial Intelligence. 2008;**21**:805-823

[27] Innocenti B, Lopez B, Salvi J. A multi agent architecture with cooperative fuzzy control for a mobile robot. Robotics and Autonomous Systems. 2007;**55**:881-891

[28] Kudelski M, Gambardella LM, Caro GAD. RoboNetSim: An integradted framework for multi robot and network simulation. Robotics and Autonomous Systems. 2013;**61**:483-496

[29] Couceiro MS, Vargas PA, Rocha RP. Bridging the reality gap between the webots simulator and E-puck robots. Robotics and Autonomous Systems. 2014;**62**:1549-1567

[30] Kuniyoshi Y, Riekki J, Rougeaux MIS. Vision based behaviours for multi robot cooperation. In: Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94 'Advanced Robotic Systems and the Real World' IROS '94. Vol. 2. 1994. pp. 923-931

[31] Al-Jarrah R, Shahzad A, Roth H. Path planning and motion coordination fro multi robot system using probabilistic neuro fuzzy. In: 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics CESCIT 2015. Vol. 48; 22-24 June 2015. pp. 46-51

[32] Pham DT, Awadalla MH, Eldukhri EE. Fuzzy and neuro fuzzy based cooperative mobile robots. In: 2nd I*PROMS Virtual International Conference; 3-14 July 2006. pp. 578-583

[33] Azuma S, Yoshimura R, Sugie T. Broadcast control of multi agents systems. Automatica. 2013;**49**:2307-2316

[34] Wang J. On sign board based inter robot communication in distributed robotics systems. In: IEEE International Conference on Robotics and Automation. 1994. pp. 1045-1050

[35] Ichikawa S, Hara F, Hosokai H. Cooperative route searching behavior of multi robot system using hello call communication. In: Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems; 26-30 July 1993. pp. 1149-1156

[36] Li W, Shen W. Swarm behavior control of mobile multi-robots with wireless sensor networks. Journal of Network and Computer Applications. 2011;**34**:1398-1407

[37] Yoshida E, Arai T, Ota J, Miki T. Effect of grouping in local communication system of multiple mobile robots. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems '94 'Advanced Robotic Systems and the Real World' IROS '94. 1994. pp. 808-815

[38] Yoshida E, Yamamoto M, Arai T, Ota J, Kurabayashi D. A design method of local communication area in multiple mobile robot system. In: IEEE International Conference on Robotics and Automation. 1995. pp. 2567-2572

[39] Yoshida E, Yamamoto M, Arai T, Ota J, Kurabayashi D. A design method of local communication range in multiple mobile robot system. In: IEEE International Conference on Robotics and Automation. 1995. pp. 274-279

[40] Sariff N, Buniyamin N. An overview of autonomous robot path planning algorithms. In: 4th Student Conference on Research and Development (SCORED 2006); Shah Alam, Malaysia. June 2006. pp. 184-188

[41] Sariff N, Ismail ZH. Investigation of simultaneous pertubation stochastic algorithm parameters effect towards multi agent robot motion coordination performances. In: 2017 IEEE 7th International Conference on Underwater System Technology: Theory and Applications, Universiti Teknologi Malaysia; Kuala Lumpur. December 2017. pp. 1-6

[42] Sariff N, Buniyamin N. Evaluation of robot path planning algorithms in global static environments: Genetic algorithm VS ant colony optimization algorithm. International Journal of Electrical and Electronic Systems Research (IEESR 2010). 2010;**3**:1-12

[43] Sariff N, Buniyamin N. Genetic algorithm versus ant colony optimization algorithm: Comparison of performances in robot path planning application. In: 7th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2010); Madeira, Portugal. June 2010. pp. 125-132

[44] Sariff N, Buniyamin N. Comparative study of genetic algorithm and ant colony optimization algorithm in global static environment of different complexities. In: 2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2009); Daejeon, Korea. December 2009. pp. 132-137

[45] Buniyamin N, Sariff N, Wan Ngah WAJ, Mohamad Z. Robot global path planning overview and a variation of ant colony system algorithm. International Journal of Mathematics and Computers in Simulation (IMACS 2011). 2011;**5**:9-16

[46] Sariff N, Buniyamin N. Ant colony system for robot path planning in global static environment. In: 9th International Conference on System Science and Simulation in Engineering (ICOSSSE'10); Iwate, Japan. October 2010. pp. 1-6

[47] Sariff N, Buniyamin N. Ant colony system for robot path planning in global static environment. In: Selected Topics in System Science & Simulation in Engineering. World Scientific and Engineering Academic and Society (WSEAS); 2010, pp 192-197

[48] Liu T, Jiang ZP. Distributed nonlinear control of mobile autonomous multi agents. Automatica. 2014;**50**:1075-1086

[49] Peng Z, Yang S, Wen G, Rahmani A, Yu Y. Adaptive distributed formation control for multiple nonholonomic wheeled mobile robots. Neurocomputing. 15 January 2016;**173**(Part 3):1485-1494

[50] Liu Z, Jin Q, Chen Z. Distributed containment control for bounded unknown second order nonlinear multi agent systems with dynamic leaders. Neurocomputing. 2015;**168**: 1138-1143

[51] Bo L, Qiang CZ, Xin LZ, Yan ZC, Qing Z. Containment control of multi agent systems with fixed time delays in fixed directed networks. Neurocomputing. 15 January 2016; **173**(Part 3):2069-2075

[52] Rong L, Shen H, Lu J, Li J. Distributed reference model based containment control of second-order multi agent systems. Neurocomputing. 2015;**168**:254-259

[53] Jolly KG, Kumar RS, Vijayakumar R. Intelligent task planning and action selection of a mobile robot in a multi agent system through a fuzzy neural network approach. Engineering Applications of Artificial Intelligence. 2010;**23**:923-933

[54] Ren W, Cao Y. Overview of recent research in distributed muti-agent coordination. Distributed Coordination of Multi-agent Networks Emergent Problems, Models and Issues; 2011:23-41

[55] Buniyamin N, Sariff N, Wan Ngah WAJ, Mohamad Z. A simple local path planning algorithm for autonomous mobile robots. International Journal of Systems Applications, Engineering & Development (ISAED 2011). 2011;**5**:151-159

[56] Sariff N, Elyana N. Mobile robot obstacles avoidance by using braitenberg approach. In: 2nd International Conference on Emerging Trends in Scientific Research (ICETSR); Kuala Lumpur, Malaysia; November 2014

[57] Mohamed F, Sariff N, ZainalAbidin IZ. Low cost serving robot using fuzzy logic techniques. In: Proceedings of the Second International Conferences on Advances in Automation and Robotics (AAR 2013); Kuala Lumpur, Malaysia; May 2013

[58] Hakim Z, Sariff N, Buniyamin N. The development of a low cost remote control partner lawnmower robot. In: 4th Student Conference on Research and Development (SCORED 2006). June 2006. pp. 152-155

[59] Farina M, Perizzato A, Scattolini R. Application of distributed predictive control to motion and coordination problems for unicycle autonmous robots. Robotics and Autonomous Systems. 2015;**72**:248-260

[60] Peng K, Yang Y. Leader-following consensus problem with a varyning velocity leader and time-varying delays. Physica D: Statictical Mechanics and Its Applications. 2009;**388**: 193-208

[61] Seyboth GS, Dimarogonas DV, Johansson KH. Event based broadcasting for multi agent average consensus. Automatica. 2013;**49**:245-252

[62] Saber RO, Fax JA, Murray RM. Consensus and coooperation in networked multi agent systems. Proceedings of the IEEE. 2007;**95**:215-233

[63] Zhu W, Jiang ZP, Feng G. Event-based consensus of multi-agent systems with general linear models. Automatica. 2014;**50**:552-558

[64] Ren W, Beard RW, Atkins EM. Information consensus in multivechicle cooperative control. IEEE in Control Systems. 2007;**27**:71-82

[65] Wang A. Event based consensus control for single integrator networks with communication time delay. Neurocomputing. 15 January 2016;**173**(part 3):1715-1719

[66] Hou B, Sun F, Li H, Chen Y, Liu G. Observer based cluster consensus control of high order multi agent systems. Neurocomputing. 2015;**168**:979-982

[67] Nowzari C, Cortes J. Zeno-free, distributed event triggered communication and control for multi agent average consensus. In: 2014 American Control Conference (ACC); June 2014. pp. 4-6

[68] Wieland P, Sepulchre R, Allgower F. An internal model principle is necessary and sufficient for linear output synchronization. Automatica. 2011;**47**:1068-1074

[69] Wang X, Ji H, Wang C. Distributed output regulation of leader follower multi agents systems. International Journal of Robust and Nonlinear Control. 10 January 2013;**23**(1): 48-66

[70] Su Y, Huang J. Cooperative output regulation of linear multi-agent systems. IEEE Transactions on Automatic Control. 2012;**57**:1062-1066

[71] Lee TH, Park JH, Ji DH, Jung HY. Leader following consensus problem of heterogeneous multi-agent systems with nonlinear dynamics using fuzzy disturbance observer. Complexity. 2014;**19**:20-31

[72] Li J, Yan W. Consensus problems for multi agent system with broadcasting type topology. In: 2012 Second International Conference on Instrumentation & Measurement, Computer, Communication and Control. 2012. pp. 967-970

[73] Azuma S, Yoshimura R, Sugie T. Multi-agent consensus under a communication broadcast mixed environment. International Journal of Control. 2014;**87**:1103-1116

[74] Das K, Ghose D. Positional consensus in multi agent systems using a broadcast control mechanism. In: 2009 American Control Conference; 10-12 June 2009. pp. 5731-5736

[75] Das K, Ghose D. Broadcast control mechanism for positional consensus in multiagent system. IEEE Transactions on Control Systems Technology. 5 Sept. 2015;**23**(5):1807-1826

[76] Ping HJ, Wen YH. Collective coordination of multi-agent systems guided by multiple leaders. Chinese Pyhsics B. 2009;**18**:3777-3782

[77] Wang X, Li S, Shi P. Distributed finite-time containment control for double integrator multi-agent systems. IEEE Transactions on Cybernetics. 2014;**44**:1518-1528

[78] Mei J, Ren W, Ma BLG. Containment control for networked unknown langrangian systems with multiple dynamic leaders under a directed graph. In: Proceedings of the American Control Conference. 2013. pp. 522-527

[79] Liu H, Cheng L, Hao MTZ. Containment control of double-integrator multi-agent systems with a periodic sampling: A small-gain theorem based method. Proceedings of 33nd Chinese Control Conference; Nanjing, China. 2014. pp. 1407-1412

[80] Youcheng L, Yiguang H. Multi-leader set coordination of multi agent systems with random switching topologies. In: Proceedings of the IEEE International Conference on Decision and Control. 2010. pp. 3820-3825

[81] Yuanshi Z, Long W. Containment control of heterogeneous multi-agent systems. International Journal of Control. 2014;**87**:1-8

[82] Chen YQ, Wang Z. Formation control: A review and a new consideration. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2005. pp. 3664-3669

[83] Oh KK, Park MC, Ahn HS. A survey of multi agent formation control. Automatica. 2015;**53**:424-440

[84] Nascimento TP, Moreira AP, Conceicao AGS. Multi robot nonlinear model predictive formation control: Moving target and target absence. Robotics and Autonomous Systems. 2013;**61**:1502-1515

[85] Consolini L, Morbidi F, Prattichizzo D, Tosques M. A geometric characterization of leader follower formation control. In: IEEE International Conference on Robotics and Automation. 2007. pp. 2397-2402

[86] Consolini L, Morbidi F, Prattichizzo D, Tosques M. Leader follower formation control as a disturbance decoupling problem. In: European Control Conference (ECC). 2007. pp. 1492-1497

[87] Balch T, Arkin RC. Behaviour-based formation control for multi-robots teams. IEEE Transactions on Robotics and Automation. 1998;**14**:926-939

[88] Lewis, Tan. High precision formation control of mobile robots using virtual structures. Autonomous Robots. 1997;**4**:387-403

[89] Das AK, Fierro R, Kumar V, Ostrowski JP, Spletzer J, Taylor CJ. A vision based formation control framework. IEEE Transactions on Robotics and Automation. 2002;**18**:813-825

[90] Sarker MOF, Dahl TS, Arcaute E, Christensen K. Local interactions over global broadcasts for improves task allocation in self organized multi robot systems. Robotics and Autonomous Systems. 2014;**62**:1453-1462

[91] Tolmidis AT, Petrou L. Multi objective optimization for dynamic task allocation in a multi robot system. Engineering Applications of Artificial Intelligence. 2013;**26**:1458-1468

[92] Nagarajan T, Thondiyath A. Heuristic based task allocation algorithm for multiple robots using agents. In: International Conference on Design and Manufacturing IConDM. Vol. 64. 2013. pp. 844-853

[93] Sariff N, Nadihah NH. Automatic mobile robot obstacles avoidances in a static environment using hybrid approaches (fuzzy logic and artificial neural network). In: 2014 International Conference Artificial Intelligence System Technology (ICAIST); Kota Kinabalu, Sabah; December 2014

[94] Mohamed F, Sariff N, Abidin IZZ. Low cost serving robot using fuzzy logic techniques. International Journal of Advancements in Mechanical and Aeronautical Engineering (IJAMAE). 2013;**1**:54-58

[95] Mohamad MF, Sariff N, Buniyamin N. Mobile robot obstacle avoidance in various type of static environments using fuzzy logic approach. In: 2014 International Conference on Electrical, Electronics and System Engineering (ICEESE2014); December 2014

[96] Akmal Jeffril M, Sariff N. The integration of fuzzy logic and artificial neural network method for mobile robot obstacles avoidance in a static environment. In: 2013 IEEE 3rd International Conferences on System Engineering and Technology (ICSET); Shah Alam, Malaysia. August 2013. pp. 326-330

[97] Hajar Ashikin S, Akmal Jeffril M, Sariff N. Mobile robot obstacles avoidances by using fuzzy logic techniques. In: 2013 IEEE 3rd International Conferences on System Engineering and Technology (ICSET); Shah Alam, Malaysia. 2013. pp. 332-335

[98] Tosic PT, Vilalta R. A unified farmework for reinforcement learning, co-learning and meta-learning how to coordinate in collaborative multi agents systems. In: International Conference on Computational Science ICCS. Vol. 1. 2012. pp. 2217-2226

[99] Azuma S, Yoshimura R, Sugie T. Broadcast control of group of Markov chains. In: 51st IEEE Conference on Decision and Control; 10-13 December 2012. pp. 2059-2064

[100] Kumar G, Kothare MV. Broadcast stochastic receding horizon control of multi agent systems. Automatica. 2013;**49**:3600-3606

Chapter 2

Motion Control and Velocity-Based Dynamic Compensation for Mobile Robots

Felipe Nascimento Martins and
Alexandre Santos Brandão

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79397

Abstract

The design of motion controllers for wheeled mobile robots is often based only on the robot's kinematics. However, to reduce tracking error it is important to also consider the robot dynamics, especially when high-speed movements and/or heavy load transportation are required. Commercial mobile robots usually have internal controllers that accept velocity commands, but the control signals generated by most dynamic controllers in the literature are torques or voltages. In this chapter, we present a velocity-based dynamic model for differential-drive mobile robots that also includes the dynamics of the robot actuators. Such model can be used to design controllers that generate velocity commands, while compensating for the robot dynamics. We present an explanation on how to obtain the parameters of the dynamic model and show that motion controllers designed for the robot's kinematics can be easily integrated with the velocity-based dynamic compensation controller. We conclude the chapter with experimental results of a trajectory tracking controller that show a reduction of up to 50% in tracking error index *IAE* due to the application of the dynamic compensation controller.

Keywords: velocity-based dynamic model, dynamic modeling, dynamic compensation, motion control, tracking control

1. Introduction

A common configuration for mobile robots is the differential drive, which has two independently driven parallel wheels and one (or more) unpowered wheel to balance the structure [1]. For several years differential-drive mobile robots (DDMR) have been widely used in many applications because of their simple configuration and good mobility. Some applications of

DDMR are surveillance [2], floor cleaning [3], industrial load transportation [4], autonomous wheelchairs [5], and others.

In the literature, most of the motion controllers for DDMR are based only on its kinematics. The main reasons for that are: (a) the kinematic model is simpler than the dynamic model, therefore the resulting controllers are less complex and simpler to tune; (b) the accuracy of the dynamic model depends on several parameters that might change or are difficult to measure, like the robot's mass and moment of inertia; and (c) dynamic controllers usually generate torque or voltage commands, while mobile robots frequently have internal velocity controllers that take velocity as input [6]. However, the robot's low-level velocity control loops do not guarantee perfect velocity tracking, especially when high-speed movements and/or heavy load transportation are required. In such cases, to reduce tracking error, it becomes essential to consider the robot dynamics as well, as shown in [7].

A possible solution to overcome the problem described above is to design a controller that compensates for the robot's dynamics. Commercial mobile robots usually have internal controllers that accept velocity commands, like the Pioneer 3 from Adept Mobile Robots, the Khepera from K-Team Corporation, and the robuLAB-10 from Robosoft Inc. However, the control signals generated by most dynamic controllers in the literature are torques or voltages, as in [8–14]. Because of that, some researchers have proposed dynamic controllers that generate linear and angular velocities as commands [15, 16]. In some works, the dynamic model is divided in to two parts, allowing the design of independent controllers for the robot kinematics and dynamics [17–20]. Finally, to reduce performance degradation in applications in which the robot dynamic parameters may vary (such as load transportation) or when the knowledge of the dynamic parameters is imprecise, adaptive controllers can also be considered [7, 21].

The above-mentioned works applied a dynamic model that has linear and angular velocities as inputs, which illustrates the interest on such kind of dynamic model. In such context, this chapter explains the velocity-based dynamic model and its mathematical properties, which are useful for the design of controllers that compensate for the robot dynamics. It also illustrates how to design a trajectory tracking motion controller based on the robot's kinematics, and how to integrate it with a velocity-based dynamic compensation controller.

2. Dynamic model

The classical equation to represent the dynamics of mobile robots can be obtained via Lagrangian formulation, resulting in [22].

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V_m}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F_m}(\dot{\mathbf{q}}) + \mathbf{G_m}(\mathbf{q}) + \tau_{\mathbf{d}} = \mathbf{B}(\mathbf{q})\tau - \mathbf{A^T}(\mathbf{q})\lambda, \tag{1}$$

where $\mathbf{q} = \begin{bmatrix} q_1 & q_2 & \dots & q_n \end{bmatrix}^T$ is the vector of generalized coordinates of the system with n degrees of freedom, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the matrix of inertia, $\mathbf{V_m}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ is the matrix of Coriolis and centrifugal forces, $\mathbf{F_m}(\dot{\mathbf{q}}) \in \mathbb{R}^{n\times 1}$ is the vector that represents viscous friction, $\mathbf{G_m}(\mathbf{q}) \in \mathbb{R}^{n\times 1}$ is the vector of gravitational torques, $\tau_{\mathbf{d}} \in \mathbb{R}^{n\times 1}$ is the disturbance vector,

$\tau \in \mathbb{R}^{r\times 1}$ is the vector of input torques, where r is the number of inputs, $\mathbf{B}(\mathbf{q}) \in \mathbb{R}^{n\times r}$ is the input transformation matrix, $\lambda \in \mathbb{R}^{m\times 1}$ is the vector that represents restriction forces, and $\mathbf{A}(\mathbf{q}) \in \mathbb{R}^{m\times n}$ is the matrix associated to such restrictions. Two well known properties of such model are [9, 22]:

1. $\mathbf{M}(\mathbf{q})$ is a symmetric and positive definite matrix, that is, $\mathbf{M}(\mathbf{q}) = \mathbf{M}(\mathbf{q})^T > 0$;
2. $\left(\dot{\mathbf{M}} - \mathbf{2V_m}\right)$ is antisymmetric.

The above properties are widely used on the development and stability analysis of controllers for mobile robots, as shown in [8, 9, 22, 23]. But, such controllers generate torque commands, not velocities, as usually accepted by commercial robots. The conversion from torque to velocity commands requires knowledge of the actuation system of the robot (model of its motors and its speed controllers). On the other hand, a controller designed from a velocity-based dynamic model generates linear and angular velocities that can be directly applied as commands for mobile robots.

In such a context, now the dynamic model for the DDMR proposed in [16] is reviewed. For convenience, we first present its equations again. Then, the dynamic model is written in such a way that it becomes similar to the classical dynamic equation based on torques. **Figure 1** depicts a DDMR with the variables of interest. There, u and ω are, respectively, the linear and angular velocities, G is the center of mass, h is the point of interest (whose position should be controlled)

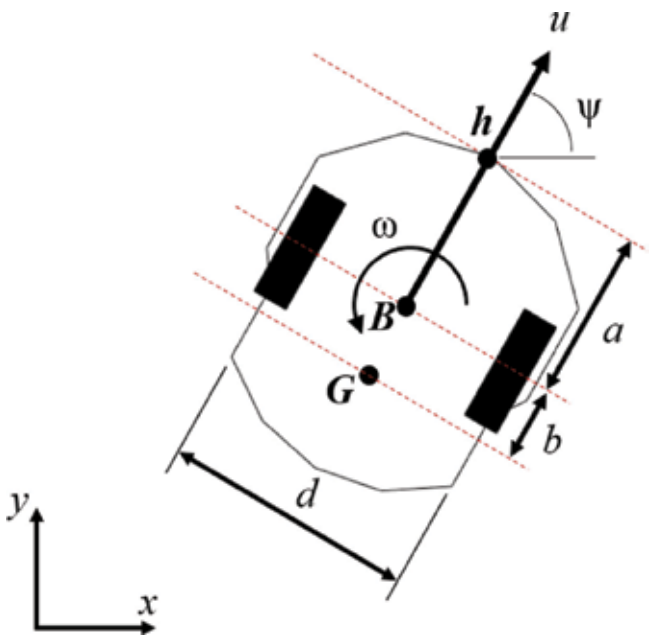


Figure 1. The differential drive mobile robot (DDMR). u and ω are, respectively, the linear and angular velocities, G is the center of mass, h is the point of interest with coordinates x and y in the XY plane, ψ is the robot orientation, a is the distance from the point of interest to the point in the middle of the virtual axle that links the traction wheels (point B), b is the distance between points G and B, and d is the distance between the points of contact of the traction wheels to the floor.

with coordinates x and y in the XY plane, ψ is the robot orientation, a is the distance from the point of interest to the point in the middle of the virtual axle that links the traction wheels (point B), b is the distance between points G and B, and d is the distance between the points of contact of the traction wheels to the floor. The complete mathematical model is written as [16].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{u} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} u\cos\psi - a\omega\sin\psi \\ u\sin\psi + a\omega\cos\psi \\ \omega \\ \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}u \\ -\frac{\theta_5}{\theta_2}u\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2} \end{bmatrix} \begin{bmatrix} u_r \\ \omega_r \end{bmatrix} + \begin{bmatrix} \delta_x \\ \delta_y \\ 0 \\ \delta_u \\ \delta_\omega \end{bmatrix}, \tag{2}$$

where $\theta = [\theta_1, \ldots, \theta_6]^T$ is the vector of identified parameters and $\delta = \left[\delta_x, \delta_y, 0, \delta_u, \delta_\omega\right]^T$ is the vector of parametric uncertainties associated to the mobile robot. The equations describing the parameters θ are presented in Section 3. The model is split into kinematic and dynamic parts. The kinematic model is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi & -a\sin\psi \\ \sin\psi & a\cos\psi \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ \omega \end{bmatrix} + \begin{bmatrix} \delta_x \\ \delta_y \\ 0 \end{bmatrix}, \tag{3}$$

which is a modified approach to describe the robot kinematics. The classical unicycle model is obtained when $a = 0$ in (3), but here we consider the case in which $a \neq 0$, which means that the (x, y) position described by the model is not in the center of the line between the traction wheels, but at a distance a from it (see point h in **Figure 1**). We use this model because it is useful on the design of the trajectory tracking controller, as shown in Section 4.

The part of the equation that represents the dynamics is given by

$$\begin{bmatrix} \dot{u} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}u \\ -\frac{\theta_5}{\theta_2}u\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix} + \begin{bmatrix} \frac{1}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2} \end{bmatrix} \begin{bmatrix} u_r \\ \omega_r \end{bmatrix} + \begin{bmatrix} \delta_u \\ \delta_\omega \end{bmatrix}. \tag{4}$$

As shown in [21], by rearranging its terms the Eq. (4) can be written as

$$\begin{bmatrix} -\theta_1 & 0 \\ 0 & -\theta_2 \end{bmatrix} \begin{bmatrix} \delta_u \\ \delta_\omega \end{bmatrix} + \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \theta_4 & -\theta_3\omega \\ \theta_5\omega & \theta_6 \end{bmatrix} \begin{bmatrix} u \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_r \\ \omega_r \end{bmatrix}, \tag{5}$$

or, in a compact form, as

$$\Delta + \mathbf{H}'\dot{\mathbf{v}} + \mathbf{c}(\mathbf{v})\mathbf{v} = \mathbf{v_r}, \tag{6}$$

where $\mathbf{v_r} = [u_r \quad \omega_r]^T$ is the vector of reference velocities, $\mathbf{v} = [u \quad \omega]^T$ is the vector containing the actual robot velocities, and the matrices $\mathbf{H}'$ and $\mathbf{c}(\mathbf{v})$, and the vector Δ are given by

$$\mathbf{H}' = \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix}, \qquad \mathbf{c}(\mathbf{v}) = \begin{bmatrix} \theta_4 & -\theta_3\omega \\ \theta_5\omega & \theta_6 \end{bmatrix} \quad \text{and} \quad \Delta = \begin{bmatrix} -\theta_1 & 0 \\ 0 & -\theta_2 \end{bmatrix} \begin{bmatrix} \delta_u \\ \delta_\omega \end{bmatrix}. \tag{7}$$

Let us rewrite $\mathbf{c}(\mathbf{v})$ by adding and subtracting the term $i\theta_3 u$ to its fourth element (where $i = 1rad^2/s$), such that

$$\mathbf{c}(\mathbf{v}) = \begin{bmatrix} \theta_4 & -\theta_3\omega \\ \theta_5\omega & \theta_6 + (i\theta_3 - i\theta_3)u \end{bmatrix}, \tag{8}$$

so that the term $\mathbf{c}(\mathbf{v})\mathbf{v}$ can be written as

$$\begin{bmatrix} 0 & -\theta_3\omega \\ \theta_3\omega & 0 \end{bmatrix} \begin{bmatrix} iu \\ \omega \end{bmatrix} + \begin{bmatrix} \theta_4 & 0 \\ 0 & \theta_6 + (\theta_5 - i\theta_3)u \end{bmatrix} \begin{bmatrix} u \\ \omega \end{bmatrix}. \tag{9}$$

The role of the term $i = 1rad^2/s$ is to make the units consistent to allow us to split $\mathbf{c}(\mathbf{v})$ into two matrices, while keeping the numerical values unchanged. Now, let us define $\mathbf{v}' = [iu \quad \omega]^T$ as the vector of modified velocities, so that

$$\mathbf{v}' = \begin{bmatrix} i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ \omega \end{bmatrix}. \tag{10}$$

The terms in the vector of modified velocities are numerically equal to the terms in the vector of actual velocities $\mathbf{v}$, only its dimensions are different. By rewriting the model equation, the following matrices are defined:

$$\mathbf{H} = \begin{bmatrix} \theta_1/i & 0 \\ 0 & \theta_2 \end{bmatrix}, \qquad \mathbf{F}(\mathbf{v}') = \begin{bmatrix} \theta_4/i & 0 \\ 0 & \theta_6 + (\theta_5/i - \theta_3)iu \end{bmatrix}, \quad \text{and} \quad \mathbf{C}(\mathbf{v}') = \begin{bmatrix} 0 & -\theta_3\omega \\ \theta_3\omega & 0 \end{bmatrix}. \tag{11}$$

Finally, the dynamic model of a DDMR can be represented by

$$\mathbf{v_r} = \mathbf{H}\dot{\mathbf{v}}' + \mathbf{C}(\mathbf{v}')\mathbf{v}' + \mathbf{F}(\mathbf{v}')\mathbf{v}' + \Delta, \tag{12}$$

or

$$\begin{bmatrix} u_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} \theta_1/i & 0 \\ 0 & \theta_2 \end{bmatrix} \begin{bmatrix} i\dot{u} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} 0 & -\theta_3\omega \\ \theta_3\omega & 0 \end{bmatrix} \begin{bmatrix} iu \\ \omega \end{bmatrix} + \begin{bmatrix} \theta_4/i & 0 \\ 0 & \theta_6 + (\theta_5/i - \theta_3)iu \end{bmatrix} \begin{bmatrix} iu \\ \omega \end{bmatrix} + \begin{bmatrix} \delta_u \\ \delta_\omega \end{bmatrix}. \tag{13}$$

Notice that $\mathbf{c}(\mathbf{v})\mathbf{v} = \mathbf{C}(\mathbf{v}')\mathbf{v}' + \mathbf{F}(\mathbf{v}')\mathbf{v}'$ and $\mathbf{H}'\dot{\mathbf{v}} = \mathbf{H}\dot{\mathbf{v}}'$, that is, the dimensions of the resulting vector $\mathbf{v_r}$ are kept unchanged.

The model represented by Eq. (13) is mathematically equivalent to the one proposed in [16] and used in [7], where it was validated via simulation and experiments. Nevertheless, the model presented here is written in such a way that some mathematical properties arise. Such

properties, which are presented and discussed in the next session, can be applied on the design and stability analysis of dynamic controllers.

3. Dynamic parameters and model properties

To calculate the dynamic parameters of the vector $\boldsymbol{\theta}$, one has to know physical parameters of the robot, like its mass, its moment of inertia, friction coefficient of its motors, etc. The equations describing each one of the parameters θ_i are

$$\begin{aligned}
\theta_1 &= \left[\frac{R_a}{k_a}\left(mr^2 + 2I_e\right) + 2rk_{DT}\right]\frac{1}{(2rk_{PT})} \quad [s],\\
\theta_2 &= \left[\frac{R_a}{k_a}\left(I_e d^2 + 2r^2\left(I_z + mb^2\right)\right) + 2rdk_{DR}\right]\frac{1}{(2rdk_{PR})} \quad [s],\\
\theta_3 &= \frac{R_a}{k_a}\frac{mbr}{2k_{PT}} \quad \left[sm/rad^2\right],\\
\theta_4 &= \frac{R_a}{k_a}\left(\frac{k_a k_b}{R_a} + B_e\right)\frac{1}{rk_{PT}} + 1,\\
\theta_5 &= \frac{R_a}{k_a}\frac{mbr}{dk_{PR}} \quad [s/m], \text{ and}\\
\theta_6 &= \frac{R_a}{k_a}\left(\frac{k_a k_b}{R_a} + B_e\right)\frac{d}{2rk_{PR}} + 1,
\end{aligned} \tag{14}$$

where m is the mass of the robot, I_z is its moment of inertia at G, R_a, k_b and k_a are the electrical resistance, the electromotive constant, and the constant of torque of its motors, respectively, B_e is the coefficient of friction, I_e is the moment of inertia of each group rotor-reduction gear-wheel, r is the radius of each wheel, and b and d are distances defined in **Figure 1**. It is assumed that the internal motor controllers are of type PD (proportional-derivative) with proportional gains $k_{PT} > 0$ and $k_{PR} > 0$, and derivative gains $k_{DT} \geq 0$ and $k_{DR} \geq 0$. It is also assumed that the inductances of the motors are negligible, and both driving motors are identical.

Obtaining accurate values of all physical parameters of a robot might be difficult, or even not possible. Therefore, it is useful to discuss an identification procedure to directly obtain the values of the dynamic parameters $\boldsymbol{\theta}$. Such procedure is explained in Section 3.2.

It is interesting to point out that the dynamic model adopted here considers that the robot's center of mass G can be located anywhere along the line that crosses the center of the structure, as illustrated in **Figure 1**. This means that the formulation of the proposed dynamic model is adequate for robots that have a symmetrical weight distribution between their left and right sides. Because most differential drive robots have an approximately symmetrical weight distribution (with each motor and wheel on either left or right sides), such assumption does not introduce significant modeling errors on most cases. It should also be noticed that $\theta_i > 0$ for

$i = 1, 2, 4, 6$. The parameters θ_3 and θ_5 can be negative and will be null if, and only if, the center of mass G is exactly in the center of the virtual axle, that is, $b = 0$. Finally, in [21], it was shown that the model parameters θ_1 to θ_6 cannot be written as a linear combination of each other, that is, they are independent.

3.1. Model properties

The mathematical properties of the dynamic model (12) are:

1. The matrix $\mathbf{H}$ is symmetric and positive definite, or $\mathbf{H} = \mathbf{H}^T > 0$;
2. The inverse of $\mathbf{H}$ exists and is also positive definite, or $\exists \quad \mathbf{H}^{-1} > 0$;
3. The matrix $\mathbf{F}(\mathbf{v}')$ is symmetric and positive definite, or $\mathbf{F}(\mathbf{v}') = \mathbf{F}^T > 0$, if $\theta_6 > -(\theta_5/i - \theta_3)iu$;
4. The matrix $\mathbf{H}$ is constant if there is no change on the physical parameters of the robot;
5. The matrix $\mathbf{C}(\mathbf{v}')$ is skew symmetric;
6. The matrix $\mathbf{F}(\mathbf{v}')$ can be considered constant if $\theta_6 \gg |(\theta_5/i - \theta_3)iu|$ and there is no change on the physical parameters of the robot;
7. The mapping $\mathbf{v_r} \rightarrow \mathbf{v}'$ is strictly output passive if $\theta_6 > -(\theta_5/i - \theta_3)iu$ and $\Delta = 0$.

To analyze the above mathematical properties, first recall that $\theta_i > 0$ for $i = 1, 2, 4, 6$. Properties 1 and 2 can be confirmed by observing that $\mathbf{H}$ is a diagonal square matrix formed by θ_1 and θ_2. $\mathbf{F}(\mathbf{v}')$ is also a diagonal square matrix formed by θ_4 and $\theta_6 + (\theta_5/i - \theta_3)iu$. Property 3 holds if $\theta_6 > -(\theta_5/i - \theta_3)iu$. Property 4 holds if there is no change on the physical parameters of the robot (i.e., if there is no change on the robot's mass, moment of inertia, etc.). $\mathbf{C}(\mathbf{v}')$ is a square matrix formed by $\theta_3\omega$ and $-\theta_3\omega$, whose transpose is also its negative, which proves property 5. Property 6 holds if there is no change on the physical parameters of the robot and $\theta_6 \gg |(\theta_5/i - \theta_3)iu|$. Finally, the proof for property 7 is given in [21].

3.2. Identified parameters

The values of the dynamic parameters $\boldsymbol{\theta}$ can be estimated via an identification procedure, described as follows. Let a system be represented by the regression model

$$\mathbf{Y}=\mathbf{W}\boldsymbol{\theta}, \tag{15}$$

where $\boldsymbol{\theta}$ is the vector of parameters and $\mathbf{Y}$ is the system output. The least squares estimate of $\boldsymbol{\theta}$ is given by

$$\widehat{\boldsymbol{\theta}} = \left(\mathbf{W}^{\mathbf{T}}\mathbf{W}\right)^{-1}\mathbf{W}^{\mathbf{T}}\mathbf{Y}, \tag{16}$$

where $\widehat{\boldsymbol{\theta}}$ is the vector with the estimated values of $\boldsymbol{\theta}$ and $\mathbf{W}$ is the regression matrix. By rearranging (4) and ignoring uncertainty, the dynamic model can be represented by

$$\begin{bmatrix} u_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} \dot{u} & 0 & -\omega^2 & u & 0 & 0 \\ 0 & \dot{\omega} & 0 & 0 & u\omega & \omega \end{bmatrix} [\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6]^T, \tag{17}$$

where

$$\mathbf{Y} = \begin{bmatrix} u_r \\ \omega_r \end{bmatrix}, \mathbf{W} = \begin{bmatrix} \dot{u} & 0 & -\omega^2 & u & 0 & 0 \\ 0 & \dot{\omega} & 0 & 0 & u\omega & \omega \end{bmatrix}, \boldsymbol{\theta} = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6]^T. \tag{18}$$

In order to obtain an estimate for the values of $\boldsymbol{\theta}$, each robot needs to be excited with speed reference signals (u_r, ω_r), while the actual values of its velocities (u, ω) and accelerations $(\dot{u}, \dot{\omega})$ are measured and stored. In our case, the excitation signals consisted of a sum of six sine waves with different frequencies and amplitudes. All data were stored and the regression model was assembled so that the vector $\mathbf{Y}$ and the matrix $\mathbf{W}$ had all values obtained in each sampling instant. Subsequently, the value of $\boldsymbol{\theta}$ for each robot was calculated by least squares method.

In order to verify the assumptions that $\theta_6 \gg |(\theta_5/i - \theta_3)iu|$ and $\theta_6 > -(\theta_5/i - \theta_3)iu$, we have analyzed the dynamic parameters of five differential drive robots obtained via identification procedure. The analysis was done considering the parameters of the following robots: a Pioneer 3-DX with no extra equipment ($P3$), a Pioneer 3-DX with a LASER scanner and omnidirectional camera ($P3_{laser}$), a robotic wheelchair while carrying a 55 kg person (RW_{55}), a robotic wheelchair while carrying a 125kg person (RW_{125}), and a Khepera III ($KIII$). The Khepera III robot weighs 690 g, has a diameter of 13 cm and is 7 cm high. Its dynamic parameters were identified by Laut and were originally presented in [24]. By its turn, the Pioneer robots weigh about 9 kg, are 44 cm long, 38 cm wide, and 22 cm tall (without the LASER scanner). The LASER scanner weighs about 50% of the original robot weight, which produces an important change in the mass and moment of inertia of the structure. Finally, the robotic wheelchair presents an even greater difference in dynamics because of its own weight (about 70 kg) and the weight of the person that it is carrying. The dynamic parameters for the above-mentioned robots are presented in **Table 1**.

The value of u is limited to 0.5 m/s for the Khepera III robots, to 1.2 m/s for the Pioneer robots, and to 1.5 m/s for the robotic wheelchair. Therefore, using the values presented in **Table 1** one

	$P3$	$P3_{laser}$	RW_{55}	RW_{125}	$KIII$
$\theta_1[s]$	0.5338	0.2604	0.3759	0.4263	0.0228
$\theta_2[s]$	0.2168	0.2509	0.0188	0.0289	0.0568
$\theta_3[sm/rad^2]$	−0.0134	−0.0005	0.0128	0.0058	−0.0001
θ_4	0.9560	0.9965	1.0027	0.9883	1.0030
$\theta_5[s/m]$	−0.0843	0.0026	−0.0015	0.0134	0.0732
θ_6	1.0590	1.0768	0.9808	0.9931	0.9981

Table 1. Identified dynamic parameters of a Pioneer 3-DX with no extra equipment ($P3$), a Pioneer 3-DX with a LASER scanner ($P3_{laser}$), a robotic wheelchair while carrying a 55 kg person (RW_{55}), a robotic wheelchair while carrying a 125 kg person (RW_{125}), and a Khepera III ($KIII$).

can verify that the conditions of $\theta_6 > -(\theta_5/i - \theta_3)iu$ and $\theta_6 \gg |(\theta_5/i - \theta_3)iu|$ are valid for all sets of identified parameters. Therefore, the dynamic model of the above-mentioned robots can be represented as in (12), with properties 1–7 valid under the considered conditions.

4. Controller design

To illustrate the usefulness of the modified model and its properties, in this section we show the design of a trajectory tracking controller and a dynamic compensation controller. The controller design is split in two parts, as in [7]. The first part is based on the inverse kinematics and the second one compensates for the robot dynamics. The use of the dynamic model properties is shown on the second part.

The control structure is shown in **Figure 2**, where blocks **K**, **D**, and **R** represent the kinematic controller, the dynamic compensation controller, and the robot, respectively. **Figure 2** shows that the kinematic controller receives the desired values of position $\mathbf{h_d} = [x_d \;\; y_d]^T$ and velocity $\dot{\mathbf{h}}_\mathbf{d}$ from the trajectory planner (which is not considered in this work). Then, based on those values and on the actual robot position $\mathbf{h} = [x \;\; y]^T$ and orientation ψ, the kinematic controller calculates the desired robot velocities $\mathbf{v_d} = [u_d \quad \omega_d]^T$. The desired velocities $\mathbf{v_d}$ and the actual robot velocities $\mathbf{v} = [u \quad \omega]^T$ are fed into the dynamic controller. Such controller uses those values and the estimates of the robot parameters $\boldsymbol{\theta}$ to generate the velocity commands $\mathbf{v_r} = [u_r \quad \omega_r]^T$ that are sent as references to the robot internal controller.

4.1. Kinematic controller

The same kinematic controller presented in [7, 21] is shown here. It is a trajectory tracking controller based on the inverse kinematics of the robot. If only the position of the point of interest $\mathbf{h} = [x \;\; y]^T$ is considered, the robot's inverse kinematics can be written as

$$\begin{bmatrix} u \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\frac{1}{a}sin\psi & \frac{1}{a}\cos\psi \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}. \tag{19}$$

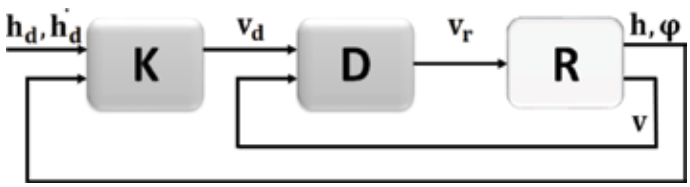


Figure 2. Structure of the control system. The kinematic controller **K** receives the desired values of position $\mathbf{h_d}$ and velocity $\dot{\mathbf{h}}_\mathbf{d}$, the actual robot position $\mathbf{h}$ and its orientation ψ, and calculates the desired robot velocities $\mathbf{v_d}$. Those values and the actual robot velocities $\mathbf{v}$ are fed into the dynamic compensation controller **D**, that generates the velocity commands $\mathbf{v_r}$ that are sent as references to the robot **R**.

The inverse kinematics described by Eq. (19) is valid only for $a \neq 0$. This is the reason why we prefer to adopt this model instead of the classical unicycle model, as discussed earlier. Considering (19), the adopted control law is

$$\begin{bmatrix} u_d \\ \omega_d \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\frac{1}{a}sin\psi & \frac{1}{a}\cos\psi \end{bmatrix} \begin{bmatrix} \dot{x}_d + l_x \tanh\left(\frac{k_x}{l_x}\tilde{x}\right) \\ \dot{y}_d + l_y \tanh\left(\frac{k_y}{l_y}\tilde{y}\right) \end{bmatrix}, \tag{20}$$

for which $\mathbf{v_d} = [u_d \quad \omega_d]^T$ is the vector of desired velocities given by the kinematic controller; $\mathbf{h} = [x \ y]^T$ and $\mathbf{h_d} = [x_d \ y_d]^T$ are the vectors of actual and desired coordinates of the point of interest h, respectively; $\tilde{\mathbf{h}} = [\tilde{x} \quad \tilde{y}]^T$ is the vector of position errors given by $\mathbf{h_d} - \mathbf{h}$; $k_x > 0$ and $k_y > 0$ are the controller gains; l_x, $l_y \in \mathbb{R}$ are saturation constants; and $a > 0$. The tanh terms are included to limit the values of the desired velocities $\mathbf{v_d}$ to avoid saturation of the robot actuators in case the position errors $\tilde{\mathbf{h}}$ are too big, considering $\dot{\mathbf{h}}_\mathbf{d}$ is appropriately bounded.

It is important to point out that the orientation of a DDMR is always tangent to the path being followed. Moreover, the desired trajectory defines the desired linear speed u_d, which means that the robot will be moving either forward or backwards. Therefore, it is not necessary for the controller to explicitly control the robot's orientation to make it successfully follow a trajectory with a desired orientation.

For the stability analysis of the kinematic controller, it is supposed a perfect velocity tracking, which allows equating (19) and (20) under the assumption of $u \equiv u_d$ and $\omega \equiv \omega_d$, which means that the dynamic effects are, at this moment, ignored. Then, the closed-loop equation is obtained in terms of the velocity errors, which is

$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{y}} \end{bmatrix} + \begin{bmatrix} l_x & 0 \\ 0 & l_y \end{bmatrix} \begin{bmatrix} \tanh\left(\frac{k_x}{l_x}\tilde{x}\right) \\ \tanh\left(\frac{k_y}{l_y}\tilde{y}\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{21}$$

Now, the output error vector $\tilde{\mathbf{h}}$ (21) can be written as

$$\dot{\tilde{\mathbf{h}}} = -\left[l_x \tanh\left(\tfrac{k_x}{l_x}\tilde{x}\right) \quad l_y \tanh\left(\tfrac{k_y}{l_y}\tilde{y}\right) \right]^T, \tag{22}$$

which has an unique equilibrium point at the origin. To conclude the stability analysis of such equilibrium, $V = \frac{1}{2}\tilde{\mathbf{h}}^T\tilde{\mathbf{h}} > 0$ is considered as the Lyapunov's candidate function. Its first time derivative is

$$\dot{V} = \tilde{\mathbf{h}}^T\dot{\tilde{\mathbf{h}}} = -\tilde{x}l_x \tanh\left(\frac{k_x}{l_x}\tilde{x}\right) - \tilde{y}l_y \tanh\left(\frac{k_y}{l_y}\tilde{y}\right) < 0, \forall\tilde{\mathbf{h}}. \tag{23}$$

Regarding these results, one can immediately conclude that the system characterized so far has a globally asymptotically stable equilibrium at the origin, which means that the position errors

$\tilde{x}(t) \to 0$ and $\tilde{y}(t) \to 0$ as $t \to \infty$. This result will be revisited latter, after adding a dynamic controller to the system in order to implement the whole control scheme.

Remark. Considering the case in which the reference is a fixed destination point, instead of a trajectory, the robot reaches such a point and stops there. Assuming $u \equiv u_d$ and $\omega \equiv \omega_d$, Eq. (20) guarantees that $\omega = 0$ when $\tilde{x} = 0$ and $\tilde{y} = 0$, therefore $\psi(t) \to \psi_{constant}$.

4.2. Dynamic compensation controller

Now, the use of the proposed dynamic model and its properties is illustrated via the design of a dynamic compensation controller. It receives the desired velocities $\mathbf{v_d}$ from the kinematic controller and generates a pair of linear and angular velocity references $\mathbf{v_r}$ for the robot servos, as shown in **Figure 2**. First, let us define the vector of modified velocities $\mathbf{v'_d}$ as

$$\mathbf{v'_d} = \begin{bmatrix} u'_d \\ \omega_d \end{bmatrix} = \begin{bmatrix} i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_d \\ \omega_d \end{bmatrix}, \tag{24}$$

and the vector of velocity errors is given by $\tilde{\mathbf{v}}' = \mathbf{v'_d} - \mathbf{v}'$.

Regarding parametric uncertainties, the proposed dynamic compensation control law is

$$\mathbf{v_r} = \widehat{\mathbf{H}}\left(\dot{\mathbf{v}}'_\mathbf{d} + \mathbf{T}(\tilde{\mathbf{v}}')\right) + \widehat{\mathbf{C}}\mathbf{v'_d} + \widehat{\mathbf{F}}\mathbf{v'_d}, \tag{25}$$

where $\widehat{\mathbf{H}}$, $\widehat{\mathbf{C}}$, and $\widehat{\mathbf{F}}$ are estimates of $\mathbf{H}$, $\mathbf{C}$, and $\mathbf{F}$, respectively, $\mathbf{T}(\tilde{\mathbf{v}}') = \begin{bmatrix} l_u & 0 \\ 0 & l_\omega \end{bmatrix} \begin{bmatrix} \tanh\left(\frac{k_u}{l_u} i\tilde{u}\right) \\ \tanh\left(\frac{k_\omega}{l_\omega}\tilde{\omega}\right) \end{bmatrix}$, $k_u > 0$ and $k_\omega > 0$ are gain constants, $l_u \in \mathbb{R}$ and $l_\omega \in \mathbb{R}$ are saturation constants, and $\tilde{\omega} = \omega_d - \omega$ and $\tilde{u} = u_d - u$ are the current velocity errors. The term $\mathbf{T}(\tilde{\mathbf{v}}')$ provides a saturation in order to guarantee that the commands to be sent to the robot are always below the corresponding physical limits, considering that $\mathbf{v'_d}$ and $\dot{\mathbf{v}}'_\mathbf{d}$ are bounded to appropriate values.

In this chapter, we consider that the dynamic parameters are exactly known, that is, $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}$. This means that $\widehat{\mathbf{H}} = \mathbf{H}$, $\widehat{\mathbf{C}} = \mathbf{C}$, and $\widehat{\mathbf{F}} = \mathbf{F}$. The analysis considering parametric error is presented in [7, 21].

Using the Lyapunov candidate function $V = \frac{1}{2}\tilde{\mathbf{v}}'^{\mathbf{T}}\mathbf{H}\tilde{\mathbf{v}}' > 0$, and considering that the dynamic parameters are constant, one has

$$\dot{V} = -\tilde{\mathbf{v}}'^{\mathbf{T}}\mathbf{H}\mathbf{T}(\tilde{\mathbf{v}}') - \tilde{\mathbf{v}}'^{\mathbf{T}}\mathbf{C}\tilde{\mathbf{v}}' - \tilde{\mathbf{v}}'^{\mathbf{T}}\mathbf{F}\tilde{\mathbf{v}}'. \tag{26}$$

Observing property 5, of antisymmetry of $\mathbf{C}$, the derivative of the Lyapunov function can be written as

$$\dot{V} = -\tilde{\mathbf{v}}'^{\mathbf{T}}\mathbf{H}\mathbf{T}(\tilde{\mathbf{v}}') - \tilde{\mathbf{v}}'^{\mathbf{T}}\mathbf{F}\tilde{\mathbf{v}}'. \tag{27}$$

According to Property 1, $\mathbf{H}$ is symmetric and positive definite. The terms of $\mathbf{T}(\tilde{\mathbf{v}}')$ have the same sign of the terms of $\tilde{\mathbf{v}}'$. Property 3 states that $\mathbf{F}$ is symmetric and positive definite if $\theta_6 > -(\theta_5/I - \theta_3)Iu$, condition that was shown to hold for our robot. Therefore, one can conclude that $\dot{V} < 0$, that is, $\tilde{\mathbf{v}}' \in L_\infty$ and $\tilde{\mathbf{v}}' \to \mathbf{0}$ with $t \to \infty$, and $\tilde{\mathbf{v}} \in L_\infty$ and $\tilde{\mathbf{v}} \to \mathbf{0}$ with $t \to \infty$.

Regarding the kinematic controller, it has been shown [7] that a sufficient condition for the asymptotic stability is

$$\|\tilde{\mathbf{h}}\| > \frac{\|\mathbf{A}\tilde{\mathbf{v}}\|}{min}(k_x, k_y), \tag{28}$$

where $\mathbf{A} = \begin{bmatrix} \cos\psi & -a\sin\psi \\ \sin\psi & a\cos\psi \end{bmatrix}$. Because $\tilde{\mathbf{v}}(t) \to \mathbf{0}$, the condition (28) is asymptotically verified for any value of $\tilde{\mathbf{h}}$. Consequently, the tracking control error $\tilde{\mathbf{h}}(t) \to \mathbf{0}$, thus accomplishing the control objective.

To sum up, by using a control structure as shown in **Figure 2** with a dynamic compensation controller given by Eq. (25), different motion controllers can be applied. In our example, the trajectory tracking controller given by Eq. (20) was used. This is the system that we have implemented and for which we present some experimental results in Section 5.

5. Experimental results

In this section, we present some experimental results using a Pioneer 3-DX, from Adept Mobile Robots. In all experiments, the robot starts at position $(0.0, 0.0)m$ with orientation $0°$, and should follow an 8-shape trajectory also starting at $(0.0, 0.0)m$. The trajectory to be followed by the robot is represented by a sequence of desired positions $\mathbf{h_d}$ and velocities $\dot{\mathbf{h}}_\mathbf{d}$, both varying in time. The reference path is illustrated in **Figure 3**.

We have implemented the control structure shown on **Figure 2** using the control laws given by Eqs. (20) and (25). In total, we have executed 10 experiments for each controller, from now on referred to as KC (kinematics controller) and DC (dynamic compensation). In the case of KC, the robot receives as commands the values $\mathbf{v_d}$ calculated by the kinematics controller and there is no dynamic compensation. On the other hand, in the case of DC, the dynamic compensation controller is active and the robot receives as commands the values of $\mathbf{v_r}$ calculated by the dynamic compensation controller. We have repeated the experiments for four cases: KC with load, KC without load, DC with load, and DC without load. The load consists of a weight of 24.8 kg placed on top of the robot, while the original weight of the robot is 10.4 kg.

The following parameters were used in all experiments: $a = 0.15$ m, sample time of 0.1 s (this is the sample time of the Pioneer 3-DX); controller gains $k_x = 0.1$, $k_y = 0.1$, $k_u = 4$, $k_w = 4$, and saturation constants $l_x = 0.1$, $l_y = 0.1$, $l_u = 1$, $l_w = 1$. The robot used in the experiments is a Pioneer 3-DX without LASER scanner, therefore the parameters used in the dynamic compensation controller are the ones in column $P3$ from **Table 1**.

http://dx.doi.org/10.5772/intechopen.79397

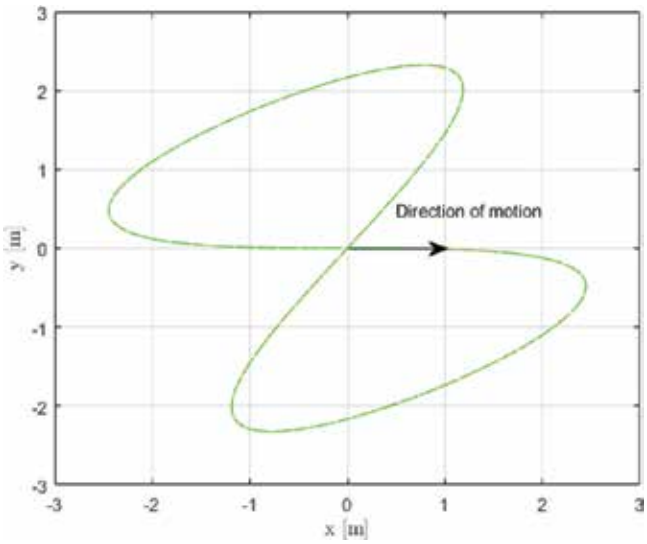


Figure 3. 8-shape reference path to be followed by the robot. Initial reference position is $(0.0, 0.0)m$ and the direction of motion is indicated in the figure. The robot starts at position $(0.0, 0.0)m$ with orientation 0°.

Figure 4 illustrates the results of 2 experiments, both without load. **Figure 4(a)** shows the 8-shape path followed by the robot without load when controlled by KC and DC. Robot path was recovered through its odometry. One can notice that the path followed by the robot is slightly different under KC or DC. The robot's linear and angular velocities also change along the path, as shown in **Figure 4(b)**.

A better visualization of the tracking error is given by **Figure 5**, which shows the evolution of the distance error during the experiments without load. The distance error is defined as the

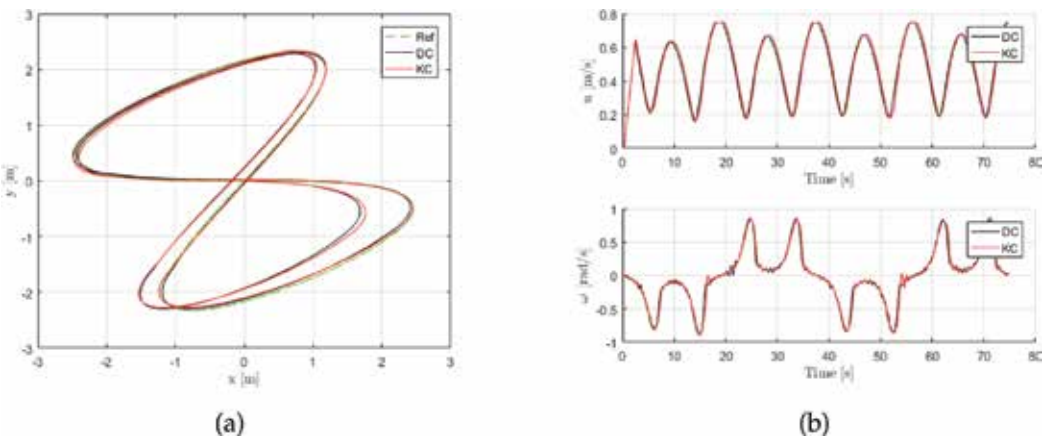


Figure 4. Experiments without load: (a) robot path; (b) linear and angular velocities. In all graphs, the black line represents the results for the case in which the dynamic compensation (DC) is active, while the red line represents the results for the kinematic controller (KC).

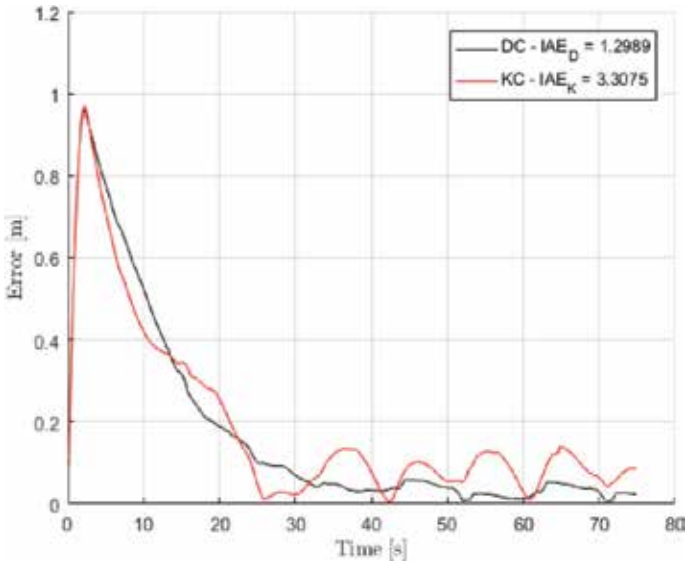


Figure 5. Evolution of tracking error without load. The black line represents the error for the case in which the dynamic compensation (DC) is active, while the red line represents the error for the kinematic controller (KC). The corresponding values of IAE_{30} for this experiment are also shown in the figure.

instantaneous distance between the desired position $\mathbf{h_d}$ and the actual robot position $\mathbf{h}$. It can be noticed that the distance error is similar for KC and DC in the first part of the path. At the beginning of the experiment, the tracking error increases quite a lot, reaching almost 1.0 m. This happens because the robot needs to accelerate from zero to catch up with the reference trajectory. After a few seconds, the error starts to decrease and around $25 - 30$ s, the robot follows the trajectory at normal speed. From this point on, it is clear that the average error is smaller when the DC is active.

Figure 6(a) shows the 8-shape path followed by the robot when carrying the load and controlled by KC and DC. One can notice that the path followed by the robot is slightly different under KC or DC, and there is more distortion in the path when compared to the case in which the robot carries no load. The robot's linear and angular velocities also change along the path, as shown in **Figure 6(b)**, and are very similar to the previous case.

The tracking error is given by **Figure 7**, which shows the evolution of the distance error during the experiments with load. As before, the robot needs to accelerate from zero to catch up with the reference trajectory, which causes the tracking error to increase in the first part of the experiments. But, in this case, the error in the first part of the experiment is actually higher for DC. This happens because the dynamic parameters used in the dynamic compensation controller remained unchanged during all experiments, with and without load. This means that the case in which the robot is carrying load is unfavorable for the dynamic compensation controller because the dynamics is not properly compensated, causing the error to increase. Even so, after about 30 s, the tracking error of DC gets smaller than the error for KC.

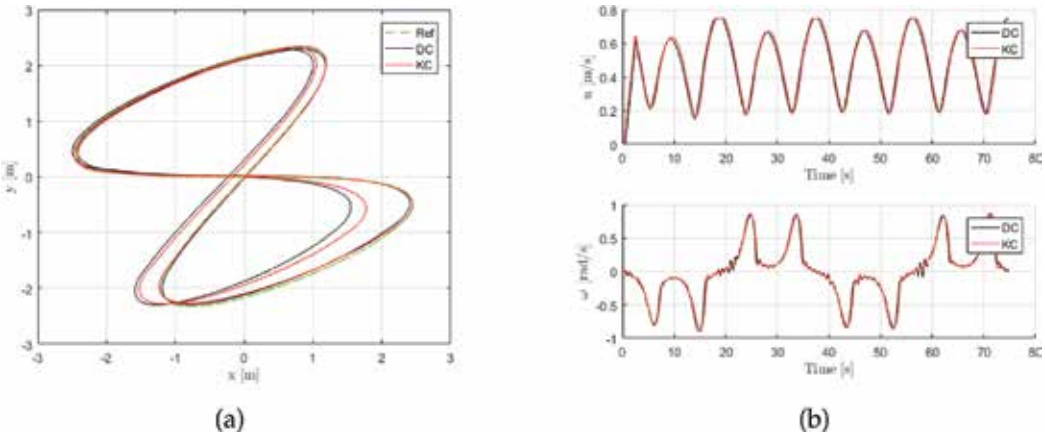


Figure 6. Experiments with load: (a) robot path; (b) linear and angular velocities. In all graphs the black line represents the results for the case in which the dynamic compensation (DC) is active, while the red line represents the results for the kinematic controller (KC).

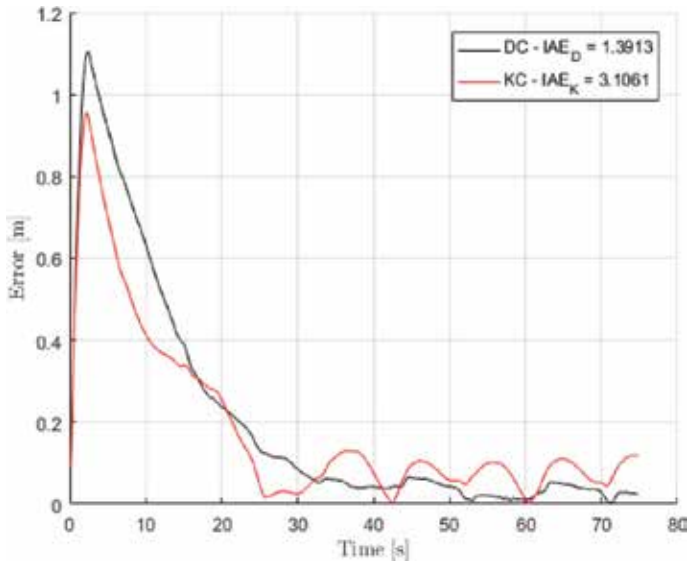


Figure 7. Evolution of tracking error with load. The black line represents the error for the case in which the dynamic compensation (DC) is active, while the red line represents the error for the kinematic controller (KC). The corresponding values of IAE_{30} for this experiment are also shown in the figure.

To evaluate the performance of the system we have calculated the *IAE* performance index, where $IAE = \int_{t_1}^{t_2} E(t)dt$, $E(t) = \sqrt{\tilde{x}^2 + \tilde{y}^2}$ is the instantaneous distance error and $t_2 - t_1$ is the period of integration. The average and standard deviation values of *IAE* for all experiments are reported in **Table 2**. There, IAE_{tot} was calculated considering $t_2 = 75$ s and $t_1 = 0$, that is, for the total period of each experiment. By its turn, the value of IAE_{30} was calculated only for the

	With load		**Without load**	
	IAE_{tot}	IAE_{30}	IAE_{tot}	IAE_{30}
Kinematic controller	14.30 ± 0.66	3.10 ± 0.05	14.08 ± 0.18	3.11 ± 0.12
Dynamic compensation	15.39 ± 0.42	$\mathbf{1.41 \pm 0.13}$	$\mathbf{13.05 \pm 0.07}$	$\mathbf{1.29 \pm 0.02}$

Here, $IAE = \int_{t_1}^{t_2} |E(t)|dt$, $E(t) = \sqrt{\tilde{x}^2 + \tilde{y}^2}$ is the instantaneous distance error, and $t_2 - t_1$ is the period of integration. For IAE_{tot}, $t_2 = 75$ s and $t_1 = 0$. For IAE_{30}, $t_2 = 75$ s and $t_1 = 35$ s.

Table 2. Average and standard deviation of *IAE* performance index calculated for experiments with and without load (lower value is better, highlighted in bold).

final 30 seconds of each experiment, that is, considering $t_2 = 75$s and $t_1 = 35$s. Therefore, IAE_{30} gives a good indication of the performance of the system after the error due to initial acceleration have faded out. From the results highlighted in bold in **Table 2**, it is clear that the performance of the system with the dynamic compensation controller is better in the long run because the correspondent values of IAE_{30} are about 50% of those for the kinematic controller. This is true even for the case in which the robot is carrying load.

It is important to emphasize that the dynamic parameters used in the dynamic compensation controller remained unchanged during all experiments, which means that the dynamics is not properly compensated when carrying load. This is illustrated by the fact that IAE_{tot} is bigger when the dynamic compensation is active and the robot is carrying load. Even so, in our experiments the performance was better in the long run when the dynamic compensation controller remained active.

One should notice that an increase in controller gains k_x and k_y could result in better performance (smaller tracking error), especially when the robot is carrying load. Nevertheless, we kept the same values of controller gains during all experiments to be able to compare the results.

6. Conclusion

In this chapter, we illustrate that the performance (in term of *IAE*) of a motion control system for a mobile robot can be up to 50% better under certain conditions when dynamic compensation is included. Such dynamic compensation can be implemented as shown in **Figure 2**, in which Eq. (25) is used with parameters identified via the procedure described in Section 3.2.

It is worth mentioning that the values of controller gains used in the experiments here reported were not optimum. The values of the gains were chosen empirically so that we could compare different cases. Optimization of controller gains can be executed to reduce tracking error, energy consumption, or a weighted combination of both, as shown in [25]. This means that the performance of the overall system could potentially be better than reported here.

We also presented a formulation of a dynamic model for differential-drive mobile robots, and discussed its mathematical properties. When compared to the classical dynamic model based on torques, the model used in this chapter has the advantages of accepting velocities as inputs,

and modeling the dynamics or the robot's actuators. We have shown that such model and its properties are useful on the design and stability analysis of a dynamic compensation controller for a differential-drive mobile robot. Moreover, because the mathematical structure of (12) is similar to the classical torque-based model, classical strategies for controller design [8, 26] can be adapted for designing controllers for mobile robots using the model presented in this chapter.

The dynamic model presented in this chapter can be used in connection with other kinematic controllers designed for commercial mobile robots, even in the context of coordinated control of multi-robot formations [27]. This integration requires no change on the original controller equations since the dynamic model accepts the same velocity commands as commercial robots. We invite the interested reader to download our toolbox for MATLAB/Simulink®, which include blocks to simulate the differential-drive kinematics and dynamics, a kinematic controller and two dynamic compensation controllers, one of which being the one presented in this chapter [28].

Acknowledgements

The authors thank the Institute of Engineering, Hanze University of Applied Sciences, for the partial financial support given for the publication of this chapter.

Author details

Felipe Nascimento Martins[1]* and Alexandre Santos Brandão[2]

*Address all correspondence to: fe.nascimento.martins@pl.hanze.nl

1 Institute of Engineering, Hanze University of Applied Sciences, Assen, The Netherlands

2 Department of Electrical Engineering, Federal University of Viçosa, Viçosa, Brazil

References

[1] Siegwart R, Nourbakhsh IR, Scaramuzza D. Introduction to Autonomous Mobile Robots. 2nd ed. London, England: MIT Press; 2011

[2] Birk A, Kenn H. RoboGuard, a teleoperated mobile security robot. Control Engineering Practice. 2002;**10**:1259-1264

[3] Prassler E et al. A short history of cleaning robots. Autonomous Robots. 2000;**9**(3):211-226

[4] Stouten B, de Graaf AJ. Cooperative transportation of a large object-development of an industrial application. In: IEEE International Conference on Robotics and Automation. Vol. 3. 2004. pp. 2450-2455. New Orleans, LA, USA: IEEE; DOI: 10.1109/ROBOT.2004.1307428

[5] Andaluz VH et al. Robust control with dynamic compensation for human-wheelchair system. In: Intelligent Robotics and Applications. Switzerland: Springer; 2014. pp. 376-389

[6] Morin P, Samson C. Motion control of wheeled mobile robots. In: Springer Handbook of Robotics. Heidelberg, Germany: Springer; 2008. pp. 799-826

[7] Martins FN et al. An adaptive dynamic controller for autonomous mobile robot trajectory tracking. Control Engineering Practice. 2008;**16**:1354-1363. DOI: 10.1016/j.conengprac.2008.03.004

[8] Fierro R, Lewis FL. Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics. Journal of Robotic Systems. 1997;**14**(3):149-163

[9] Das T, Kar IN. Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots. IEEE Transactions on Control Systems Technology. 2006;**14**(3):501-510

[10] Lapierre L, Zapata R, Lepinay P. Combined path-following and obstacle avoidance control of a wheeled robot. The Int. Journal of Robotics Research. 2007;**26**(4):361-375

[11] Shojaei K, Mohammad Shahri A, Tarakameh A. Adaptive feedback linearizing control of nonholonomic wheeled mobile robots in presence of parametric and nonparametric uncertainties. Robotics and Computer-Integrated Manufacturing. 2011;**27**(1):194-204

[12] Wang K. Near-optimal tracking control of a Nonholonomic mobile robot with uncertainties. International Journal of Advanced Robotic Systems. 2012;**9**(66):1-11

[13] Do KD. Bounded controllers for global path tracking control of unicycle-type mobile robots. Robotics and Autonomous Systems. 2013;**61**(8):775-784. ISSN: 0921-8890. https://doi.org/10.1016/j.robot.2013.04.014

[14] Onat A, Ozkan M. Dynamic adaptive trajectory tracking control of nonholonomic mobile robots using multiple models approach. Advanced Robotics. 2015;**29**(14):913-928

[15] Antonini P, Ippoliti G, Longhi S. Learning control of mobile robots using a multiprocessor system. Control Engineering Practice. 2006;**14**:1279-1295

[16] De La Cruz C, Carelli R. Dynamic model based formation control and obstacle avoidance of multi-robot systems. Robotica. 2008;**26**(03):345-356

[17] Chen CY et al. Design and implementation of an adaptive sliding-mode dynamic controller for wheeled mobile robots. Mechatronics. 2009;**19**(2):156-166. ISSN: 0957-4158

[18] De La Cruz C, Celeste WC, Bastos-Filho TF. A robust navigation system for robotic wheelchairs. Control Engineering Practice. 2011;**19**(6):575-590

[19] Rossomando FG, Soria C, Carelli R. Adaptive neural sliding mode compensator for a class of nonlinear systems with unmodeled uncertainties. Engineering Applications of Artificial Intelligence. 2013;**26**(10):2251-2259

[20] Utstumo T, Berge TW, Gravdahl JT. Non-linear model predictive control for constrained robot navigation in row crops. In: IEEE International Conference on Industrial Technology (ICIT 2015). Seville, Spain: IEEE; 2015. DOI: 10.1109/ICIT.2015.7125124

[21] Martins FN, Sarcinelli-Filho M, Carelli R. A velocity-based dynamic model and its properties for differential drive mobile robots. Journal of Intelligent & Robotic Systems. 2017; **85**(2):277-292

[22] Fukao T, Nakagawa H, Adachi N. Adaptive tracking control of a Nonholonomic mobile robot. IEEE Transactions on Robotics and Automation. 2000;**16**(5):609-615

[23] Kim MS, Shin JH, Lee JJ. Design of a robust adaptive controller for a mobile robot. In: Proc. of the IEEE/RSJ Int. Conf. On Intelligent Robots and Systems. Vol. 3. Takamatsu, Japan: IEEE; 2000. pp. 1816-1821. DOI: 10.1109/IROS.2000.895235

[24] Laut J. A dynamic parameter identification method for migrating control strategies between heterogeneous wheeled mobile robots. [PhD thesis]. Worcester Polytechnic Institute; 2011

[25] Martins FN, Almeida GM. Tuning a velocity-based dynamic controller for unicycle mobile robots with genetic algorithm. In: Joranadas Argentinas de Robótica. Olavarría, Argentina: Universidad Nacional del Centro de la Provincia de Buenos Aires; 2012

[26] Spong MW, Hutchinson S, Vidyasagar M. Robot Modeling and Control. Vol. 3. New York: Wiley; 2006

[27] Brandao AS et al. A multi-layer control scheme for multi-robot formations with adaptive dynamic compensation. In: IEEE International Conference on Mechatronics, 2009. ICM 2009. Malaga, Spain: IEEE; 2009

[28] F.N. Martins. Velocity-based Dynamic Model and Adaptive Controller for Differential Steered Mobile Robot. 2017. Available from: http://www.mathworks.com/matlabcentral/fileexchange/44850

Chapter 3

Theoretical and Experimental Collaborative Area Coverage Schemes Using Mobile Agents

Sotiris Papatheodorou and Anthony Tzes

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.78940

Abstract

This chapter is concerned with the development of collaborative control schemes for mobile ground robots for area coverage purposes. The simplest scheme assumes point omnidirectional robots with heterogeneous circular sensing patterns. Using information from their spatial neighbors, each robot (agent) computes its cell relying on the power diagram partitioning. If there is uncertainty in inferring the locations of these robots, the Additively Weighted Guaranteed Voronoi scheme is employed resulting in a rather conservative performance. The aforementioned schemes are enhanced by using a Voronoi-free coverage scheme that relies on the knowledge of any arbitrary sensing pattern employed by the agents. Experimental results are offered to highlight the efficiency of the suggested control laws.

Keywords: area coverage, multiagent systems, mobile robot systems, distributed control, cooperative control

1. Introduction

The problem of area coverage is one that has been widely studied in the past decade and consists of the deployment of a sensor-equipped mobile robot team. It is usually categorized as either blanket or sweep coverage. In blanket or static coverage the goal of the robot team is a final static configuration at which an objective function is maximized [1–3]. In sweep or dynamic coverage on the other hand the mobile agents are tasked with maximizing a constantly changing objective, resulting in potentially continuous motion of the agents [4–6].

Several aspects of the area coverage problem have been studied over the years, including the effect of robot dynamics [7–9], communication constraints among agents [10–12], complex

non-convex regions [13–15] or guaranteeing collision avoidance among the mobile robots [16, 17]. A wide variety of methods has also been employed for multirobot area coverage such as geometric optimization [18], optimal control [19] or event-triggered control [20]. Due to the widespread adoption of unmanned aerial vehicles (UAVs), they have become a popular platform for area coverage [21–23] since they are usually equipped with visual sensors [24–26].

In this chapter we focus on the blanket coverage problem for a convex region of interest. The techniques outlined are based on geometric optimization principles and result in distributed control schemes. In a distributed control law, each agent uses only local information from its neighboring agents in order to compute its own control input so that a common objective function is maximized. Distributed control laws are highly desirable in multiagent systems because they are easily scalable to large robot teams and because they significantly reduce the computational burden and communication requirements on the agents. Moreover, they are more robust to failures and can adapt to unexpected changes without the need to recompute a new solution as is the case with most centralized control schemes.

The chapter is organized as follows. Section 2.1 contains some mathematical preliminaries which will be relevant throughout the chapter. In Section 2.2 the problem of blanket area coverage in a convex region by a heterogeneous team of agents with omnidirectional sensors is examined. In Section 2.3 the results are extended by taking into account the uncertain positioning of the mobile robots. Section 2.4 presents a tessellation-free method for area coverage by agents with anisotropic sensing patterns. Section 2.5 contains some experimental results and it is followed by concluding remarks.

2. Area coverage using mobile agents

2.1. Mathematical preliminaries

Throughout the chapter we assume a compact, convex region $\Omega \subset \mathbb{R}^2$ to be covered by the mobile agents and a space density function $\phi : \Omega \to \mathbb{R}_+$. The space density function is used to encode any a priori information regarding the importance of points in Ω, for example the likelihood that an event may occur at a given point. The boundary of a set S is denoted ∂S and its interior is denoted $Int(S)$. The set $\{1, \ldots, n\}$ is denoted I_n. The indicator function $\mathbf{1}_S(q)$ for a set S and the 2×2 rotation matrix $\mathbf{R}(\theta)$ are respectively

$$\mathbf{1}_S(q) = \begin{cases} 1 & \text{if } q \in S \\ 0 & \text{if } q \notin S \end{cases}, \qquad \mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix},$$

while the 2×2 identity matrix is denoted $\mathbb{I}_2$.

2.2. Heterogeneous agents with omnidirectional sensing

One of the simplest variants of the area coverage problem is the case of a team of homogeneous agents with circular sensing footprints. This was one of the first variants to be studied and

there is extensive work on the topic [27, 28]. One generalization of this problem arises by allowing each agent to have a different sensing performance, resulting in a heterogeneous team [29–31]. In this chapter we will focus in the coverage of a convex region by a team of unicycle robots equipped with heterogeneous omnidirectional sensors.

2.2.1. Problem statement

A team of n mobile ground agents is deployed inside the region of interest Ω. Each agent $i \in I_n$ is approximated by a point mass located at $q_i \in \Omega$ which is governed by the following kinematic model

$$\dot{q}_i = u_i, \quad q \in \Omega, \quad u_i \in \mathbb{R}^2 \tag{1}$$

where u_i is the velocity control input of the agent.

Each agent has been equipped with an omnidirectional sensor with limited sensing radius R_i, which is allowed to differ among agents, resulting in a circular sensing pattern

$$S_i(q_i, R_i) = \{q \in \Omega : \quad \|q - q_i\| \ \leq \ R_i\}. \tag{2}$$

Since the goal of the mobile agent team is the maximization of the covered area using their sensors, while also taking into account the space density function, we define the coverage objective as

$$\mathscr{H} = \int_{\Omega} \max_{i \in I_n} \ \mathbf{1}_{S_i}(q) \ \phi(q) \ dq. \tag{3}$$

The control objective is the design of a distributed control law for the mobile agents in order to guarantee monotonic increase of the coverage objective $\mathscr{H}$ over time.

2.2.2. Space partitioning

The first step in designing a distributed control law is finding a method to distribute the computation of the coverage objective $\mathscr{H}$. Due to the heterogeneous sensing performance of the agents, the Voronoi diagram is inadequate for the task as it does not take this information into account. To that extent the power diagram will be used in order to assign a region of responsibility to each agent. In contrast to the Voronoi diagram whose generators are points, the generators of the power diagram are disks.

Given a planar region Ω and a set of disks $S = \{S_1, \ldots, S_n\}$ with centers $Q = \{q_1, \ldots, q_n\}$ and radii $R = \{R_1, \ldots, R_n\}$, the power diagram assigns a convex cell $P_i \subseteq \Omega$ to each disk S_i

$$P_i(\Omega, S) = \left\{q \in \Omega : \|q - q_i\|^2 - {R_i}^2 \ \leq \ \|q - q_j\|^2 - {R_j}^2, \ \forall j \in I_n \backslash i\right\}, \ i \in I_n.$$

The power diagram is a complete tessellation of Ω, thus it holds that

$$\Omega = \bigcup_{i \in I_n} P_i, \qquad Int(P_i) \cap Int(P_j) = \emptyset, \quad \forall i \neq j.$$

A notable property of power diagrams is their duality with power-weighted Delaunay graphs. It has been shown that in order to compute the power cell P_i of point q_i, only the power-weighted Delaunay neighbors N_i of point q_i need to be considered. The power-weighted Delaunay neighbors of agent i have the property that

$$N_i = \{j \in I_n \backslash i : P_i \cap P_j \neq \emptyset\}, \tag{4}$$

By using the previous definition, the power diagram can be formulated as

$$P_i(\Omega, Q) = \left\{q \in \Omega : \|q - q_i\|^2 - {R_i}^2 \ \leq \ \|q - q_j\|^2 - {R_j}^2, \ \ \forall j \in N_i\right\}, \ \ i \in I_n. \tag{5}$$

Since it holds that $N_i \subseteq I_n$, each agent i requires information only from its power-weighted Delaunay neighbors in order to compute its own power cell P_i, thus rendering the computation of the power diagram distributed.

Remark 2.1. *When the agents' sensing radii are equal, i.e., $R_i = R_j$, $\forall i, j$, the power diagram converges to the Voronoi diagram. In that case the computation of the cell of agent i requires information only from the Delaunay neighbors of agent i. Thus the power diagram can be also utilized in the case of agents with homogeneous sensing performance.*

For any two agents i and j with $S_i \cap S_j \neq \emptyset$ it holds that $S_i \backslash P_i \in P_j \cap S_j$ and $S_j \backslash P_j \in P_i \cap S_i$ due to the properties of the power diagram. Thus if a part of some agent i 's sensing pattern is inside the cell of some other agent j, then that part is guaranteed to be sensed by j. Consequently, we define the r-limited power cell of agent i as $P_i^R = P_i \cap S_i$. Thus by utilizing the power diagram, the coverage objective $\mathscr{H}$ can be computed as follows

$$\mathscr{H} = \sum_{i \in I_n} \int_{P_i^R} \phi(q) \ dq. \tag{6}$$

Since $\mathscr{H}$ can be written as a sum of integrals over r-limited power cells and since an agent can compute its own power cell using information just from its power-weighted Delaunay neighbors, the computation of $\mathscr{H}$ is distributed.

2.2.3. Control law formulation

Having found a partitioning scheme that allows distributed computation of the coverage objective $\mathscr{H}$, what is left is the derivation of a distributed control law for its maximization.

Theorem 2.1. *For a team of mobile ground agents with kinematics* (1)*, sensing performance* (2) *and using the power diagram partitioning* (5)*, the control law*

$$u_i = \alpha_i \int_{\partial P_i^R \cap \partial S_i} n_i \ \phi(q) \ \mathrm{d}q, \tag{7}$$

where α_i is a positive constant and n_i is the outward unit normal vector on ∂P_i^R, leading the agents to trajectories that result in monotonic increase of the coverage objective (6).

Proof. We start by evaluating the time derivative of the objective using the agent dynamics (1) we get $\frac{\partial \mathscr{H}}{\partial t} = \sum_{i \in I_n} \frac{\partial \mathscr{H}}{\partial q_i} \frac{\partial q_i}{\partial t} = \sum_{i \in I_n} \frac{\partial \mathscr{H}}{\partial q_i} \dot{q}_i = \sum_{i \in I_n} \frac{\partial \mathscr{H}}{\partial q_i} u_i$. By selecting the control law $u_i = \alpha_i \frac{\partial \mathscr{H}}{\partial q_i}$, $\alpha_i > 0$, we can guarantee monotonic increase of the coverage objective.

The partial derivative $\frac{\partial \mathscr{H}}{\partial q_i}$ is evaluated as follows

$$\frac{\partial \mathscr{H}}{\partial q_i} = \frac{\partial}{\partial q_i} \sum_{i \in I_n} \int_{P_i^R} \phi(q) \ \mathrm{d}q = \frac{\partial}{\partial q_i} \int_{P_i^R} \phi(q) \ \mathrm{d}q \ + \sum_{j \neq i} \frac{\partial}{\partial q_i} \int_{P_j^R} \phi(q) \ \mathrm{d}q.$$

Since only the cells of power-weighted Delaunay neighbors of agent i are affected by its movement and $\frac{\partial \phi(q)}{\partial q_i} = 0$, by using the Leibniz integral rule [32], the previous equation becomes

$$\frac{\partial \mathscr{H}}{\partial q_i} = \int_{\partial P_i^R} v_i^i \ n_i \ \phi(q) \ \mathrm{d}q \ + \sum_{j \in N_i} \int_{\partial P_j^R} v_j^i \ n_j \ \phi(q) \ \mathrm{d}q$$

where v_j^i is the Jacobian matrix $v_j^i = \frac{\partial q}{\partial q_i}$, $q \in \partial P_j^R$ and n_i is the outward unit normal vector on ∂P_i^R. The boundary ∂P_i^R can be decomposed into three disjoint sets $\partial P_i^R = \left(\partial P_i^R \cap \partial S_i\right) \cup \left(\partial P_i^R \cap \partial \Omega\right) \cup \left[\underset{j \in N_i}{\cup} \left(\partial P_i^R \cap \partial P_j^R\right)\right]$, where $\partial P_i^R \cap \partial S_i$ denotes part of the r-limited cell's boundary that is also part of the boundary of the agent's sensing disk, $\partial P_i^R \cap \partial \Omega$ denotes the common boundary between the r-limited cell and the region, while $\partial P_i^R \cap \partial P_j^R$ denotes the common boundary with the r-limited cell of some neighboring agent j. This decomposition is presented in **Figure 1** where $\partial P_i^R \cap \partial S_i$, $\partial P_i^R \cap \partial \Omega$ and $\partial P_i^R \cap P_j^R$ are shown in solid green, red, and blue lines, respectively.

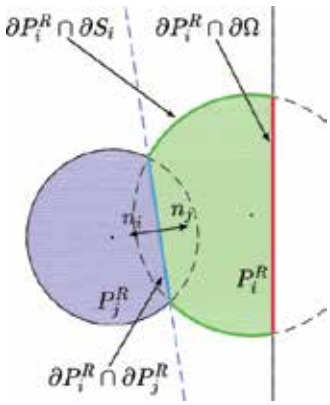


Figure 1. Decomposition of ∂P_i^R into disjoint sets and corresponding normal vectors.

Since the region Ω is assumed to be static, it holds that $v_i^i = 0, \forall q \in \partial P_i^R \cap \partial\Omega$. In addition, since $q \in \partial P_i^R \cap \partial S_i$ are points on a circle with center q_i, it holds that $v_i^i = \mathbb{I}_2, \forall q \in \partial P_i^R \cap \partial S_i$. Finally, P_j^R is only affected by the movement of agent i at the common boundary $\partial P_i^R \cap \partial P_j^R$, resulting in the expression

$$\frac{\partial \mathcal{H}}{\partial q_i} = \int\limits_{\partial P_i^R \cap \partial S_i} n_i \ \phi(q) \ \mathrm{d}q \ + \sum_{j \in N_i} \int\limits_{\partial P_i^R \cap \partial P_j^R} v_i^i \ n_i \ \phi(q) \ \mathrm{d}q \ + \sum_{j \in N_i} \int\limits_{\partial P_i^R \cap \partial P_j^R} v_j^i \ n_j \ \phi(q) \ \mathrm{d}q.$$

Since $v_i^i = v_j^i$ and $n_i = -n_j$ on the common boundary $\partial P_i^R \cap \partial P_j^R$, as shown in **Figure 1**, the two sums in the previous expression cancel out and by multiplying it with α_i we get (7). □

2.2.4. Simulation study I

An indicative simulation is presented in this section. The region Ω is chosen as the convex polygon defined by the vertices with x and y coordinates

$$\Omega_x = [0.5, 0.5, 0.45, 0.4, -0.46, -0.5, -0.48, -0.34, 0.05],$$
$$\Omega_y = [0.43, 0.2, -0.3, -0.5, -0.44, -0.1, 0.37, 0.47, 0.5]$$

respectively. The space density function was $\phi(q) = 1, \forall q \in \Omega$. A team of eight agents with heterogeneous sensing radii is deployed inside the region.

The initial and final agent configurations are shown in **Figure 2a** and **c** respectively where the agent positions are marked by black dots, the boundaries of their sensing disks are shown as dashed black lines, the boundaries of their cells are marked by solid black lines while their interiors are filled in color. The agent trajectories are shown in **Figure 2b** with the initial positions marked by dots and the final positions by circles. It is observed that the agents are successfully deployed inside the region, increasing the covered area in the process. In order to provide a more objective measure of the agents' performance, two different metrics are used. The first, denoted $\mathcal{H}^r$, is the value of the coverage objective $\mathcal{H}$ as a percentage of the objective over the whole region which in the case where $\phi(q) = 1, \forall q \in \Omega$ it is equal to the area of Ω. This

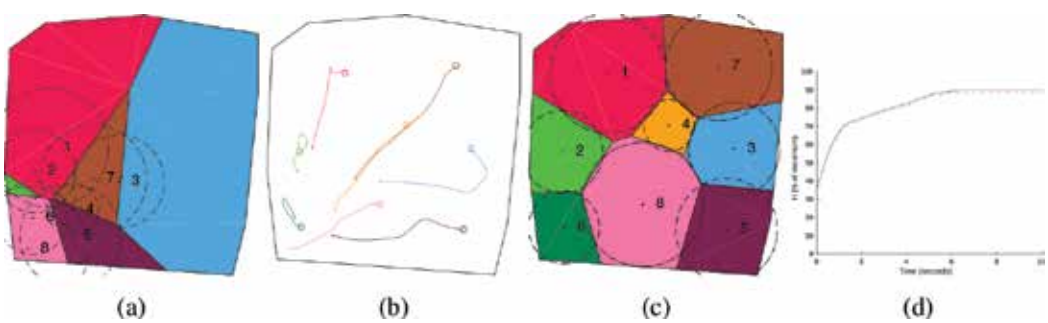


Figure 2. Simulation study I: (a) initial configuration, (b) agent trajectories, (c) final configuration and (d) evolution of the coverage objective over time.

metric indicates to what extent the agents cover the region Ω, with high values of $\mathscr{H}^r$ corresponding to high coverage over Ω. The second metric, denoted $\mathscr{H}^a$, is the value of the coverage objective $\mathscr{H}$ as a percentage of the agents' maximum possible covered area which is only meaningful in the case where $\phi(q) = 1, \forall q \in \Omega$. This metric indicates to what extent the agents' sensors are utilized, with high values of $\mathscr{H}^a$ indicating that the agents' sensors are utilized close to their full capabilities. Low values of $\mathscr{H}^a$ simultaneously with high values of $\mathscr{H}^r$ indicate an overabundance of agents given the size of the current region Ω. The two metrics are more formally defined as

$$\mathscr{H}^r = 100\frac{\mathscr{H}}{\int\limits_{\Omega}\phi(q)\mathrm{d}q}, \qquad \mathscr{H}^a = 100\frac{\mathscr{H}}{\sum\limits_{i\in I_n}\int\limits_{S_i}\mathrm{d}q}. \tag{8}$$

Figure 2d shows $\mathscr{H}^a$ in solid blue and $\mathscr{H}^r$ in dashed red with their final values being 90.0 and 88.9% respectively, indicating that the final agent configuration is an efficient one.

2.3. Heterogeneous agents with omnidirectional sensing under positioning uncertainty

The inherent uncertainty in all localization systems' measurements can often create unexpected problems in algorithms designed with precise localization in mind. Consequently algorithms robust to positioning errors have been sought for the area coverage problem [33, 34]. This section presents an extension to the control law presented in [35] which allows for teams of agents with heterogeneous sensing performance.

2.3.1. Agent model

The agents' kinematic model is described by (1) and their sensing performance by (2). Due to the localization systems' inherent inaccuracy, we assume that q_i is the agent's position as reported by the localization system, while r_i is a known upper bound on the localization error. Thus we define the positioning uncertainty region U_i as follows

$$U_i(q_i, r_i) = \{q \in \mathbb{R}^2 : \ \|q - q_i\| \ \leq r_i\}, \tag{9}$$

which is a circular disk that contains all possible positions of agent i given its reported position q_i and positioning uncertainty upper bound r_i.

In order to take into account the agents' positioning uncertainty, we define for each agent the guaranteed sensed region S_i^g as

$$S_i^g(q_i, r_i, R_i) = \bigcap_{\forall q \in U_i} S_i(q, R_i), \tag{10}$$

which is the region that is guaranteed to be sensed by agent i given all of its possible positions within its positioning uncertainty region. Given the fact that both S_i and U_i are disks, the above expression can be simplified into

$$S_i^g(q_i, r_i, R_i) = \{q \in \mathbb{R}^2 : \ \|q - q_i\| \ \leq R_i^g = R_i - r_i\}. \tag{11}$$

2.3.2. Space partitioning and problem statement

In order to take into account the agents' positioning uncertainty as well as their heterogeneous sensing capabilities, the Additively Weighted Guaranteed Voronoi (AWGV) diagram is employed. The AWGV is an extension of the Guaranteed Voronoi (GV) diagram [36] that incorporates additive weights.

Given a planar region Ω, a set of uncertain regions $U = \{U_1, \ldots, U_n\}$ and a set of weights $R^g = \{R_1^g, \ldots, R_n^g\}$, the AWGV diagram assigns a convex cell $G_i \subseteq \Omega$ to each region-weight pair (U_i, R_i^g) as follows

$$G_i(\Omega, U, R^g) = \left\{ q \in \Omega : \max_{q \in U_i} \|q - q_i\| - R_i^g \leq \min_{q \in U_j} \|q - q_j\| - R_j^g, \ \forall j \in I_n \backslash i \right\}, \quad i \in I_n.$$

Contrary to the Voronoi diagram, the AWGV diagram is not a complete tessellation of the region Ω since a part of Ω is left unassigned. This part is called the neutral region $\mathscr{O}$ and it holds that

$$\Omega = \mathscr{O} \cup \bigcup_{i \in I_n} G_i. \tag{12}$$

A notable property of AWGV diagrams is their duality with additively weighted Delaunay graphs. It has been shown that in order to compute the AWGV cell G_i of the region-weight pair (U_i, R_i^g), only the additively weighted Delaunay neighbors N_i of (U_i, R_i^g) need to be considered. By using the previous definition, the Voronoi diagram can be formulated as

$$G_i(\Omega, U, R^g) = \left\{ q \in \Omega : \max_{q \in U_i} \|q - q_i\| - R_i^g \leq \min_{q \in U_j} \|q - q_j\| - R_j^g, \ \forall j \in N_i \backslash i \right\}, \quad i \in I_n. \tag{13}$$

Since it holds that $N_i \subseteq I_n$, each agent i requires information only from its additively weighted Delaunay neighbors in order to compute its own AWGV cell G_i, thus rendering the computation of the AWGV diagram distributed.

The previous definition of the AWGV can be written as $G_i = \bigcap_{j \in N_i} H_{ij}, \quad i \in I_n$, where $H_{ij} = \left\{ q \in \Omega : \|q - q_j\| - \|q - q_i\| \geq + r_i + r_j - R_i^g + R_j^g \right\}$. Thus the boundary ∂H_{ij} is one branch of a hyperbola with foci located at q_i and q_i and semi-major axis $a_{ij} = \frac{r_i + r_j - R_i^g + R_j^g}{2}$. Since the quantity a_{ij} may be either positive or negative, ∂H_{ij} may correspond to the 'East' or 'West' branch of the hyperbola, which results in the region H_{ij} being convex or non-convex respectively.

We define the r-limited AWGV cell of agent i as $G_i^R = G_i \cap S_i^g$. We now define the coverage objective as

$$\mathscr{H} = \sum_{i \in I_n} \int_{G_i^R} \phi(q) \ dq, \tag{14}$$

which is the area of the region that is guaranteed to be closest to and at the same time sensed by each agent. Since $\mathscr{H}$ is a sum of integrals over r-limited AWGV cells and since an agent can compute its own AWGV cell using information just from the agents in N_i, the computation of $\mathscr{H}$ is distributed.

2.3.3. Control law formulation

Since the computation of the coverage objective $\mathscr{H}$ is distributed due to the AWGV partitioning, what is left is the derivation of a distributed control law for its maximization.

Theorem 2.2. *For a team of mobile ground agents with kinematics* (1)*, sensing performance* (2)*, positioning uncertainty* (9) *and using the AWGV partitioning* (13)*, the control scheme*

$$u_i = \alpha_i \int\limits_{\partial G_i^R \cap \partial S_i^g} n_i \ \phi(q) \ \mathrm{d}q \ + \alpha_i \sum_{j \in N_i} \left[\int\limits_{\partial G_i^R \cap \partial H_{ij}} \mu_i^i \ n_i \ \phi(q) \ \mathrm{d}q \ + \int\limits_{\partial G_j^R \cap \partial H_{ji}} \mu_j^i \ n_j \ \phi(q) \ \mathrm{d}q \right] \quad (15)$$

where α_i is a positive constant, n_i the outward unit normal vector on ∂G_i^R, leads the agent to trajectories that result in monotonic increase of the coverage objective (14).

Proof. We start by evaluating the time derivative of the objective using the agent dynamics (1) as in the proof of Theorem 2.1 and by selecting the control law $u_i = \alpha_i \frac{\partial \mathscr{H}}{\partial q_i}$, $\alpha_i > 0$, we can guarantee monotonic increase of the coverage objective.

The partial derivative $\frac{\partial \mathscr{H}}{\partial q_i}$ is evaluated as follows

$$\frac{\partial \mathscr{H}}{\partial q_i} = \frac{\partial}{\partial q_i} \sum_{i \in I_n} \int\limits_{G_i^R} \phi(q) \ \mathrm{d}q = \frac{\partial}{\partial q_i} \int\limits_{G_i^R} \phi(q) \ \mathrm{d}q \ + \sum_{j \neq i} \frac{\partial}{\partial q_i} \int\limits_{G_j^R} \phi(q) \ \mathrm{d}q.$$

Since only the cells of additively weighted Delaunay neighbors of agent i are affected by its movement and $\frac{\partial \phi(q)}{\partial q_i} = 0$, the previous equation becomes

$$\frac{\partial \mathscr{H}}{\partial q_i} = \int\limits_{\partial G_i^R} \mu_i^i \ n_i \ \phi(q) \ \mathrm{d}q \ + \sum_{j \in N_i} \int\limits_{\partial G_j^R} \mu_j^i \ n_j \ \phi(q) \ \mathrm{d}q$$

where μ_j^i is the Jacobian matrix

$$\mu_j^i = \frac{\partial q}{\partial q_i}, \ \ q \in \partial G_j^R$$

and n_i is the outward unit normal vector on ∂G_i^R. The boundary ∂G_i^R can be decomposed into three disjoint sets as follows

$$\partial G_i^R = \left(\partial G_i^R \cap \partial S_i^g\right) \cup \left(\partial G_i^R \cap \partial \Omega\right) \cup \left[\bigcup_{j \in N_i} \left(\partial G_i^R \cap \partial H_{ij}\right)\right], \tag{16}$$

where $\partial G_i^R \cap \partial S_i^g$ denotes part of the r-limited cell's boundary that is also part of the boundary of the agent's sensing disk, $\partial G_i^R \cap \partial \Omega$ denotes the common boundary between the r-limited cell and the region, while $\partial G_i^R \cap \partial H_{ij}$ denotes parts of the boundary that consist of hyperbolic arcs induced by some neighboring agent j. This decomposition is presented in **Figure 3** where $\partial G_i^R \cap \partial S_i^g$, $\partial G_i^R \cap \partial \Omega$, $\partial G_i^R \cap \partial H_{ij}$ and $\partial G_j^R \cap \partial H_{ji}$ are shown in solid green, red, blue and purple lines respectively.

Since the region Ω is assumed to be static, it holds that $\mu_i^i = 0, \forall q \in \partial G_i^R \cap \partial \Omega$. In addition, since $q \in \partial G_i^R \cap \partial S_i^g$ are points on a circle with center q_i, it holds that $\mu_i^i = \mathbb{I}_2, \forall q \in \partial G_i^R \cap \partial S_i^g$. Finally, G_j^R is only affected by the movement of agent i at the induced hyperbolic arc $\partial G_j^R \cap \partial H_{ji}$ and by grouping the hyperbolic arcs in pairs and multiplying by α_i we get (15). □

2.3.4. Constraining agents inside the region

When the control law (15) is used, there can be cases where the positioning uncertainty regions of some agent does not remain entirely inside Ω, i.e. it is possible that $U_i \cap \Omega \neq U_i$ for some agent i. This has the implication that the control law (15) may lead some agent i outside the region Ω, given the fact that an agent may reside anywhere within its positioning uncertainty region U_i.

In order to avoid such a situation, a subset $\Omega_i^s \subseteq \Omega$ is used instead, instead of the region Ω. This subset Ω_i^s is in the general case different among agents due to their differing measures of positioning uncertainty r_i. This subset of Ω is computed as the Minkowski difference of Ω with the disk $U_i^0(r_i) = \{q \in \Omega : \ \|q\| \leq r_i\}$ which is $\Omega_i^s = \left\{q \in \Omega \ : \ q + U_i^0 \subseteq \Omega\right\}, i \in I_n$.

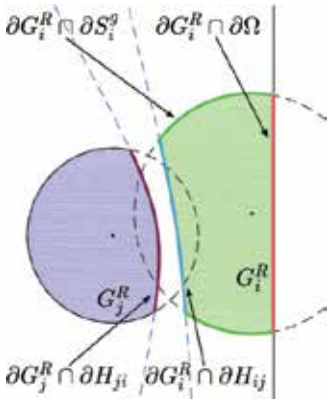


Figure 3. Decomposition of ∂G_i^R into disjoint sets and corresponding normal vectors.

By using this subset Ω_i^s, constraining agents inside Ω is simpler, since this is equivalent to constraining each agent's reported position q_i inside its respective subset region Ω_i^s. This is achieved by stopping an agent if its reported position q_i is located on the boundary of Ω_i^s and simultaneously its current control input leads the agent toward the exterior of Ω_i^s. Thus the control law being implemented is

$$\tilde{u}_i = \begin{cases} 0 & \text{if } q_i \in \partial\Omega_i^s \wedge q_i + \varepsilon u_i \notin \Omega_i^s \\ u_i & \text{otherwise} \end{cases} \tag{17}$$

where ε is an infinitesimally small positive constant.

2.3.5. Simulation study II

An indicative simulation is presented in this section. This simulation is identical to the one presented in Section 2.2.4 with the only difference being the addition of positioning uncertainty to the agents.

The initial and final agent configurations are shown in **Figure 4a** and **c** respectively where the agent positioning uncertainty regions are shown as black circles, the boundaries of their sensing disks are shown as dashed black lines, the boundaries of their cells are marked by solid black lines while their interiors are filled in color. The agent trajectories are shown in **Figure 4b** with the initial positions marked by dots and the final positions by circles. It is observed that the agents successfully deploy inside the region, increasing the covered area in the process. In order to provide a more objective measure of the agents' performance, the two metrics described in Section 2.2.4 are used which in the case of uncertainty positioning are more formally defined as

$$\mathcal{H}^r = 100\frac{\mathcal{H}}{\int_\Omega \phi(q)\mathrm{d}q}, \qquad \mathcal{H}^a = 100\frac{\mathcal{H}}{\sum_{i\in I_n}\int_{S_i^g}\mathrm{d}q}.$$

Figure 4d shows $\mathcal{H}^a$ in solid blue and $\mathcal{H}^r$ in dashed red with their final values being 94.0 and 70.0% respectively. In this simulation we observe that although $\mathcal{H}^a$ reaches a high value, this is

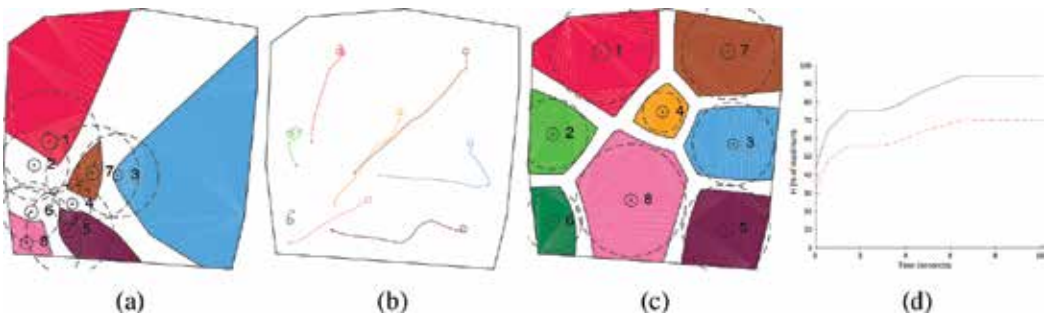


Figure 4. Simulation study II: (a) initial configuration, (b) agent trajectories, (c) final configuration and (d) evolution of the coverage objective over time.

not the case with $\mathscr{H}^r$. The first reason for this result is the fact that the computation of $\mathscr{H}$ is based on the agents' guaranteed sensing patterns S_i^g, which by definition are lower in area than their respective sensing patterns S_i. Moreover, due to the definition of $\mathscr{H}$ being conservative, only the area of the r-limited cells G_i^R counts toward the value of $\mathscr{H}$, thus parts of the neutral region $\mathscr{O}$ that are covered by the agents do not contribute to $\mathscr{H}$. Consequently, in the case of the AWGV partitioning (13), coverage objective (14) and control law (15), it is expected for $\mathscr{H}^r$ to achieve a lower value.

2.4. Heterogeneous agents with anisotropic sensing

Although the omnidirectional sensors examined in the previous two sections can significantly simplify the problem formulation and solution, they are usually inadequate for precise modeling of real-life sensors. For this reason there have been several differing approaches to area coverage using agents with anisotropic sensing patterns [37–40]. In this section we will follow the methodology presented in [41] which is a distributed optimization technique resulting in a gradient-based control law.

2.4.1. Problem statement

A team of n mobile ground agents is deployed inside the region of interest Ω. Given the anisotropic nature of the sensing patterns examined in this section, the mobile agents should be able to change their orientation as well as move around inside the region of interest. A realistic model for a mobile agent with the ability to rotate is that of the differential drive robot whose kinematic model is

$$\dot{q}_i = \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \frac{\rho_i}{2} \left(\Omega_i^R + \Omega_i^L\right), \quad q_i \in \Omega,$$
$$\dot{\theta}_i = \frac{\rho_i}{l_i} \left(\Omega_i^R - \Omega_i^L\right), \quad \theta_i \in [-\pi, \pi],$$

where Ω_i^R, Ω_i^L are the rotational velocities of the right and left wheels, respectively, ρ_i is the wheel radius, and l_i is the length of the wheel axis. In this chapter however a simpler single integrator kinematic model is used for the agents. Each agent $i \in I_n$ is approximated by a point mass located at $q_i \in \Omega$ with orientation θ_i which is governed by the kinematic model described by

$$\dot{q}_i = u_i, \quad q \in \Omega, \quad u_i \in \mathbb{R}^2, \tag{18}$$

$$\dot{\theta}_i = \omega_i, \quad \theta, \omega_i \in \mathbb{R}, \tag{19}$$

where ω_i is the rotational velocity control input of the agent. This single integrator model simplifies the derivation of the control law, although the control law can be extended for differential drive robots as well.

We define the base sensing pattern S_i^b of agent i as the region sensed by the agent when $q_i = [0, \ 0]^T$ and $\theta_i = 0$. The only requirements with regards to the base sensing pattern are

that $q_i \in Int\left(S_i^b\right)$ and that its boundary ∂S_i^b can be described by a set of parametric equations. Let the radius R_i of a base sensing pattern be defined as $R_i\left(S_i^b\right) = \max_{q \in \partial S_i^b} \|q\|$. This is the maximum distance from the origin, which is also the base sensing pattern's center of rotation, to its boundary.

The sensing pattern of agent i as a function of its position q_i and orientation θ_i, can be computed by rotating around the origin and translating its base sensing pattern as follows

$$S_i\left(q_i, \theta_i\right) = q_i + \mathbf{R}(\theta_i)\ S_i^b. \tag{20}$$

By allowing a different base sensing pattern for each agent, teams of heterogeneous agents can be effectively utilized.

Since the goal of the mobile agent team is the maximization of the covered area using their sensors, while also taking into account the space density function, we define the coverage objective as in (3). The control objective is the design of a distributed control law for the mobile agents in order to guarantee monotonic increase of the coverage objective $\mathscr{H}$ over time.

2.4.2. Space partitioning

The first step in designing a distributed control law is finding a method to distribute the computation of the coverage objective $\mathscr{H}$. Due to the agents' anisotropic sensing patterns, the partitioning scheme employed in this case is highly different from Voronoi-like partitioning schemes. Instead of partitioning the whole region Ω based on the agents' positions, only the sensed region $\bigcup_{i \in I_n} S_i$ is partitioned based on the agents' sensing patterns. Each agent is assigned the part of Ω that only itself is able to sense, with parts being sensed by multiple or no agents being left unassigned.

Given a planar region Ω and a set of sensing patterns $S = \{S_1, \ldots, S_n\}$, each agent i is assigned a cell W_i as follows

$$W_i(\Omega, S) = (\Omega \cap S_i) \setminus \bigcup_{j \in I_n \setminus i} S_j, \quad i \in_I n.$$

The part of Ω sensed by multiple agents is left unassigned but still contributes toward the coverage objective $\mathscr{H}$. This part is called the common region and is computed as follows

$$W_c(\Omega, S) = \Omega \cap \bigcup_{i \in I_n} (S_i \setminus W_i). \tag{21}$$

Having defined the cells and the common region, it holds that $\bigcup_{i \in I_n} S_i = \bigcup_{i \in I_n} W_i \cup W_c \subseteq \Omega$.

We can define the neighbors of agent i as those agents that affect the computation of its cell. Since the computation of the cells relies entirely on the agents' sensing patterns, the neighbors can be defined as

$$N_i = \left\{j \in I_n \setminus i : S_i \cap S_j \neq \emptyset\right\}. \tag{22}$$

Moreover, if the maximum base sensing radius $R^{\max} = \max_{i \in I_n} R_i$ is known by all agents, and if each agent is able to communicate with all others within a radius

$$C_i = R_i + R^{\max}, \tag{23}$$

then it is guaranteed it will be able to communicate with all of its neighbors N_i. By using the neighbor definition, the proposed partitioning scheme can be computed in a distributed manner as follows

$$W_i(\Omega, S) = (\Omega \cap S_i) \setminus \bigcup_{j \in N_i \setminus i} S_j, \quad i \in I_n. \tag{24}$$

Remark 2.2. *The partitioning scheme* (24) *may result in the cell of some agent being empty or consisting of multiple disjoint regions. It should be noted however that such cases are handled successfully by the control law presented in* Section 2.4.3.

Thus by utilizing the aforementioned partitioning scheme, the coverage objective $\mathscr{H}$ can be computed as follows

$$\mathscr{H} = \sum_{i \in I_n} \int_{W_i} \phi(q) \ dq + \int_{W_c} \phi(q) \ dq. \tag{25}$$

Since $\mathscr{H}$ can be written as a sum of integrals over the assigned cells and since an agent can compute its own cell using information just from its neighbors, the computation of $\mathscr{H}$ is distributed.

2.4.3. Control law formulation

Having found a partitioning scheme that allows distributed computation of the coverage objective $\mathscr{H}$, what is left is the derivation of a distributed control law for its maximization.

Theorem 2.3. *For a team of mobile ground agents with kinematics* (18, 19)*, sensing performance* (20) *and using the partitioning* (24)*, the control scheme*

$$u_i = \alpha_{i,u} \int_{\partial W_i \cap \partial S_i} n_i \, \phi(q) \ \mathrm{d}q, \tag{26}$$

$$\omega_i = \alpha_{i,\omega} \int_{\partial W_i \cap \partial S_i} n_i \ \boldsymbol{R}\left(\frac{\pi}{2}\right)(q - q_i) \ \phi(q) \ \mathrm{d}q, \tag{27}$$

where $\alpha_{i,u}, \alpha_{i,\omega}$ are positive constants and n_i is the outward unit normal vector on ∂W_i, leading the agent to trajectories that result in monotonic increase of the coverage objective (25).

Proof. We start by evaluating the time derivative of the objective using the chain rule and the agent dynamics (18, 19)

$$\frac{\partial \mathcal{H}}{\partial t} = \sum_{i \in I_n} \frac{\partial \mathcal{H}}{\partial q_i} \frac{\partial q_i}{\partial t} + \frac{\partial \mathcal{H}}{\partial \theta_i} \frac{\partial \theta_i}{\partial t} = \sum_{i \in I_n} \frac{\partial \mathcal{H}}{\partial q_i} \dot{q}_i + \frac{\partial \mathcal{H}}{\partial \theta_i} \dot{\theta}_i = \sum_{i \in I_n} \frac{\partial \mathcal{H}}{\partial q_i} u_i + \frac{\partial \mathcal{H}}{\partial \theta_i} \omega_i.$$

By selecting the control law

$$u_i = \alpha_{i,u} \frac{\partial \mathcal{H}}{\partial q_i}, \qquad \omega_i = \alpha_{i,\omega} \frac{\partial \mathcal{H}}{\partial \theta_i}, \qquad \alpha_{i,u}, \alpha_{i,\omega} > 0,$$

we can guarantee monotonic increase of the coverage objective.

The partial derivative $\frac{\partial \mathcal{H}}{\partial q_i}$ is evaluated as follows

$$\frac{\partial \mathcal{H}}{\partial q_i} = \frac{\partial}{\partial q_i} \int_{W_i} \phi(q) \ \mathrm{d}q \ + \sum_{j \neq i} \frac{\partial}{\partial q_i} \int_{W_j} \phi(q) \ \mathrm{d}q \ + \ \frac{\partial}{\partial q_i} \int_{W_c} \phi(q) \ \mathrm{d}q.$$

Due to the partitioning scheme (24) only the common region W_c is affected by the movement of agent i and since $\frac{\partial \phi(q)}{\partial q_i} = 0$, by using the Leibniz integral rule [32], the previous equation becomes

$$\frac{\partial \mathcal{H}}{\partial q_i} = \int_{\partial W_i} v_i^i \ n_i \ \phi(q) \ \mathrm{d}q \ + \int_{\partial W_c} v_c^i \ n_c \phi(q) \ \mathrm{d}q$$

where v_j^i is the Jacobian matrix

$$v_j^i = \frac{\partial q}{\partial q_i}, \ \ q \in \partial W_j$$

and n_i is the outward unit normal vector on ∂W_i. The boundary ∂W_i can be decomposed into three disjoint sets as follows

$$\partial W_i = (\partial W_i \cap \partial S_i) \cup (\partial W_i \cap \partial \Omega) \cup (\partial W_i \cap \partial W_c), \tag{28}$$

where $\partial W_i \cap \partial S_i$ denotes part of the cell's boundary that is also part of the boundary of the agent's sensing disk, $\partial W_i \cap \partial \Omega$ denotes the common boundary between the cell and the region, while $\partial W_i \cap \partial W_c$ denotes the common boundary of the cell and the common region. This decomposition is presented in **Figure 5** where $\partial W_i \cap \partial S_i$, $\partial W_i \cap \partial \Omega$ and $\partial W_i \cap \partial W_c$ are shown in solid green, red and blue lines respectively.

Since the region Ω is assumed to be static, it holds that $v_i^i = 0, \forall q \in \partial W_i \cap \partial \Omega$. In addition, from Eq. (20) we get that $v_i^i = \mathbb{I}_2, \forall q \in \partial W_i \cap \partial S_i$. Finally, on all the common boundaries $\partial W_j \cap \partial W_c, j \in I_n$ it holds that $v_i^j = v_i^c$ and $n_j = -n_c$, as shown in **Figure 5**, leaving only an integral over $\partial W_i \cap \partial S_i$. By multiplying with $\alpha_{i,u}$ we get (26). The same procedure is used for the derivation of the rotational part of the control law (27). □

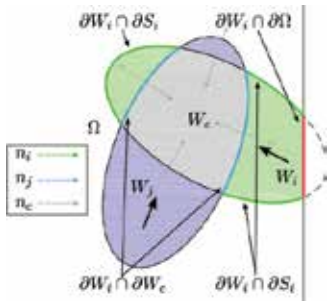


Figure 5. Decomposition of ∂W_i into disjoint sets and corresponding normal vectors.

2.4.4. Simulation study III

An indicative simulation is presented in this section. The region Ω, the space density function $\phi(q)$ and the agent initial positions are the same as in the simulation presented in Section 2.2.4. In this simulation however the agents are equipped with heterogeneous sensors with elliptical sensing patterns.

The initial and final agent configurations are shown in **Figure 6a** and **c** respectively where the agent positions are marked by black dots, the agent orientations are marked by black arrows, the boundaries of their sensing disks are shown as dashed black lines, the boundaries of their cells are marked by solid black lines while their interiors are filled in color. The agent trajectories are shown in **Figure 6b** with the initial positions marked by dots and the final positions by circles. It is observed that the agents successfully deploy inside the region, increasing the covered area in the process. In order to provide a more objective measure of the agents' performance, the two metrics defined in Eq. (8) are used. **Figure 6d** shows $\mathscr{H}^a$ in solid blue and $\mathscr{H}^r$ in dashed red with their final values being 91.3 and 93.5% respectively. This indicates that the final configuration results in both high coverage of Ω and efficient use of the agents sensors.

2.4.5. Simulation study IV

This simulation study serves to highlight the need for taking into account the agents' anisotropic sensing patterns instead of approximating them with circular ones. To that end,

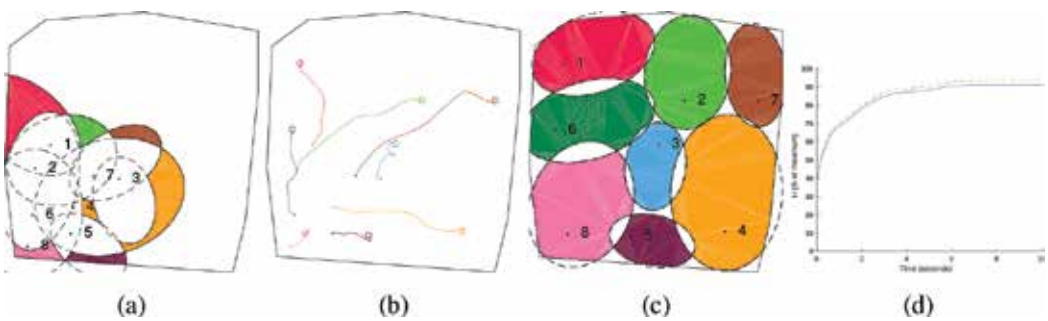


Figure 6. Simulation study III: (a) initial configuration, (b) agent trajectories, (c) final configuration and (d) evolution of the coverage objective over time.

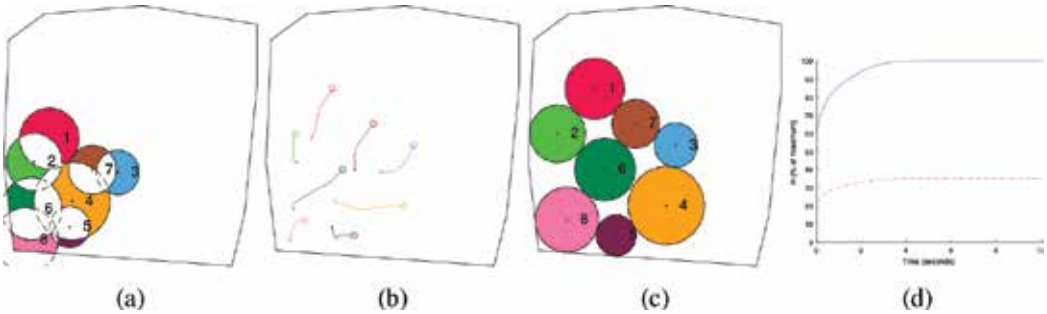


Figure 7. Simulation study IV: (a) initial configuration, (b) agent trajectories, (c) final configuration and (d) evolution of the coverage objective over time.

Simulation Study III was repeated by approximating the agents' elliptical sensing patterns with their maximal inscribed circles. The initial agent configuration, agent trajectories and final agent configuration are shown in **Figure 7a**, **b** and **c** respectively. It is observed that the agent's performance is decreased significantly when using this underapproximation of their sensing patterns. In order to provide a more objective measure of the agents' performance, the two metrics defined in Eq. (8) are used. **Figure 7d** shows $\mathscr{H}^a$ in solid blue and $\mathscr{H}^r$ in dashed red with their final values being 100% and 35.2% respectively, indicating a 62.4% decrease in the coverage of Ω compared to Simulation Study III.

2.5. Experimental implementation

An experimental implementation of a simplified version of one of the previously examined control schemes is briefly presented in this section. This experimental study first appeared and is presented in greater detail in [42]. The experiment consisted of three differential-drive robots, a visual pose tracking system using fiducial markers and a computer communicating with the robots and pose tracking system via a WiFi router. The methodology presented in Section 2.3 was used in order to take into account the positioning uncertainty of the pose tracking system. The experimental results are compared with a simulation using the same initial conditions.

2.5.1. Experimental setup

The robots used in the experiment were the differential-drive *AmigoBot*s by *Omron Adept MobileRobots*. The robots are 33 cm $\times$ 28 cm $\times$ 15 cm in size, weigh 3.6 *kg* and are able to carry a payload of up to 1 kg. Their maximum linear and rotational velocities are $v^{\max} = 1$ m/s and $\omega^{\max} = 100^{\circ}$/s. Although these robots are equipped with encoders measuring $39,000$ ticks/revolution which can be used for estimating their pose, an external pose tracking system was used instead due to the encoders' drifting error. Since the *AmigoBot*s lack any omnidirectional sensors, for the sake of the control law it was assumed that they were equipped with sensors with a common sensing radius of $R = 0.3$ m.

The external pose tracking system consists of a USB camera and an ODROID-XU4 computer. Pose tracking is achieved by attaching a fiducial marker on top of each robot and using the

ArUco [43] library to estimate the pose of these markers. As is the case with all positioning systems, ArUco has a certain degree of uncertainty in its pose estimations. In order to get an estimate of this uncertainty, a fiducial marked was placed on the vertices and the centroid of the region Ω resulting in a maximum error of 0.032 m, which was used as the measure of positioning uncertainty r for all robots.

The control scheme was implemented as a loop in the main computer with an iteration period of $T_s = 0.1$ seconds. At each iteration, a simplified version of the control law (15) is computed for each agent, and from that, a target point q_i^t is derived for each agent. Then a feedback controller is used in order to lead each robot to each respective target point. Once all robots are within a predefined distance $d^t = 0.02$ m of their respective target points, new target points are computed from the robots' current positions. The feedback control law used for each robot was

$$v_i = \min\left(\frac{\|q_i^t - q_i\|}{T_s}, v^{\max}\right)\cos(d\theta_i), \quad \omega_i = \min\left(\frac{|d\theta_i|}{T_s}, \omega^{\max}\right)\sin(d\theta_i),$$

where q_i and θ_i are the robot's current position and orientation, v_i and ω_i the robots linear and rotational velocity control inputs respectively and $d\theta_i = \measuredangle(q_i^t - q_i) - \theta_i$.

2.5.2. Experimental results

The robots' initial configuration, which is common between the experiment and simulation is shown in **Figure 8a**. The final configurations of the experiment and the simulation are shown in **Figure 8c** and **d**, respectively. The boundaries of the agents' positioning uncertainty regions are shown as black circles, the boundaries of their sensing disks are shown in dashed black line and the boundaries of their cells are marked by solid black lines while their interiors are filled in color. Some photographs of the robots' initial and final configurations are presented in **Figure 9a** and **b** respectively where the ArUco fiducial markers can be seen. In both the experiment and the simulation it is observed from the robots' final configurations that their guaranteed sensed regions are completely contained within their respective AWGV cells, i.e. $S_i^g \subset G_i, \forall i \in I_n$, which is a globally optimal configuration. The robots' trajectories are depicted in **Figure 8b** in blue for the experiment and in red for the simulation, with the initial and final positions marked by dots and circles respectively. The simulation trajectories are smooth but

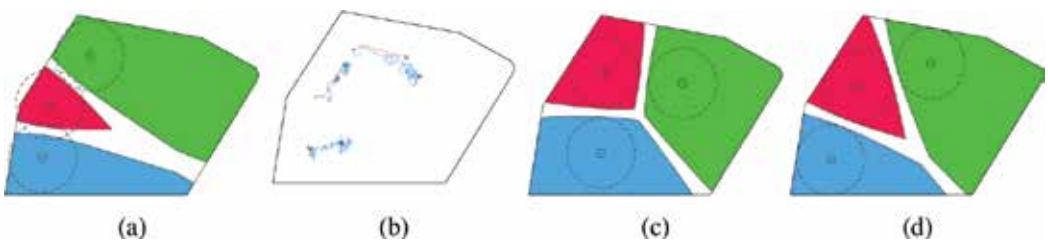


Figure 8. Experiment: (a) initial configuration, (b) experimental (blue) and simulated (red) robot trajectories, (c) experiment final configuration and (d) simulation final configuration.

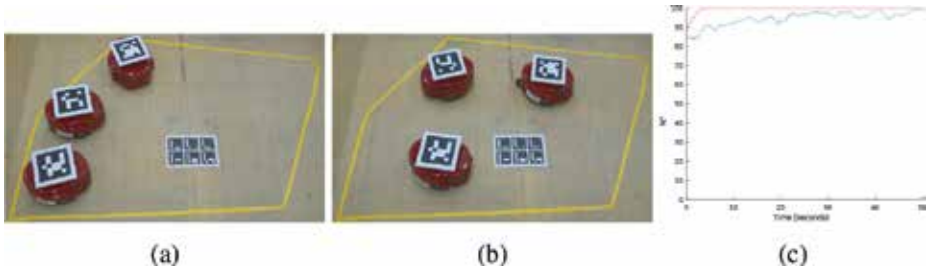


Figure 9. Experiment: (a) initial configuration, (b) final configuration and (c) evolution of the coverage objective over time for the experiment and simulation.

the experimental trajectories have many turns due to the robots moving to target points. The robots' final positions have an error of 9.27% the diameter of Ω between the experiment and the simulation. This large error is attributed to the difference between the implemented control laws as well as the existence of multiple global optima for this particular coverage setup. Figure shows the evolution of the metric $\mathcal{H}^a$ over time for the experiment in blue and the simulation in red where it is seen that it increased from 83.70 to 98.95% in the experiment. Although in the case of the experiment its increase was not monotonic, this is to be expected as the implemented control law differed from the theoretical one. The lower convergence speed is also attributed to this difference as well as the constraints on the robots' translational and rotational velocities.

3. Conclusions and future work

This chapter presented an overview of past and current work on area coverage problems. A strong theoretical background has been provided, along with indicative simulations results and an experimental implementation of one of the presented control schemes. The problem of multiagent area coverage still offers possibilities for original research. One possible extension would be the usage of more realistic sensor models, such as visual sensors. The usage of visual sensors can result in the incorporation of coverage quality metrics in the objective or in variable sensing patterns in the case of pan-tilt-zoom cameras. Another aspect of multirobot area coverage problem that has not been studied thoroughly yet is the development of communication systems and algorithms that allow the agents to exchange information in a distributed manner. Finally, implementations in actual robotic systems in order to solve practical problems are not yet common.

Conflict of interest

The authors declare no conflict of interest.

Author details

Sotiris Papatheodorou and Anthony Tzes*

*Address all correspondence to: anthony.tzes@nyu.edu

New York University Abu Dhabi, Engineering Division, Abu Dhabi, United Arab Emirates

References

[1] Pierson A, Figueiredo LC, Pimenta LCA, Schwager M. Adapting to sensing and actuation variations in multi-robot coverage. The International Journal of Robotics Research. 2017; **36**(3):337-354

[2] Abbasi F, Mesbahi A, Velni JM. A team-based approach for coverage control of moving sensor networks. Automatica. 2017;**81**:342-349

[3] Mahboubi H, Habibi J, Aghdam AG, Sayrafian-Pour K. Distributed deployment strategies for improved coverage in a network of mobile sensors with prioritized sensing field. IEEE Transactions on Industrial Informatics. 2013;**9**(1):451-461

[4] Palacios-Gasós JM, Montijano E, Sagüés C, Llorente S. Distributed coverage estimation and control for multirobot persistent tasks. IEEE Transactions on Robotics. 2016;**PP**(99): 1-17. ISSN: 1552-3098

[5] Zheng X, Koenig S, Kempe D, Jain S. Multirobot forest coverage for weighted and unweighted terrain. IEEE Transactions on Robotics. 2010;**26**(6):1018-1031

[6] Luo C, Yang SX, Li X, Meng MQ-H. Neural-dynamics-driven complete area coverage navigation through cooperation of multiple mobile robots. IEEE Transactions on Industrial Electronics. 2017;**64**(1):750-760

[7] Arslan O, Koditschek DE. Voronoi-based coverage control of heterogeneous disk-shaped robots. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). Stockholm, Sweden: IEEE; 2016. pp. 4259-4266

[8] Razak RA, Srikant S, Chung H. Decentralized and adaptive control of multiple nonholonomic robots for sensing coverage. International Journal of Robust and Nonlinear Control. 2018;**28**(6):2636-2650

[9] Dirafzoon A, Menhaj MB, Afshar A. Decentralized coverage control for multi-agent systems with nonlinear dynamics. IEICE Transactions on Information and Systems. 2011;**94**(1):3-10

[10] Mahboubi H, Aghdam AG. Self-deployment algorithms for coverage improvement in a network of nonidentical mobile sensors with limited communication ranges. In: Proceedings American Control Conference (ACC). Washington, DC, USA: American Automatic Control Council (AACC); June 2013. pp. 6882-6887

[11] Kantaros Y, Zavlanos M. Distributed communication-aware coverage control by mobile sensor networks. Automatica. 2016;**63**:209-220

[12] Bullo F, Carli R, Frasca P. Gossip coverage control for robotic networks: Dynamical systems on the space of partitions. SIAM Journal on Control and Optimization. 2012;**50**(1):419-447

[13] Breitenmoser A, Schwager M, Metzger JC, Siegwart R, Rus D. Voronoi coverage of non-convex environments with a group of networked robots. In: Proceedings IEEE International Conference on Robotics and Automation (ICRA). Anchorage, Alaska, USA: IEEE; May 2010. pp. 4982-4989

[14] Stergiopoulos Y, Thanou M, Tzes A. Distributed collaborative coverage-control schemes for non-convex domains. IEEE Transactions on Automatic Control. 2015;**60**(9):2422-2427

[15] Alitappeh RJ, Jeddisaravi K, Guimarães FG. Multiobjective multi-robot deployment in a dynamic environment. Soft Computing. 2016;**21**(21):1-17

[16] Franco C, Stipanović DM, López-Nicolás G, Sagüés C, Llorente S. Persistent coverage control for a team of agents with collision avoidance. European Journal of Control. 2015; **22**:30-45

[17] Breitenmoser A, Martinoli A. On combining multi-robot coverage and reciprocal collision avoidance. In: Distributed Autonomous Robotic Systems. Tokyo: Springer; 2016. pp. 49-64

[18] Cortés J, Bullo F. Coordination and geometric optimization via distributed dynamical systems. SIAM Journal on Control and Optimization. 2005;**44**(5):1543-1574

[19] Nguyen MT, Rodrigues L, Maniu CS, Olaru S. Discretized optimal control approach for dynamic multi-agent decentralized coverage. In: Proceedings IEEE International Symposium on Intelligent Control (ISIC), IEEE. Zadar, Croatia; September 2016. pp. 1-6

[20] Nowzari C, Cortés J. Self-triggered coordination of robotic networks for optimal deployment. Automatica. 2012;**48**(6):1077-1087

[21] Mavrommati A, Tzorakoleftherakis E, Abraham I, Murphey TD. Real-time area coverage and target localization using receding-horizon ergodic exploration. IEEE Transactions on Robotics. 2018;**34**(1):62-80

[22] Bentz W, Panagou D. 3D dynamic coverage and avoidance control in power-constrained UAV surveillance networks. In: Unmanned Aircraft Systems (ICUAS), 2017 International Conference on. Miami, FL, USA: IEEE; 2017. pp. 1-10

[23] Tzes M, Papatheodorou S, Tzes A. Visual area coverage by heterogeneous aerial agents under imprecise localization. IEEE Control Systems Letters. Oct 2018;**2**(4):623-628. ISSN: 2475-1456

[24] Schwager M, Julian BJ, Angermann M, Rus D. Eyes in the sky: Decentralized control for the deployment of robotic camera networks. Proceedings of the IEEE. 2011;**99**(9):1541-1561

[25] Papatheodorou S, Tzes A, Stergiopoulos Y. Collaborative visual area coverage. Robotics and Autonomous Systems. June 2017;**92**:126-138. ISSN: 0921-8890

[26] Di Franco C, Buttazzo G. Coverage path planning for UAVs photogrammetry with energy and resolution constraints. Journal of Intelligent & Robotic Systems. 2016;**83**(3-4):1-18

[27] Cortés J, Martinez S, Bullo F. Spatially-distributed coverage optimization and control with limited-range interactions. ESAIM: Control, Optimisation and Calculus of Variations. 2005;**11**(4):691-719

[28] Kwok A, Martinez S. A distributed deterministic annealing algorithm for limited-range sensor coverage. IEEE Transactions on Control Systems Technology. 2011;**19**(4):792-804

[29] Kantaros Y, Thanou M, Tzes A. Distributed coverage control for concave areas by a heterogeneous robot-swarm with visibility sensing constraints. Automatica. 2015;**53**:195-207

[30] Bartolini N, Calamoneri T, La Porta T, Silvestri S. Autonomous deployment of heterogeneous mobile sensors. IEEE Transactions on Mobile Computing. 2011;**10**(6):753-766

[31] Mahboubi H, Aghdam AG. Distributed deployment algorithms for coverage improvement in a network of wireless mobile sensors: Relocation by virtual force. IEEE Transactions on Control of Network Systems. 2016;**61**(99):1. ISSN: 2325-5870

[32] Flanders H. Differentiation under the integral sign. American Mathematical Monthly. 1973;**80**(6):615-627

[33] Habibi J, Mahboubi H, Aghdam AG. Distributed coverage control of mobile sensor networks subject to measurement error. IEEE Transactions on Automatic Control. November 2016;**61**(11):3330-3343. ISSN: 0018-9286

[34] Davis B, Karamouzas I, Guy SJ. C-OPT: Coverage-aware trajectory optimization under uncertainty. IEEE Robotics and Automation Letters. July 2016;**1**(2):1020-1027. ISSN: 2377-3766

[35] Papatheodorou S, Stergiopoulos Y, Tzes A. Distributed area coverage control with imprecise robot localization. In: 24th Mediterranean Conference on Control and Automation (MED), Mediterranean Control Association (MCA). Athens, Greece; June 2016. pp. 214-219

[36] Evans W, Sember J. Guaranteed Voronoi diagrams of uncertain sites. In: 20th Canadian Conference on Computational Geometry. Montreal, Canada; 2008. pp. 207-210

[37] Gusrialdi A, Hirche S, Asikin D, Hatanaka T, Fujita M. Voronoi-based coverage control with anisotropic sensors and experimental case study. Intelligent Service Robotics. 2009; **2**(4):195

[38] Zhang X, Chen X, Liang X, Fang Y. Distributed coverage optimization for deployment of directional sensor networks. In: Decision and Control (CDC), 2015 IEEE 54th Annual Conference on. Osaka, Japan: IEEE; 2015. pp. 246-251

[39] Panagou D, Stipanovic DM, Voulgaris PG. Distributed dynamic coverage and avoidance control under anisotropic sensing. IEEE Transactions on Control of Network Systems. 2016;**PP**(99):1. ISSN: 2325-5870

[40] Laventall K, Cortés J. Coverage control by multi-robot networks with limited-range anisotropic sensory. International Journal of Control. 2009;**82**(6):1113-1121

[41] Stergiopoulos Y, Tzes A. Cooperative positioning/orientation control of mobile heterogeneous anisotropic sensor networks for area coverage. In: Proceedings IEEE International Conference on Robotics and Automation (ICRA), IEEE. Hong Kong, China; 2014. pp. 1106-1111

[42] Papatheodorou S, Tzes A, Giannousakis K. Experimental studies on distributed control for area coverage using mobile robots. In: 25th Mediterranean Conference on Control and Automation (MED), Mediterranean Control Association (MCA). Valletta, Malta; July 2017. pp. 690-695

[43] Garrido-Jurado S, Munoz Salinas R, Madrid-Cuevas FJ, Marín-Jiménez MJ. Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition. 2014;**47**(6):2280-2292. ISSN: 0031-3203

Chapter 4

Mobile Robot Feature-Based SLAM Behavior Learning, and Navigation in Complex Spaces

Ebrahim A. Mattar

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.81195

Abstract

Learning mobile robot space and navigation behavior, are essential requirements for improved navigation, in addition to gain much understanding about the navigation maps. This chapter presents mobile robots feature-based SLAM behavior learning, and navigation in complex spaces. Mobile intelligence has been based on blending a number of functionaries related to navigation, including learning SLAM map main features. To achieve this, the mobile system was built on diverse levels of intelligence, this includes principle component analysis (PCA), neuro-fuzzy (NF) learning system as a classifier, and fuzzy rule based decision system (FRD).

Keywords: SLAM, PAC, NF classification, fuzzy rule based decision, navigation

1. Introduction

1.1. Study background

Interactive mobile robotics systems have been introduced by researcher's worldwide. The main focus of such research directions, are how to let a mobile robotic system to navigate in an unstructured environment, while learning its features. To meet these objectives, mobile robots platforms are to be equipped with AI tools. In particular to achieve this, Janglová in [1], describes an approach for solving the motion-planning problem in mobile robot control using neural networks-based technique. The proposed system consists of a head artificial neural network, which was used to determine the free space using ultrasound range finder data. In terms of maps building with visual mobile robot capabilities, a remote controlled vision guided mobile robot system was introduced by Raymond et al. [2]. The drive of the

work, was to describe exploratory research on designing remote controlled emergency stop and vision systems for an autonomous mobile robot. Camera modeling and distortion calibration for mobile robot vision was also introduced by Gang et al. [3]. In their paper they presented an essential camera calibration technique for mobile robot, which is based on PIONEER II experiment platform. Bonin-Font et al. [4], presented a map-based navigation and mapless navigation, as they subdivided in metric map-based navigation and topological map based navigation. Abdul et al. [5] have introduced a hybrid approach for vision based self-localization of autonomous mobile robots. They presented a hybrid approach towards self-localization of tiny autonomous mobile robots in a known but highly dynamic environment. Kalman filter was used for tracking of the globally estimated position. In [8], Filliata and Meyer, presented a 3-level hierarchy of localization strategies, and a direct position inference, single-hypothesis tracking, and multiple-hypothesis tracking.

They stated the advantages and drawbacks of these strategies. In [6], Andreja et al. have presented a fuzzy ART neural architecture for robot map learning and navigation. Araujo proposed methods that are integrated into a navigation architecture. Further, intelligence based navigation was further discussed by [9–11]. In [12], Vlassis et al., motioned, "method for building robot maps by using a Kohonen's self-organizing artificial neural network, and describe how path planning can be subsequently performed on such a map". The built ANN related SOM is shown in **Figure 1**. Stereo vision-based autonomous mobile robot was also given by Changhan et al. [13]. In their research, they proposed a technique to give more autonomy to a mobile robot by providing vision sensors. In [14], Thrun reported an approach that integrates two paradigms: grid-based and topological. The intelligent control of the mobile robot, was based on image processing was also given by Nima et al. [15]. In terms of leaning intelligent navigation, intelligent robot control using an adaptive critic with a task control center and dynamic database was also introduced by Hall et al. [16]. This involves development and simulation of a real time controller for an intelligent, vision guided robot. Such models are also necessary for sizing the actuators, tuning the controller, and achieving superior performance. A novel feature of the proposed approach is that the method is applicable to both robot arm manipulators and robot bases such as wheeled mobile robots. Stereo vision based self-localization of autonomous mobile robots was furthermore introduced by Abdul et al. [17]. In reference to the work presented, a vision based self-localization of tiny autonomous mobile robots in a known but highly dynamic environment. A learning mobile robots was shown by Hall et al. [18]. They presented a discussion of recent technical advances in learning for intelligent mobile robots. Novel application of a laser range finder with vision system for wheeled mobile robot was presented by Chun et al. [19], where their research presents a trajectory planning strategy of a wheeled mobile robot in an obstructed environment. A vision-based intelligent path following control of a four-wheel differentially driven skid steer mobile robot was given by Nazari and Naraghi [20]. In this work, a Fuzzy Logic Controller (FLC) for path following of a four-wheel differentially skid steer mobile robot is presented. Color learning and illumination invariance on mobile robots survey was given by Mohan et al. [21]. A major challenge to the widespread deployment of mobile robots is the ability to function autonomously. Two arms and two legs like an ape, was aimed to study a variety of vision-based behaviors. In addition, robot based on the remote-brained approach

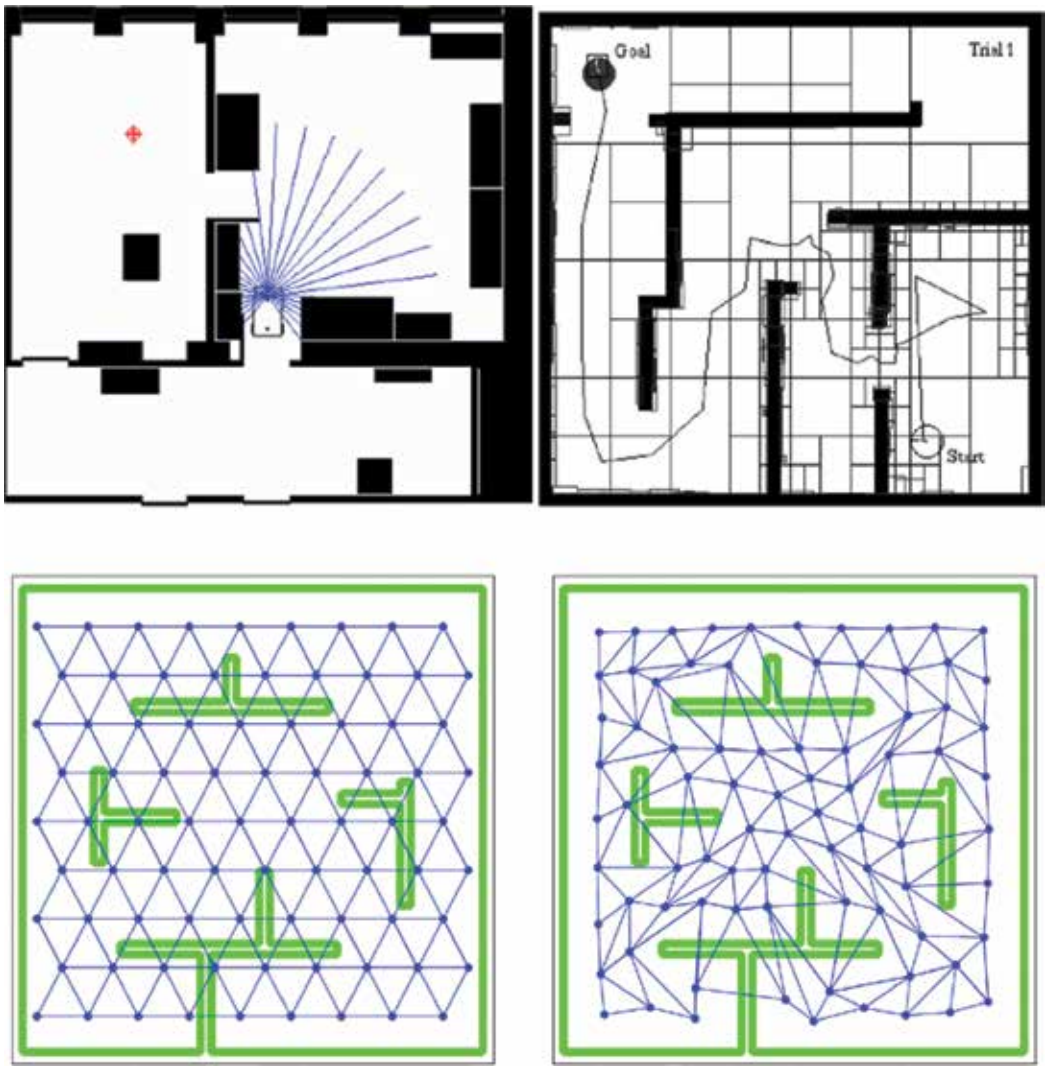


Figure 1. ANN-self organizing maps, for learning mobile robot navigation [1, 6–8].

was given by Masayuki et al. [22]. Localization algorithm of mobile robot based on single vision and laser radar have been presented by Xiaoning [23]. In order to increase the localization precision of mobile robot, a self-localization algorithm based on odometry, single vision and laser radar is proposed. The data provided by odometry, single vision, and laser radar were fused together by means of an Extended Kalman filter (EKF) technique. Mobile robot self-localization in complex indoor environments using monocular vision and 3D model, was moreover presented by Andreja et al. [24], they considered the problem of mobile robot pose estimation using only visual information from a single camera and odometry readings. Human observation based mobile robot navigation in intelligent space was also given by

Takeshi and Hashimoto [25], they investigated a mobile robot navigation system which can localize the mobile robot correctly and navigate based on observation of human walking. Similar work was also given by Manoj and Ernest [26].

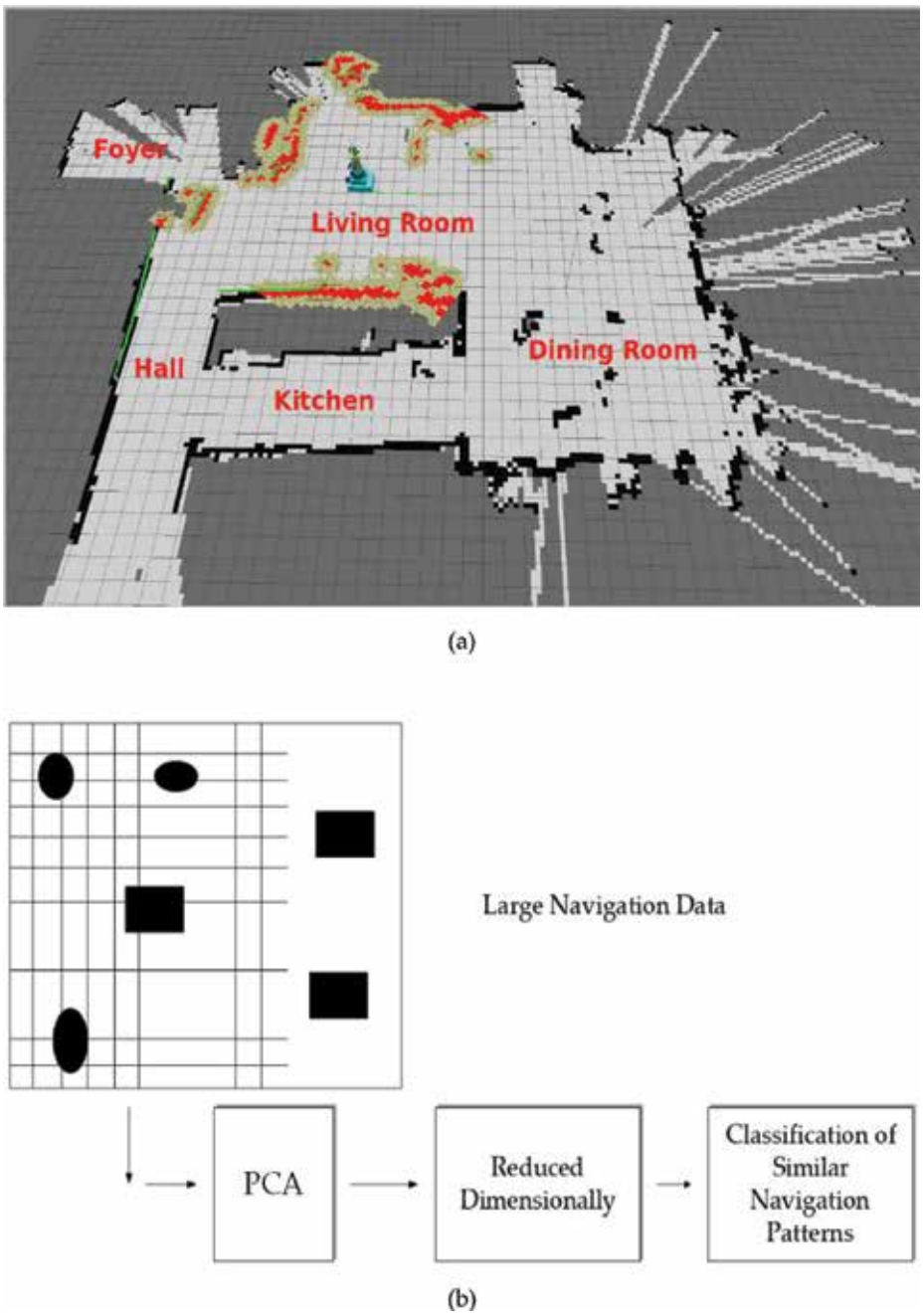


Figure 2. (a) A sample of SLAM details for learning, picture source robot cartography [27]. (b) Mobile robot training patterns generation. A space is represented by maps space's basis.

1.2. Research objectives

Given the previous background, this current presented work is focusing on learning navigation maps with intelligent capabilities. The system is based on PCA representation of large navigation maps, Neuro-fuzzy classifier, and a fuzzy decision based system. For bulky amount of visual and non-visual mobile data measurements (odometry, and the observations), the approach followed, is to reduce the mobile robot observation dimensionality using principle component analysis (PCA), thus to generate a reduced representation of the navigation map (SLAM), refer to **Figure 2** for details. A learning system was used to learn navigation maps details, hence to classify the representations, (in terms of observation features). The learned system was employed for navigating maps, and other mobile robot routing applications.

2. Building navigation maps

2.1. Simultaneous localization and mapping (SLAM)

SLAM, is a routine that estimates a pose of a mobile robot, while mapping the environment at the same time. SLAM is computationally intensive, since maps represent localization, hence accurate pose estimate is needed for mapping. For creating navigation intelligence capabilities during path planning, this is achieved by learning spaces, once robot was in motion. This is based on learning path and navigation behavior. There are four important stages for building SLAM, this localization, map building and updates, searching for optimal path and planning. Optimal path search is done by A^*, occupancy grids mapping. Given a mobile robot control inputs U as set of controls, $U_{1:k} = (u_1, u_2, \ldots, u_k)$, with mobile parameters measurements (mobile observations) as $Z_{1:k} = (z_1, z_2, \ldots, z_k)$. For odometry, refer to **Figure 3**.

$$x_k = \begin{pmatrix} x_r \\ m_1 \\ m_2 \\ . \\ . \\ m_n \end{pmatrix} \text{ and } C_k = \left(\begin{pmatrix} C_r & \cdots & C_{RM_n} \\ \vdots & \ddots & \vdots \\ C_{M_nR} & \cdots & C_{M_n} \end{pmatrix} \right) \tag{1}$$

In reference to **Figure 3**, the mobile robot starts to move in the space, with a target to research into a predefined final position. While the robot in movement, a SLAM is built, and measurements are recorded from the mobile observations, as $Z_{1:k} = (z_1, z_2, \ldots, z_k)$. All recorded observations are considered as inputs to the PAC, hence they are tabulated into predefined format, for later processing using the PCA algorithm.

2.2. Monte-Carlo (MC) localization

Monte-Carlo localization, is a well-known technique in literature, and still being used for localization parameters estimation. In sampling-based methods, one represents the density by

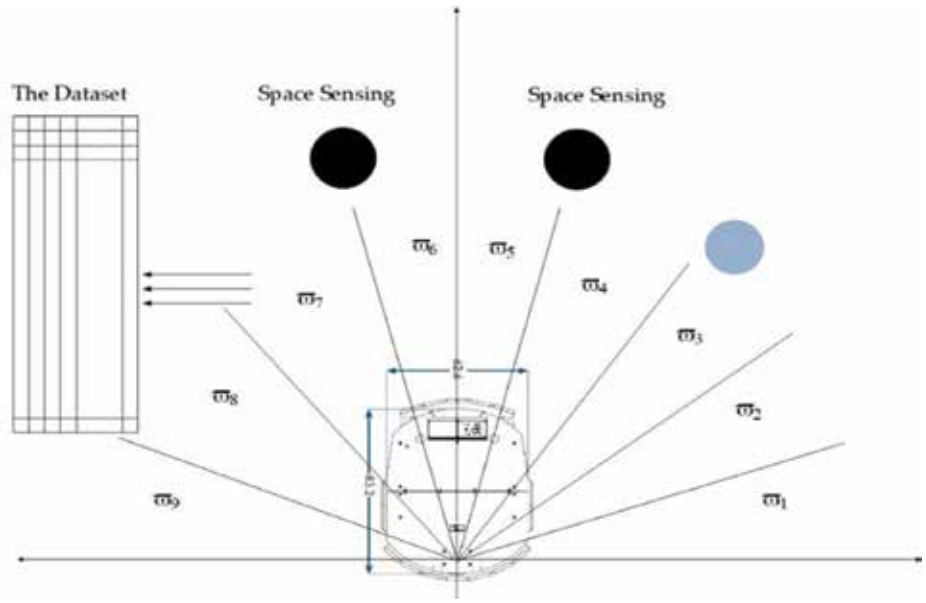


Figure 3. Odometry, generation of navigation patterns for spaces, the (maps).

a set of η random samples of particles. The goal is then to recursively compute at each time step k set of samples of Z_k that is drawn from density $p(x_k : Z^k)$. A particularly elegant algorithm to accomplish this has recently been suggested independently by various authors. In analogy with the formal filtering problem, the algorithm proceeds in two phases. In the first phase we start from a set of particles $S_{(k-1)}$ computed in the previous iteration, and apply the motion model to each particle $S^i_{(k-1)}$ by sampling from the density $p(x_k : S^i_{(k-1)}, u_{(k-1)})$ for each particle $S^i_{(k-1)}$: draw one sample $S^i_{(k)}$ from $(x_k : S^i_{(k-1)}, u_{(k-1)})$. We have used a motion model and set of particles $S^i_{(k-1)}$ to build an empirical predictive density function of:

$$(p)' = f(x, y, \theta, \Delta s_r, \Delta s_l), (p)' = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{pmatrix} \quad (2)$$

In Eq. (2), we describe a blended density approximation to $p\left(x_k : Z^{(k-1)}\right)$. The environment, or the mobile robot space is highly redundant, once used to describe maps.

3. Principle component analysis (PCA)

3.1. PCA based statistically and dimensionality reduction

While in navigation, each traveled path, region, zone, etc. are characterized by diverse behavior (i.e. features), **Figure 4**. If x is matrix of representation for distances and measurements at

http://dx.doi.org/10.5772/intechopen.81195

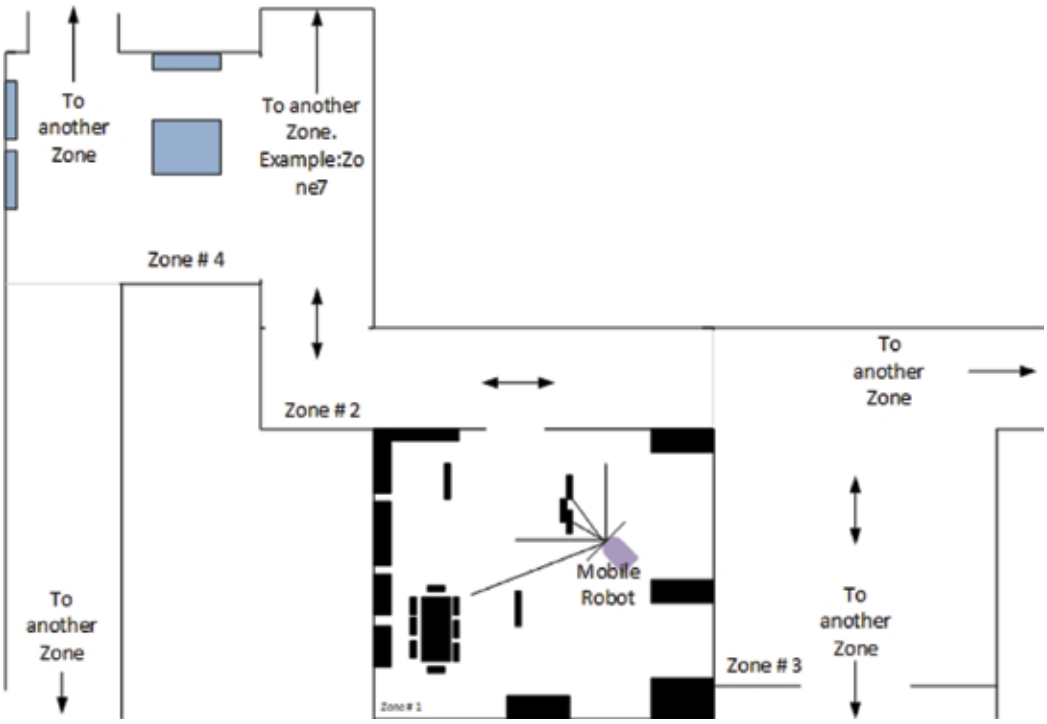


Figure 4. Robot navigation spaces. A representation of navigation segments, zones, areas, ….. (S_1, S_2, S_3, …. S_n, Z_1, Z_2, Z_3, …. Z_m, A_1, A_2, A_3, …. A_n).

each location during navigation, a covariance matrix for the set of maps is considered highly non-diagonal. Mathematically, the previous notation is:

$$\rho = XX^t\left(\begin{pmatrix} \sigma_{11}^X & \cdots & \sigma_{1,w\times h}^X \\ \vdots & \ddots & \vdots \\ \sigma_{w\times h,1}^X & \cdots & \sigma_{w\times h,w\times h}^X \end{pmatrix}\right) \tag{3}$$

σ_{ij}^X represents covariance between distances for location (w) and location (h). There is a relation between covariance coefficients and correlation coefficients. The covariance matrix ρ is expressed as:

$$\rho = \frac{1}{j}\sum_{n=1}^{j}\left(\lambda_n \times \lambda_n^T\right) \tag{4}$$

Since principal components are calculated linearly, let P be a transformation matrix:

$$Y = P^T \times X \quad and \quad X = P \times Y$$

In fact, $P = P^{-1}$, since the $\boldsymbol{P}$'s columns are orthonormal to each other, $P^T \times P = I$. Now, the question is, what is the value of P given the condition that S_y must be a diagonal matrix, i.e.

$$S_y = Y \times Y^T = P^T \times X \times X^T \times P$$
$$S_y = P^T \times S_y \times P$$

in such away S_y is a rotation of S_x by P. Choosing P as being a matrix containing eigenvectors of S_x:

$$S_x \times P = \Lambda \times P$$

where Λ is a diagonal matrix containing eigenvalues of S_x. In this regard,

$$S_y = P^T \times \Lambda \times P = \Lambda \times P^T \times P = \Lambda$$

and S_y is a diagonal matrix containing eigenvalues of S_x. Since the diagonal elements of S_y are the variance of components of training patterns in the (in navigation) space, the eigenvalues of S_x are those variances.

This is further expanded into:

$$\rho = \begin{pmatrix} \beta_{1,1-}\overline{\beta}_1 & \cdots & \beta_{1,k-}\overline{\beta}_k \\ \vdots & \ddots & \vdots \\ \beta_{j,1-}\overline{\beta}_1 & \cdots & \beta_{j,k-}\overline{x}_k \end{pmatrix} * \begin{pmatrix} \beta_{1,1-}\overline{\beta}_1 & \cdots & \beta_{j,1-}\overline{\beta}_1 \\ \vdots & \ddots & \vdots \\ \beta_{1,k-}\overline{\beta}_k & \cdots & \beta_{j,k-}\overline{\beta}_k \end{pmatrix} \tag{5}$$

Finally, covariance matrix ρ is further expressed by:

$$\rho = \begin{pmatrix} cov(\beta_1, \beta_1) & \cdots & cov(\beta_1, \beta_k) \\ \vdots & \ddots & \vdots \\ cov(\beta_k, \beta_1) & \cdots & cov(\beta_k, \beta_k) \end{pmatrix} \tag{6}$$

The p, is a symmetric (around the main diagonal), and a $\Re^{n \times n}$ matrix. The diagonal of, represents the covariance between the matrix and it self. To recognize normalized dataset for patterns, the covariance ρ of Eq. (6) plays an important role. This can be achieved by getting the eigenvectors of covariance ρ matrix of Eq. (6). Given this background, therefore we need to compute eigenvalues and eigenvectors using numerical approach. For a $\Re^{k \times k}$ matrix ρ, if we search for a row vector $\Re^{k \times 1}$ X that could be multiplied by ρ and get the same vector X multiplied by eigenvalues λ and eigenvector. Matrix ρ transforms the vector X to scale positions by an amount equal to λ, gives a transformation matrix:

$$(\rho X) = (\lambda X) \tag{7}$$

In reference to Eq. (7), and for a $\Re^{k \times k}$ matrix ρ, we shall compute for (k) eigenvalues. The (k) eigenvalue (λ), are hence used for scaling every (k) eigenvectors. Individual eigenvalues (λ), are also found by solving the below defined identity as expressed by Eq. (8):

$$[(\rho - I \times \lambda)\mathrm{X} = 0] \tag{8}$$

where I is an identity matrix. We shall compute for the determinant of Eq. (8), i.e., $|(\rho - I\lambda)| = 0$, while solving for the eigenvalues, λ. While substituting for the (λ) in Eq. (8), and solving for (X), this will result in finding the eigenvector (X), once λ are satisfying the following:

$$|(\rho - I \times \lambda)| = 0 \tag{9}$$

Following Eq. (5) to Eq. (9), and computing for eigenvalues, hence reordering the eigenvalues according to a descending order, this represents a major step in building a PAC based recognition system for the mobile robot dataset generated by the navigation system.

4. Learning system: learning mobile robot navigation maps

4.1. Feature-based SLAM learning

While stating in Section 3 steps for PAC computations, in this section we shall make a focus on building a learning system for the mobile navigation. In reference to **Figure 4**, the mobile robot will be generating navigation dataset. Dataset is generated at different locations, during the mobile robot motion. Navigation dataset involves sensory measurements, odometry, and locality information (i.e. zones, areas, segments …), refer to **Figure 4**. An important part of the dataset, is also the part generated by the SLAM, as already described in Section 2. It is not achievable to encompass all dataset, as this is massive dataset. However, we shall rely on features of the mobile dataset, i.e. the PCA based features of navigation dataset, (the SLAM features). Robot navigation spaces. A representation of navigation segments, zones, areas, ….. are designated as (S_1, S_2, S_3, …. S_n, Z_1, Z_2, Z_3, …. Z_m, A_1, A_2, A_3, …. A_n). For each of such different segments, zones, and areas of navigation, there will be features associated with it. Features during navigation will be used for further processing. This includes a five layers feature learning NF architecture (classifier), and a fuzzy decision system (**Figure 5**).

4.2. Neuro-fuzzy features classifier (NFC) architecture

For the case of fuzzy decision making system, it is essential to incorporate a priori knowledge for the mobile movements in the space. In this respect, many conventional approaches rely on depth physical knowledge describing the system. An issue with fuzzy decision is that knowledge are mathematically impervious. This results in and there is no formal mathematical representation of the system's behavior. This prevents the application of conventional empirical modeling techniques to fuzzy systems, making knowledge validation and comparison hard to perform. Creating a benchmark measure of performance by a minimum distance classifier.

Decision rule adopted by such system, is to assign X to a class whose mean feature vector is closest (Euclidean Distance) to X. A decision is given by:

$$\begin{cases} \|d - \overline{d}_1\| & \leq \|d - \overline{d}_2 \Longrightarrow d \in h_2 \\ else & d \in h_2 \end{cases} \tag{10}$$

Rule-based structure of fuzzy knowledge allows for integrating heuristic knowledge with information obtained from process measurements. The global operation of a system is divided into several local operating conditions. Within each region R_i, a representation:

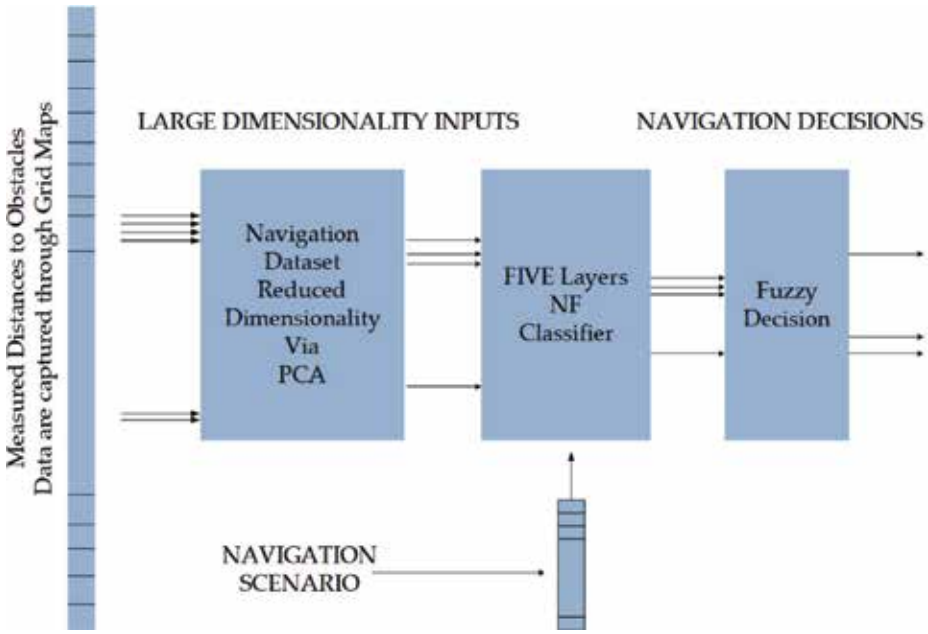


Figure 5. The recognition system. PAC for SLAM features computations, a five layers neuro-fuzzy classifier, and last a fuzzy decision based system.

$$R_i\hat{y}_i(k) = \sum_{ij}^{o} \chi_{ij} y(k) + \sum_{ij}^{h} \psi_{ij} u(k) \text{ for h} = 1, 2, \ldots \text{r} \tag{11}$$

In reference to **Figure 6**, for Eq. (11), $\hat{y}_i$ is the computed fuzzy output, u is the system input, in the i^{th} operating region, (h) is the number of fuzzy operating regions. In addition, both (i) and (o) do represent the time lags in the input and the output, respectively, μ_i is the membership function. Finally, χ_{ij} and ψ_{ij} are the few parameters. The membership function for inputs, is constructed in a number of ways.

The fuzzy knowledge system (Neuro-fuzzy) illustrated in **Figure 6**, is an exceptional architecture of network topology. This architecture combines advantages of fuzzy reasoning and the classical neural networks. In its broader sense, the architecture rule (r_i), demonstrates a relation between the input map feature space, and named classes. This is further expressed as follows:

$$Rule\ r_i \implies if\ \ \chi_{si}\ and\ \chi_{sj}\ is\ A_{ij} \ldots\ and\ \chi_{sn}\ is\ A_{in},\ \implies class\ name\ is\ C_k \tag{12}$$

In Eq. (12), the Gaussian membership function is defined as:

$$\mu_{ij}\left(\chi_{sj}\right) = exp\left(-\left(\chi_{sj} - c_{ij}\right)^2 | 2\sigma_{ij}^2\right)$$

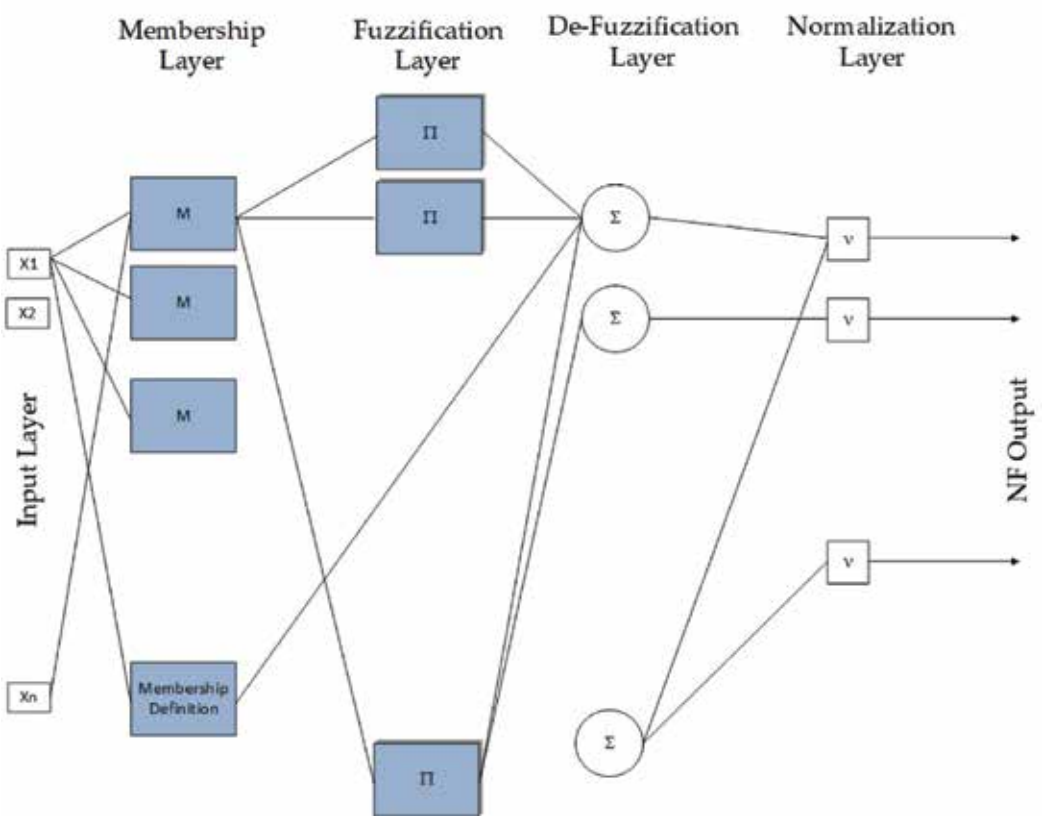


Figure 6. NF classifier architecture. The architecture is used to classify features of navigation maps.

$\mu_{ij}(\chi_{sj})$ is the membership grade of i^{th} rule and j^{th}. That is, the (*if*) parts of the rules are same as in the ordinary fuzzy (*if-then*) rules, (*then*) parts are some combinations of the input variables. For each i^{th} node in this layer is a square node with a node function.

$$\alpha_{is} = \prod_{j=1}^{n} \mu_{ij}(\nu_{sj}) \tag{13}$$

where v_{sj} is the input to i^{th} node given as the linguistic label (*small, large, .. etc.*) associated with this node function, n is the number of features. The membership is a bell-shape type, and ranged between (1 and 0).

$$O_{sk} = \frac{\beta_{sk}}{\sum_{l=1}^{k} \beta_{sl}} \tag{14}$$

As values of these parameters change, membership shaped functions vary accordingly, thus exhibiting various forms of membership functions on the linguistic label A_i. For every node in

this layer, there is a circle node which multiplies incoming signals and sends their product out. Stages of the adopted Neuro-fuzzy classifier, is shown in **Figure 6**.

$$\chi_i = \mu A_i x(k_1) \times \mu B_i y(k_2) \;\; for \;\; i = 1, 2 \tag{15}$$

The output node computes the system output as summation of incoming signals, i.e.:

$$X_i^o = \sum_i \overline{Y}_i f_i \rightarrow X_i^o = \frac{\sum_i \overline{Y}_i f_i}{\sum_i \overline{Y}_i} \tag{16}$$

More precisely, the class label for the s^{th} sample is obtained by the maximum $\{O_{sk}\}$ value as follows:

$$C_s = max_{k=1,2,....K}\{O_{sk}\} \tag{17}$$

The consequent parameters thus identified are optimal (in the consequent parameter space) under the condition that the premise parameters are fixed. The knowledge system's weights are conventionally identified by performing maximum likelihood estimation. Given a training data set $Z^n = (y(k), x(k))_{k=l}^n$, the task is to find a weight vector which minimizes the following cost function:

$$J_n(w) = \frac{1}{n}\sum_{k=1}^{n} (y(k) - \hat{y}(x(k), w))^2 \tag{18}$$

As the knowledge based system, $\hat{y}(x(k), w)$ is much interrelated with respect to the weights, linear optimization techniques cannot be applied. The adopted Neuro-fuzzy system has number of inputs (n) (representing the features) and (m) outputs (representing classes of features). In reference to **Figure 7**, there are dataset about the mobile area and zone of navigation. This is due to the large amount of information coming from the visual system. Here comes the potential of employing the PCA to reduce the dimensionality of the input spaces.

4.3. Fuzzy decision based system

The last stage of the mobile robot maps learning system, is the fuzzy decision system. Within this stage, the hard decisions are undertaken by the mobile robot system during a course of navigation. A fuzzy system is constructed typically from the following rules:

$$\begin{gathered} D = G \cap C \Leftrightarrow \mu_D(a) = \mu_G(a) \bigwedge \mu_{GC}(a), \;\; for \; a \in A \\ a^* = \text{ARG}max\big(\mu_G(a) \wedge \mu_c(a)\big), \ldots\ldots \;\; a \in A \end{gathered} \tag{19}$$

The rules (*if*) parts, are identical to an ordinary fuzzy *IF-THEN* rules. Given an fuzzy inputs $u = (u_1, u_2, \ldots., \ldots u_n)^T$ the output $\hat{y}(k)$ of a fuzzy system is computed as the weighted average of the y_s^l, that is:

$$\hat{y}(k) = \frac{\sum_{i=l}^{m} y^l w^l}{\sum_{i=l}^{m} w^l} \tag{20}$$

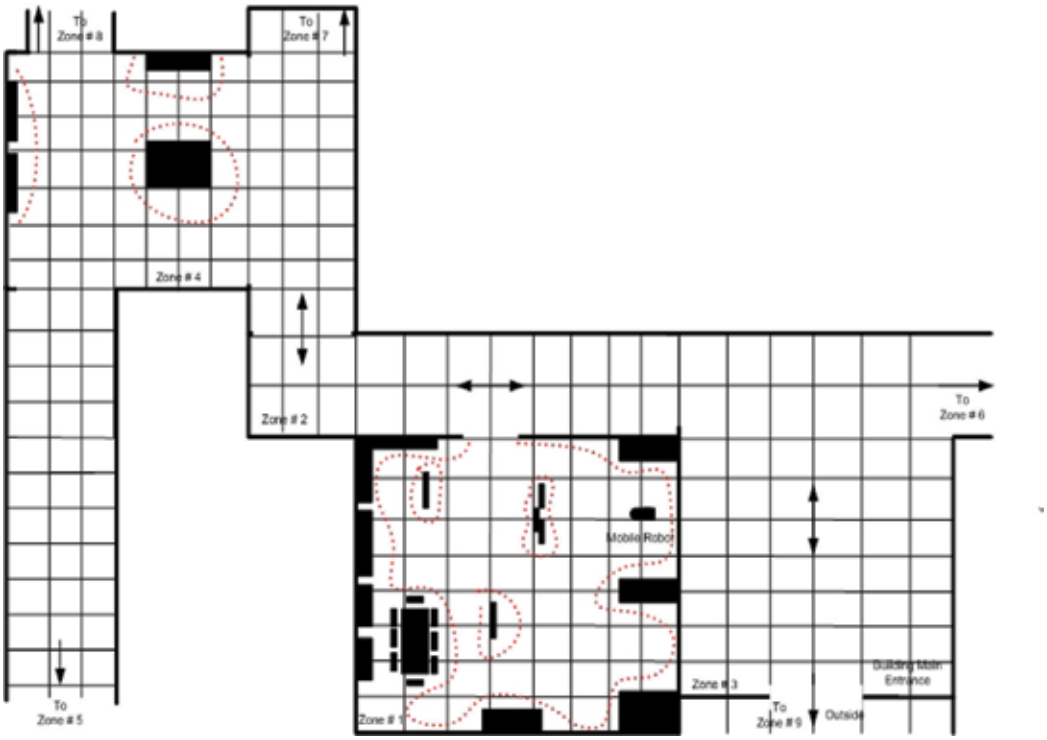


Figure 7. Data from navigation spaces in different zones and areas, (z_1, z_2, z_3, …. z_m, a_1, a_2, a_3, …. A_n), they represent inputs to the PCA.

weights w^i are computed as

$$w^i = \prod_{i=1}^{n} \mu c_i^l(x_i) \tag{21}$$

A dynamic TSK fuzzy system is constructed from the following rules:

$$\begin{aligned} &if\ (y(k))\ is\ \left(A_l^P\right)\ \delta\ (y(k_{k-n+l})\ is\ \left(A_n^P\right)\ \delta\ (u(k))\ is\ \left(B^P\right) \\ &\Rightarrow \left(y_{n+1} = a_l^p y_k + \ldots a_n^P y_{k-n+l} + b^P u(k)\right) \end{aligned} \tag{22}$$

where $\left(A_n^P\right)$ and $\left(B^P\right)$ are fuzzy sets, $\left(a_l^p\right)$ and $\left(b^P\right)$ are constants, $p = (1, 2, \ldots., n, u(k))$ are the inputs to the system, and $u(k) = (x_1(k),\ x_2(k), \ldots, x_{n+1}(k))$ is the fuzzy system knowledge vector. Typically, the output of the fuzzy decision based system is computed as:

$$x(k) = \frac{\sum_{p=1}^{n} x^p(\beta) v^p}{\sum_{p=1}^{n} v^p} \tag{23}$$

	Inputs		Outputs	
1	x	*Robot Zone*	O_1	*Behavior1*
2	y	*Robot Area*	O_2	*Behavior2*
3	z	*Robot Segment*	O_3	*Behavior3*
4	w	*Delicate Observations*	O_4	*Behavior4*

Table 1. Fuzzy decision based system input-outputs representation.

	Inputs			Outputs	
1	First mobile stimuli	Identification of zone of navigation	Zones identification, and obstacles in zones, z_1, z_2, z_3, z_4, z_5, ……. z_m	First mobile behavior. Behavior1	Rotate around, move robot forward, move robot backward, rotate right, rotate left, …
2	Second mobile stimuli	Identification of area of locality	Areas identification, and obstacles in areas. Obstacles in a1: area floor, obstacles in a2: Out_Corridor, Obstacles in a3: building. Entry, Obstacles in a4:	Second mobile behaviour. Behavior2	Image focus, image capture, … image processing of a scene.
3	Third mobile stimuli	Identification of segment of navigation	Obstacles at different segments within an area, ..	Third mobile behaviour. Behavior3	Video recording zooming with video capture, ..
4	Fourth mobile stimuli	Mobile robot delicate sensory observation	Rotate around, move robot forward, move robot backward, rotate right, rotate left, …	Fourth mobile behaviour. Behavior4	Delicate mobile action.

Table 2 Neuro-fuzzy classifier, input-outputs representation.

where $x^p(k_1)$ is given in Eq. (23) and:

$$v^p = \prod_{i=1}^{n} \mu A_i^p(x_k) \mu B^p(u(k)) \tag{24}$$

The mobile robot training datasets, do consist of four inputs. There are also four output parameters. This is summarized in **Tables 1** and **2**.

5. The experimentation

Within this section, we shall discuss few experimentations results. In order to implement the proposed navigation methodology, the (914 PC BOT) has been reengineered in such a way to allow more control and observations to be communicated through. The main high-level coding was achieved using Matlab. Matlab toolboxes have been integrated in such a way to allow PCA computation, Neuro-fuzzy learning capabilities, and fuzzy decision making routines. This is further indicated to in **Figure 8**.

http://dx.doi.org/10.5772/intechopen.81195

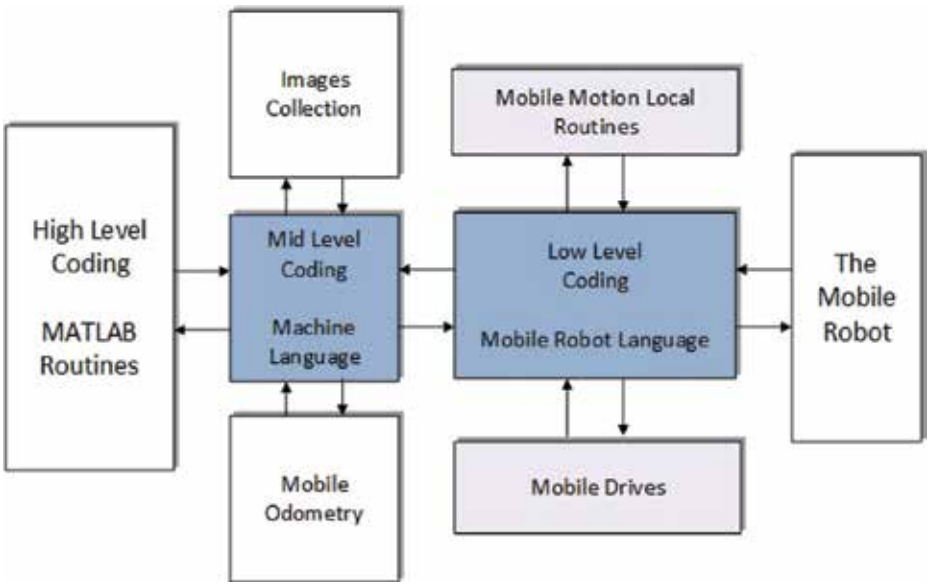


Figure 8. Implementation system hierarchy.

5.1. Behaviour knowledge building

For building the mobile robot behaviour at localities, the mobile system was maneuvered over a space in the laboratory for several trails. Typical physical readings from the robot odometry, and sensory observations were recorded. Typical readings are shown in **Figure 9**. Different mobile behaviors (for learning) were also recorded, beside the odometry, and sensory observations.

5.2. Navigation intelligence

Building the mobile robot navigation intelligence is the next phase. This phase requires blinding all the previous inputs (readings, situations, and behaviors). This will help to take the most appropriate actions. The designated learning and decision making architecture is a Neuro-fuzzy. Typical information, that constitute the Neuro-fuzzy classifier inputs are:

The classifier inputs:

ZONES of navigation. This represents typical zones where the mobile robot is located.

AREAS of navigation. This represents typical areas where the mobile robot is moving.

SEGMENT of navigation. This represents typical segment where the mobile robot is moving.

OBSERVATIONS. This represents typical Observations, the mobile is experiencing at a locality.

The classifier outputs:

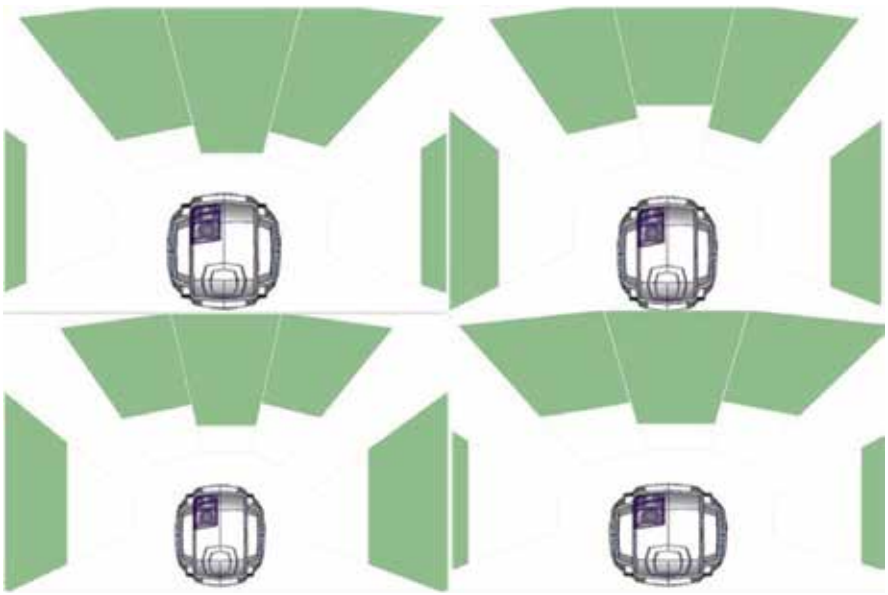


Figure 9. Mobile robot real sensory observations, by experimentation. Dataset have been collected through a number of runs for the PCA.

First mobile behavior. Behavior1, Rotate Around, Move Robot forward, Move Robot Backward, Rotate Right, Rotate Left, ….

Second mobile behavior. Behavior2, Image Processing of a Scene, … .

Third mobile behavior. Behavior3, Video Recording.

Fourth mobile behavior. Behavior4, Delicate Mobile Action.

With such inputs and system outputs, a good degree of a mixture of mobile behaviors can therefore be created. This is further listed below:

The implementation system hierarchy, is shown in **Figure 8**. An adequate of mobile intelligence was created for a mobile navigation within hazardous environments. Inputs to the Neuro-fuzzy decision based system are coming from the PCA network.

5.3. Fuzzy if-then decision system

In addition, the four system inputs-outputs, do represent the system outputs the mobile robot should undertake also at any particular situation. While relying on the fuzzy (if-then) statements, we are able to make further final decision to be undertaken by the mobile robot. Within this sense, we are able to build an (if then statement), as follows:

Typical Fuzzy Rules are:

If (Input_#1 is ….. and Input_#2 is ….) then (Output_#1 is …. and Output_2 is ….) .. ...

If (Input_#3 is ….. and Input_#2 is ….) then (Output_#4 is …. and Output_2 is ….) .. ...

If (Mobile is in zone1 ….. and in area1 ….) then (do image FOUCS).

If (Mobile is in zone1, ….. and in segment2 and, Mobile special task) then (set an ALAM and GAZE).

If (Mobile is in zone3 and in area5 and segment 3, and Image Capture), … then (do image analyze).

If (Mobile in zone41 and in area2 and, … and Special task, then (move back).

Given the defined navigation strategy, the mobile robot is able to undertake much detailed navigation and behaviors tasks.

6. Conclusions

Learning mobile robot navigation behavior, is an essential feature, for improved navigation. In addition, it helps to gain further understanding about the spaces of navigation. In this study, navigation maps details have been created while relying on dataset collected by SLAM routines. Due to enormous sensory and environmental data observation to be analyzed during navigation, we have reduced the dimensionally and size of environmental and sensory observation information with PCA technique. Reduction of environment information (*i.e. getting features*), are hence used as learning inputs to a neuro-fuzzy classifier. Examples of Neuro-fuzzy feature inputs are, navigation locations, areas, …, and behaviors related to particular localities. The final stage of mobile robot map building is a fuzzy decision based system. Within this stage, mobile robot navigation decisions are undertaken. With multi-levels of mobile robot sensory and navigation observation dataset, we have designed a learning system for mobile robot maps learning with navigating capabilities.

Author details

Ebrahim A. Mattar

Address all correspondence to: ebmattar@uob.edu.bh

College of Engineering, University of Bahrain, Sukhair, Kingdom of Bahrain

References

[1] Janglová D. Neural networks in mobile robot motion. Inernational Journal of Advanced Robotic Systems. 2004;**1**(1):15-23 ISSN 1729-8806

[2] Raymond A, Tayib S, Hall L. Remote controlled, vision guided, mobile robot system. In: Intelligent Robots and Computer Vision XVI: Algorithms, Techniques, Active Vision, and Materials Handling, vol. 3208; Proceedings of SPIE. The International Society for Optical Engineering; 1997. pp. 126-132

[3] Gang P, Xinhan H, Jian G, Cheng L. Camera modeling and distortion calibration for mobile robot vision. In: Proceedings of the 7th World Congress on Intelligent Control and Automation (WCICA), WCICA'08. 2008. pp. 1657-1662

[4] Bonin-Font F, Ortiz A, Oliver G. Visual navigation for mobile robots: A survey. Journal of Intell Robot System. 2008;**53**:263-296. DOI: 10.1007/s10846-008-9235-4

[5] Abdul B, Robert S, Yahya K, Gloom M. A hybrid approach towards vision based self-localization of autonomous mobile robots. In: Proceedings of the International Conference on Machine Vision, ICMV-2007. 2007. pp. 1-6

[6] Araujo R. Prune-able fuzzy ART neural architecture for robot map learning and navigation in dynamic environments. IEEE Neural Network Transactions. 2006;**17**(4)

[7] Vlassis N, Papakonstantinou G, Tsanakas P. Robot map building by Kohonen's self-organizing neural networks. In: Proceedings of 1st Mobinet Symposium; Athens, Greece. 1997

[8] Filliata D, Meyer J. Map-based navigation in mobile robots: I. A review of localization strategies. Cognitive Systems Research. 2003;**4**:243-282

[9] Miftahur R, Rahman M, Rizvi H, Haque Abul L, Towhidul IM. Architecture of the vision system of a line following mobile robot operating in static environment. In: Pakistan Section Multitopic Conference (INMIC 2005). 2005

[10] Tomohiro S, Yoshio M, Taichi K, Masayuki I, Hirochika I. Development and integration of generic components for a teachable vision-based mobile robot. IEEE/ASME Transactions on Mechatronics. 1996;**1**(3):230-236

[11] Ryosuke M, Fumiaki T, Fumio M. Skill acquisition of a ball lifting task using a mobile robot with a monocular vision system. In: IEEE International Conference on Intelligent Robots and Systems. 2006. pp. 5141-5146

[12] Vlassis N, Papakonstantinou G, Tsanakas P. Robot map building by Kohonen's self-organizing neural networks. In: Proceedings of the 1st Mobinet Symposium; Greece. 1997

[13] Changhan P, Sooin K, Joonki P. Stereo vision-based autonomous mobile robot. In: Intelligent Robots and Computer Vision XXIII: Algorithms, Techniques, and Active Vision; vol. 6/6. Proceedings of SPIE. The International Society for Optical Engineering; 2005

[14] Thrun S. Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence Journal. 1998;**99**:21-71

[15] Nima F, Mohammad T, Sadaf S. Intelligent real time control of mobile robot based on image processing. IEEE Proceedings Intelligent Vehicles. 2007;**IV**:410-415

[16] Hall EL, Ghaffari M, Liao X, Alhaj ASM. Intelligent robot control using an adaptive critic with a task control center and dynamic database. In: Intelligent Robots and Computer Vision XXIV: Algorithms, Techniques, and Active Vision, vol. 6384. Proceedings of SPIE. The International Society for Optical Engineering; 2006

[17] Abdul B, Robert S, Jason G, Yahya K, Usman M, Muhammad H, et al. Stereo vision based self-localization of autonomous mobile robots. In: Proceedings (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Robot Vision—Second International Workshop, RobVis., vol. 493-1 LNCS. 2008. pp. 367-380

[18] Hall L, Liao X, Alhaj AM. Learning for intelligent mobile robots. In: Proceedings of SPIE, vol. 5267. The International Society for Optical Engineering; 2003. pp. 12-25

[19] Ya-Chun C, Hidemasa K, Yoshio Y. Novel application of a laser range finder with vision system for wheeled mobile robot. In: Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM. 2008. pp. 280-285

[20] Nazari V, Naraghi M. A vision-based intelligent path following control of a four-wheel differentially driven skid steer mobile robot. In: Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision, ICARCV-2008. 2008. pp. 378-383

[21] Mohan S, Peter S. Color learning and illumination invariance on mobile robots: A survey. Robotics and Autonomous Systems. 2009;**57**(6–7):629-644

[22] Masayuki I, Fumio K, Satoshi K, Hirochika I. Vision-equipped apelike robot based on the remote-brained approach. In: Proceedings IEEE International Conference on Robotics and Automation, vol. 2. 1995. pp. 2193-2198

[23] Xiaoning C, Yuqing H, Gang L, Jin G. Localization algorithm of mobile robot based on single vision and laser radar. In: Proceedings of the 7th World Congress on Intelligent Control and Automation, WCICA'08. 2008. pp. 7661-7666

[24] Andreja K, Sanjin B, Ivan P. Mobile robot self-localization in complex indoor environments using monocular vision and 3D model. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics AIM. 2007

[25] Takeshi S, Hashimoto H. Human observation based mobile robot navigation in intelligent space. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006. 2006. pp. 1044-1049

[26] Manoj K, Ernest L H. Intelligent robot control using omnidirectional vision. In: Proceedings of SPIE, vol. 1825. The International Society for Optical Engineering; 1993. pp. 573-584

[27] Robot Cartograsphy: ROS + SLAM. http://www.pirobot.org/blog/0015/

Chapter 5

Mobile Robot Navigation in Indoor Environments: Geometric, Topological, and Semantic Navigation

Ramón Barber, Jonathan Crespo, Clara Gómez,
Alejandra C. Hernámdez and Marina Galli

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79842

Abstract

The objective of the chapter is to show current trends in robot navigation systems related to indoor environments. Navigation systems depend on the level of abstraction of the environment representation. The three main techniques for representing the environment will be described: geometric, topological, and semantic. The geometric representation of the environment is closer to the sensor and actuator world and it is the best one to perform local navigation. Topological representation of the environment uses graphs to model the environment and it is used in large navigation tasks. The semantic representation is the most abstract representation model and adds concepts such as utilities or meanings of the environment elements in the map representation. In addition, regardless of the representation used for navigation, perception plays a significant role in terms of understanding and moving through the environment.

Keywords: robot navigation, environment modeling, topological navigation, semantic navigation, geometric navigation

1. Introduction: navigation problem

Navigation of mobile robots has been traditionally understood as solving the problem proposed by these three questions (Levitt [1]):

- Where am I?
- Where are other places related to me?
- How do I get to other places from here?

The answer to these questions refers to localization to determine where the robot is, path-planning to know how to reach other places from where the robot is, and navigation to perform the trajectory commanded by the path-planning system.

This approach has ruled robot navigation paradigm and many successful solutions have been proposed. However, the advances in technical developments allow a wider thought of mobile robot navigation giving a solution for a "bigger" problem. This solution would give answer to two additional questions, which are:

- How is this place like?
- How is the structure of the environment I am in?

The answer to these new questions focuses on the importance of perception to determine the place and the elements of the place where the robot is and the importance of modeling the environment to determine the structure of the place and infer the connections between places. When robots are integrated in human environments and live together with humans, perception and modeling gain importance to enable future tasks related to service robots such as manipulation or human-robot interaction. In this chapter, we give a conceptual answer to the traditional robot navigation question and to the questions proposed above.

Robot navigation has been tackled from different perspectives leading to a classification into three main approaches: geometric navigation, topological navigation, and semantic navigation. Although the three of them differ in their definition and methods, all of them have the focus of answering the same questions.

From the beginning, authors have focused on generating metric maps and moving through the map using metric path planners. The most well-known algorithm to build metric maps is simultaneous localization and mapping (SLAM) as proposed by Bailey & Durrant-Whyte [2]. Wang et al. [3] use SLAM and Rapidly-exploring Random Tree (RTT) planning with Monte Carlo localization to drive a wheelchair in indoor environments. Pfrunder et al. [4] use SLAM and occupancy grids to navigate in heterogeneous environments. However, as larger maps are considered, it is computationally expensive to keep metric maps and other authors have focused on topological representations. Fernández-Madrigal et al. [5], design a hierarchical topological model to drive a wheelchair and perform reactive navigation and path-planned navigation. Ko et al. [6] works with topological maps where nodes are bags of visual words and a Bayesian framework is used for localization.

Regarding topological navigation, since the first developments, the global conception of the system has attracted the interest of several authors. In Kuipers & Levitt [7], the authors presented a four-level hierarchy (sensorimotor interaction, procedural behaviors, topological mapping, and metric mapping) to plan trajectories and execute them. Giralt et al. [8] defined a navigation and control system integrating modeling, planning, and motion control and stated that the key to autonomous robot was system integration and multisensory-driven navigation. Mataric [9] defined a distributed navigation model whose main purposes were collision-free path-planning, landmark detection, and environment learning. In Levitt [1], Levitt established a visual-based navigation where landmarks are memorized and paths are sequences of

http://dx.doi.org/10.5772/intechopen.79842

landmarks. These authors tackled for the first time robot navigation concepts from a topological approach and some of them noticed the importance of perception in the navigational processes.

At the last level of the semantic navigation paradigm, the ability to reason and to infer new knowledge is required. In today's world of robotics, there is a general tendency to implement behavioral mechanisms based on human psychology, taking as example the natural processing of thought. This allows a greater understanding of the environment and the objects it contains. This trend is also valued in the field of mobile robot navigation. Navigators have been increasing their level of abstraction over time. Initially, navigation was solved with geometric navigators that interpreted the environment as a set of accessible areas and areas that were not accessible. Then, the concept of node and the accessibility between nodes was introduced, which allowed increasing the level of abstraction generating graphs and calculating trajectories with algorithms of graphs. However, the level of abstraction has increased a step further, introducing high-level concepts that classify the rooms according to more complex and abstract data such as the utility and the objects they contain.

Another important aspect is the collection of the information of the environment, which has to be available for the navigation systems, among other tasks carried out by mobile robots. This information is provided by a perceptual system, therefore non-trivial problems appear related to object detection and place recognition. One of the most challenging issues in scene recognition is the appearance of a place. Sometimes, the same place may look different or different places may look similar. Also, the position of the robot and the variation of its point of view can affect the identification of the place when it is revisited. Environment models and the way to define the navigation tasks can be remodeled taking into account new technologies and trends. This will provide more autonomy to mobile robots and will help the interaction with humans in their usual environments. In this trend, vision is the main sensor for navigation, localization, and scene recognition. Place recognition is a very well-known challenging problem that not only has to do with how a robot can give the same meaning that a human do to the same image, but also with the variability in the appearance of these images in the real world. Place recognition has a strong relation with many major robotics research fields including simultaneous localization and mapping.

2. Geometric navigation

Geometric navigation consists of moving the robot from one point of the environment to another one, given those points by its coordinates in a map. The map of the environment is classically represented by a grid of points, and the trajectory between two points is a sequence of points the robot must reach in the given order. The controller of the robot must reach the next point of the path closing a loop with the encoder information about its position (distance and angle). One of the main objectives in the geometric navigation researches is the path-planning task from an initial point to a final point, creating an algorithm able to find a path ensuring completeness.

Although the path-planning task is widely treated in mobile robots, computational capacity has increased exponentially and more complex algorithms can be developed. Algorithms try to find the shortest or fastest path while maintaining safety constraints. Another challenge is to try to get solutions that provide smoother trajectories, trying to imitate the human trajectories.

In the literature, many different algorithms can be found. One of the most important precedents were the works of LaValle [10], where a classification into two big groups depending on the way information is discretized is proposed: combinatorial planning and sampling-based planning. Combinatorial planning constructs structures containing all the necessary information for route planning; see De Berg et al. [11]. Sampling-based planning is based on an incremental representation of space and uses collision-free algorithms for path search. Here are some of the algorithms most used in the path-finding problem.

2.1. Deterministic algorithms

One of the first approaches tries to get all the possible paths between two points and choose the shortest one. Two example algorithms use potential fields and Voronoi diagrams.

2.1.1. Potential fields

The potential fields method is based on reactive planning techniques, which can be used to plan locally in unexplored environments. This method assigns to the obstacles similar characteristics that an electrostatic potential might have; in this way, the robot is considered as a particle under the influence of an artificial potential field that pulls it toward a target position, **Figure 1a**. The generation of trajectories is due to an attractive field toward the final position and another repulsive one with respect to the obstacles [12].

In this way, navigation through potential fields is composed of three phases:

- Calculation of the potential acting on the robot using the sensor data.
- Determination of the vector of the artificial force acting on the robot.
- Generation of the movement orders of the robot.

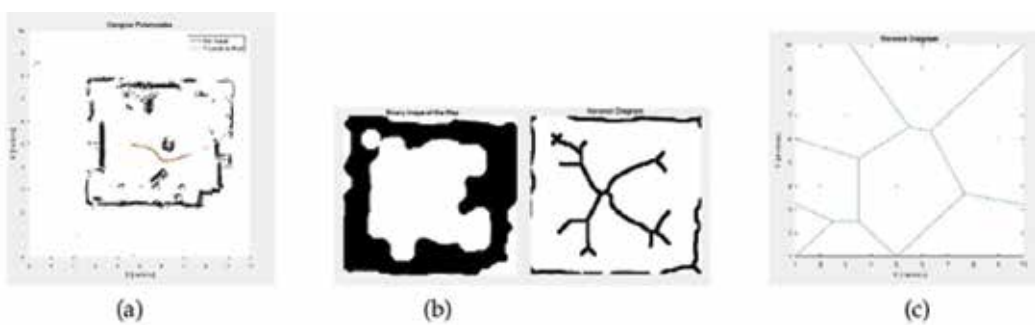


Figure 1. Trajectories created using potential fields and Voronoi diagram methods. (a) Trajectory of a real robot created using Potential Fields Method, (b) Trajectories created using Voronoi Diagram by Skeletonization and (c) Trajectories created by Voronoi Diagram Method.

On the one hand, the attractive potential must be a function of the distance to the final destination, decreasing when the robot approaches this point. On the other hand, the repulsive potential should only be influenced when the robot is at a dangerous distance from obstacles. Therefore, the potential fields method allows to be performed in real time, as a local planner, considering obstacles only when the robot is at a minimum distance from them.

The problem that exists in this type of method is the local minimums [13], places where the potential is null but it is not the final position.

2.1.2. Voronoi diagram

The generation of Voronoi diagrams seeks to maximize the distance between the robot and the obstacles, looking for the safest path between two points in the space [14]. In this way, the diagram is defined as the locus of the configurations that are at the same distance from the obstacles.

The algorithm divides the space into sections formed by vertices and segments that fulfill the cost function of maximum distance between obstacles, **Figure 1**. Then, the trajectory is sought from an initial point to the objective. For a more realistic representation, obstacles are considered as polygons, since physically an obstacle is not a point.

One way of building the Voronoi diagram is using image-processing methods (skeletonization) based on the method of Breu [15]. These present a linear (and therefore asymptotically optimal) time algorithm in order to calculate the Euclidean distance of a binary image.

The algorithm is built with the pixels that result after performing the morphological operations on the image, with the nodes being the points where the lines that pass through the pixels of the image intersect.

A trajectory generated by Voronoi diagrams has the disadvantage that it is not optimal from the point of view of length traveled, it can also present a large number of turns [16]. Moreover, this method is not efficient for more than two dimensions.

2.2. Probabilistic algorithms

Over the years, a large number of solutions for the navigation problem have been presented using algorithms with random components, specifically in generic environments and with a large number of dimensions.

2.2.1. PRM algorithm

Probabilistic roadmap (PRM) is a trajectory planner that searches the connectivity of different points in the free space from a starting point to the final goal avoiding collisions with obstacles, using random sampling methods [17].

If the environment in which the robot is located is very complex, this type of algorithm allows finding a solution without a large computational requirement, so it is used in environments with a large number of dimensions.

One of the main problems of PRM is that the solutions it finds do not have to be the optimal trajectory. Also, since the way it generates the nodes is completely random, it produces a nonhomogeneous distribution of the samples in the space. These two disadvantages are seen in **Figure 2a**.

2.2.2. Rapidly exploring random tree algorithm

Rapidly exploring random tree (RRT) [18] provides a solution by creating random branches from a starting point. The collision-free branches are stored iteratively and new ones are created until the target point is reached. The algorithm is started with a tree whose source is a single node, the starting point. In each iteration, the tree expands by selecting a random state and expanding the tree to that state. The expansion is performed by extending the nearest node of the tree to the selected random state, which will depend on the size of the selected step. The algorithm creates branches until the tree takes a certain extension approaching the goal.

The size of the step is an important parameter of the algorithm. Small values result in slow expansion, but with finer paths or paths that can take fine turns [19].

2.3. Fast marching square algorithm

Fast marching square (FM^2) is a path-planning algorithm, that searches for the optimal path between two points. It uses the fast marching method (FMM) as a basis for calculation. It is a modeling algorithm of a physical wave propagation.

The fast marching method uses a function that behaves similar to the propagation of a wave. The form that this wave propagates, following the Eikonal equation, from an initial point to reach a goal position is the most efficient way in terms of time to reach it. The fast marching algorithm calculates the time (T) that the front of the wave, called interface, spends to reach each point of the map from a starting point. The FMM requires as previous step a

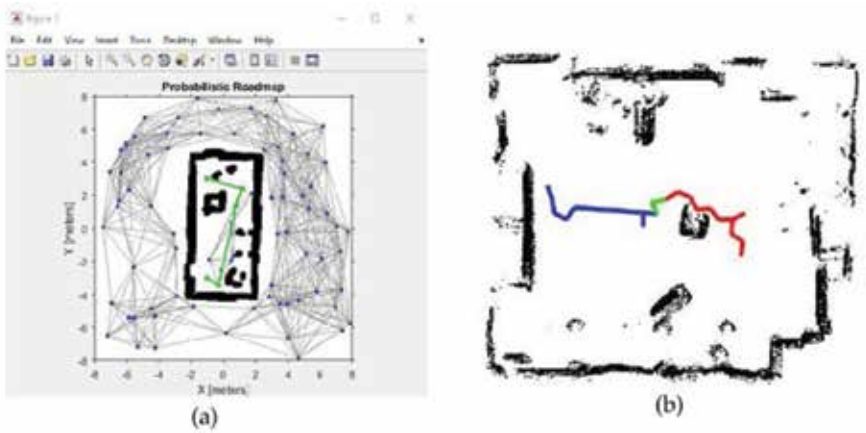


Figure 2. PRM generated and trajectory created using bidirectional RRT. (a) PRM generated for a real environment and (b) Trajectory created using Bidirectional Rapidly Exploring Ramdom Trees.

http://dx.doi.org/10.5772/intechopen.79842

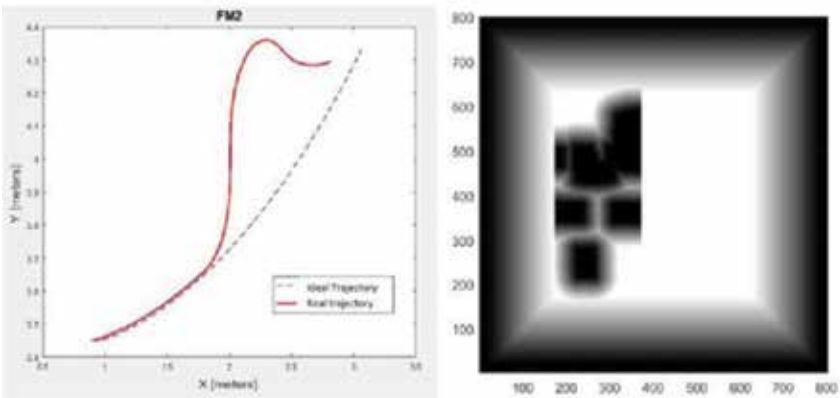


Figure 3. Trajectory calculated by fast marching square.

discretization of the environment in a grid of cells, on which the Eikonal equation is solved iteratively (**Figure 3**).

The trajectories obtained using FMM present two drawbacks: great abruptness of turns and trajectories very close to the obstacles. This makes it impossible to use the algorithm as a trajectory planner in real robotics. The main change in FM^2 [20] solves these problems by generating a speed map that modifies the expansion of the wave taking into account the proximity to obstacles.

3. Topological navigation

Topological navigation refers to the navigational processes that take place using a topological representation of the environment. A topological representation is characterized by defining reference elements of the environment according to the different relations between them. Reference elements are denominated nodes and the relations between them are characterized as arches. The aim of topological navigation is to develop navigational behaviors that are closer to those of humans in order to enhance the human-robot understanding. Summing up the main implications of topological navigation [21]:

- Topological navigation permits efficient planning. However, as they are based on relations, they do not minimize distance traveled or execution time.
- Topological navigation does not require precise localization.
- It is easy to understand by humans due to its natural conception.
- A complex recognition model is needed.
- Huge diagrams are involved in large environments and diagrams scale better than geometrical representations.

Topological representation classifications, as the one proposed by Vale & Ribeiro [22], differentiate mainly two ways of representing the environment: topological representations based on movements, for example the works developed by Kuipers & Byun [23], and topological representations based on geometrical maps, as proposed by Thrun [24]. These two conceptions differ mainly in the spatial relation between the real world and its representation. Regarding topological maps based on geometry, an exact relation between the environment and the representation is mandatory. Every topological node is metrically associated with a position or place in the environment, whereas, in topological representations based on movements, it is not necessary to have a metrical correspondence with the elements of the environment. An example of the same environment represented as a topological map based on geometry and as a topological map based on movements is shown in **Figure 4**.

In topological representations based on geometry, nodes normally correspond to geometrical positions (characterized as (x, y, θ)) that correspond to geometrical events such as junctions, dead-ends, etc. and arches correspond to the geometrical transition between positions. In topological representations based on movements, the relation between nodes is more abstract as it does not have a geometrical meaning, being a qualitative relation instead of a quantitative one. That is why arches can be associated to different and more complex behaviors. In addition, nodes are associated to important sensorial events which can be determined for each application ranging from important geometrical events to landmarks and objects.

Although there are substantial differences between topological representations based on geometry and topological representations based on movements, both of them share the definition of the required modules so the system works efficiently. These modules will be explained in the following subsections, which are: modeling of the environment, planification and navigation, and perception interface.

3.1. Modeling of the environment as a topological graph

A topological graph, as defined by Simhon & Dudek [25], is a graph representation of an environment in which the important elements of an environment are defined along with the transitions among them. The topological graph can be given to the system by the user or can be built through exploration. Exploration strategies differ if the system works with a topological

Figure 4. Topological representations based on geometry (left) and on movements (right).

http://dx.doi.org/10.5772/intechopen.79842

representation based on geometry or a topological representation based on movements. In the case of using representations based on geometry, exploration and map acquisition strategies such as the ones explained previously for geometrical representations can be used adding a processing stage to extract the relevant geometrical positions. In the case of representations based on movements, strategies like Next Best View, as proposed in Amigoni & Gallo [26], or Frontier Exploration, as in the work proposed by Arvanitakis et al. [27] can be used. Generally, these representations are translated into a text file that lists the information of the graph in order to maximize the graph efficiency. In **Figure 5**, an example of a map text file structure and its graphical interpretation is shown. In this example, a topological representation based on movements is used so the position of the nodes in the graph is not necessarily linked to the position of reference elements in the environment.

This text file contains all the information required for navigation. Nodes are ordered according to an identifier and they are associated with their corresponding event type (odometry, marker, closet, etc.). Arches are associated with the behavior or translational ability that the robot has to perform (GP, go to point; R, turn; etc.).

A model of the environment can be formed by several topological graphs containing different information or representing different levels of abstraction, in that case the environment is modeled as a hierarchical topological graph.

3.2. Topological planification and navigation

Topological navigation behaviors are determined mainly by the path-planning and navigation (meaning strict motor abilities execution) strategies as considered in Mataric [28]. The different nodes conforming the path are associated with the topological place where the robot perceives a sensorial stimulus or where it reaches a required position. In addition, while navigating, the behavior of the robot is subjugated to real-time perception and movement in the environment.

Topological planification is in charge of finding the topological route that the robot has to follow; it has to determine the path that the robot has to perform in order to move between two

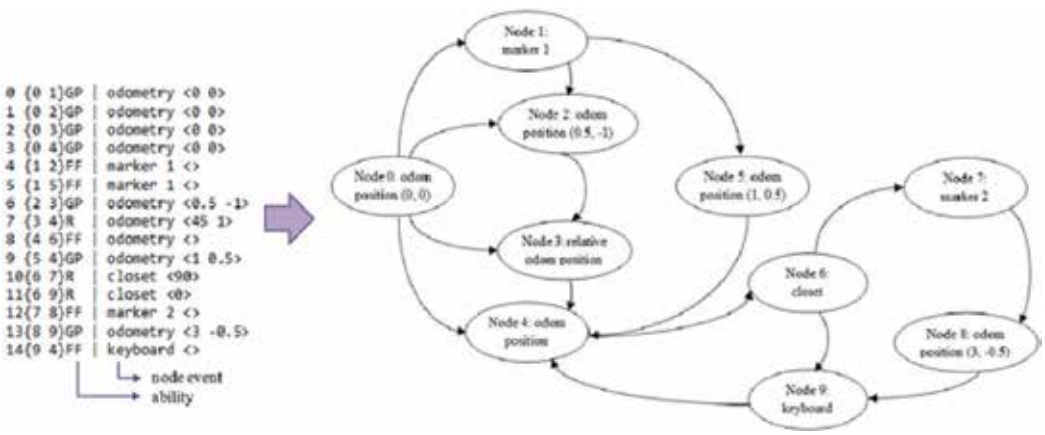


Figure 5. Example of a topological map and its graphical representation.

nodes. The path is a subgraph of the original graph of the environment. In order to plan a trajectory in a topological representation of the environment, a graph path-planning algorithm has to be used. There are many algorithms in the literature for this purpose, such as Dijkstra's algorithm, Skiena [29]. The main objective of Dijkstra's algorithm is to obtain the shortest path between nodes in a graph according to some heuristics or cost function. Given an initial node, it evaluates adjacent nodes and chooses the node that minimizes the cost function. This process is iterated until the goal node is reached or every connection between nodes has been explored. Using traditional algorithms such as Dijkstra, many modifications can be implemented to establish heuristics that fit better to real environments. For example, the cost of a translation between nodes can be varied according to previous executions or pursing a specific global behavior, such as the personality factor proposed in Egido et al. [30].

The navigator is in charge of performing the path and reaching the goal node analyzing events and abilities or positions and transitions. An ability is the order the robot has to execute to reach coming nodes and it is intrinsically related to motor control. An event is the sign indicating the robot has reached a node, through sensorial information or odometrically in the case of representations based on geometry. A navigator can be based on a single behavior or on multiple behaviors. Topological representations based on geometry are all based on a single behavior that can be understood as "Going to point" or "Reaching next position." Topological representations based on movements can be based on single behaviors also or define a set of behaviors that would be used for optimizing the transitions between nodes. A multiple-behavior system contains several strategies and abilities in order to perform navigation behaviors and it should enable the inclusion of new strategies. Some of the common abilities implemented are Go to point, Turn, Contour following, and Go to object. Navigation systems are complemented with reactive obstacle avoidance modules to guarantee safe operation.

3.3. Managing the information of the environment: perception interface

If the topological navigation system is designed to work with multiple exteroceptive and proprioceptive events and it has to handle them simultaneously, these events have to be managed carefully. In order to manage the information that will be assigned to the nodes, a new module is needed. In this chapter, we will refer to this module as the perception interface.

The perception interface is decoupled from the system translating the specific information of each perception to a general structure which interacts with the other modules of the topological system, as shown in **Figure 6**. When translating the information to a general structure, the power of the system is multiplied exponentially, as adding new perception types becomes very

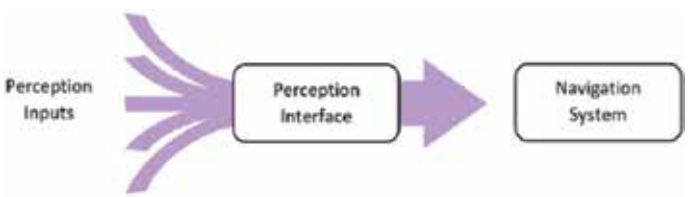


Figure 6. Perception interface and event manager modules distribution.

http://dx.doi.org/10.5772/intechopen.79842

simple. Perceptions are mainly based on vision, such as object detection and landmark detection, but some perceptions such as magnetic signals, ultrasounds, or proprioceptive perceptions such as odometry can be used. Another advantage of the perception interface is the possibility of establishing priorities and relations between perceptions easily.

4. Semantical navigation

The current tendency in robotics is to move from representation models that are closest to the robot´s hardware such as geometric models to those models closer to the way how humans reason, with which the robot will interact. It is intended to bring closer the models the way robots represent the environment and the way they plan to the way the humans do. The current trend is to implement behavior mechanisms based on human psychology. Robots are provided with cognitive architectures in order to model the environment, using semantics concepts that provide more autonomy, and which helps the navigation to be robust and more efficient.

In theory, any characteristic of the space can be represented on a map. But in general, it tends to identify a map with geometric information more or less complemented with additional information. When a mobile robot builds a map, the techniques used generally ignore relevant descriptive information of the environment and quite close to the process of made by humans, such as the navigation of the environment, the nature of the activity that there is, what objects it contains, etc.

The problem of the construction of semantic maps consists of maps that represent not only the occupation and geometric appearance of the environment but the same properties. Semantics is needed to give meaning to the data, mainly to make the simplest interpretation of data. The application of this idea to the construction of maps for mobile robots allows a better understanding of the data used to represent the environment and also allows an exchange of information between robots or between robots and people, if needed, easily. It also allows to build more accurate models and more useful environment models.

Semantic navigation is another step to the logical representation, it implies an additional knowledge about the elements of the world and it allows the robot to infer new information. For instance, the root can perform this type of associations: “There is a photocopier in this room-> there must be paper nearby”.

Semantic navigation allows the robot to relate what it perceives to the places in which it is located. This way, an environment model is managed using the objects and on the concepts that they represent. All this information is used to classify the place where objects are located and it is used to reach a given location or a specific target. Therefore, the robot can find places or rooms Vasudevan & Siegwart [31] semantically related with the target, even if it is in a little-explored or unknown environment. This navigation level allows to complete the information that is necessary from a partial knowledge of the environment.

Semantic navigation is related to classification methods of the environment object and places, which provides more effectiveness, as is described in Vasudevan & Siegwart [32]. An example

of place identification using a Naive Bayes Classifier to infer place identification is shown in Duda et al. [33] and Kollar & Roy [34]. These works show that relations between object-object and objet-scenario can be used to predict the location of a variety of objects and scenarios.

4.1. Knowledge representation of the environment

One of the most important issues is how the robot models the environment. In semantic navigation, a large amount of information related to the environment and to the objects of the environment is used in the environment representation. For this representation, an ontology can be used to define the concepts and the links between objects in the environment. **Figure 7a** shows the concepts of the ontology and its relations.

4.2. An ontology design through a database

The ontology design can be implemented using different tools. In this work, a relational model using a database is proposed. **Figure 7b** shows a relational database diagram scheme. This scheme is used to understand the elements of the system and its relations. Tables used to model the ontology are described below.

4.2.1. Basic information tables

Basic information tables are used to store the environment elements. Four elements have been considered: physical rooms (real locations sensorially perceived by the robot), conceptual rooms (the type of room that the robot can recognize), physical objects (objects perceived by sensors), and conceptual objects (each semantic information of the object that the robot must recognize).

- **ConceptualRoom**: This table stores the information of the rooms understood as concepts. The table records are those corresponding to the concepts *bathroom*, *bed room*, *kitchen*, etc.
- **ConceptualObject**: This table contains data related to objects abstractly defined and understood as concepts. For example, *Oven*, *Computer*, *Washbasin*, etc.

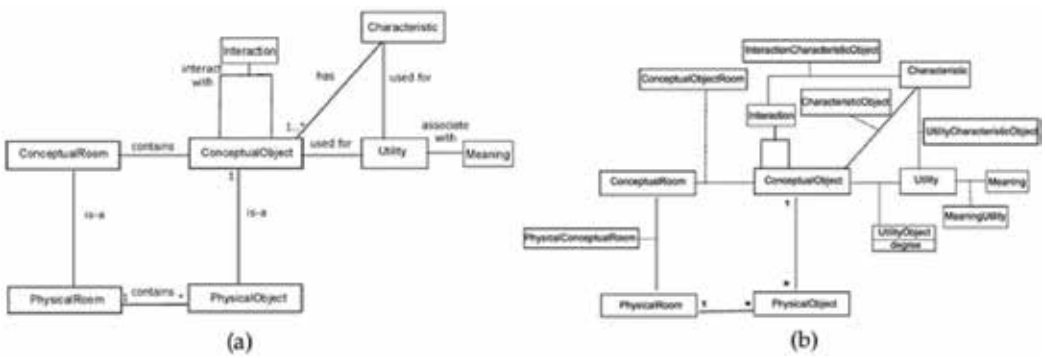


Figure 7. Ontological model for semantic navigation and its implementation on a database. (a) Ontological model for semantic navigation and (b) Database design for semantic navigation.

http://dx.doi.org/10.5772/intechopen.79842

- **PhysicalRoom**: It stores specific locations, places in space, regions delimited by walls, having real coordinates, etc., where the robot can move.
- **PhysicalObject**: This table stores each real object that the robot sensor identifies. This table contains specific objects: *Table-1*, *Table-2*, *Computer-1*, etc.

With these tables, objects and places have been modeled. More tables are needed in order to complete the relations and information of the environment that are described below.

4.2.2. Links between primary tables

To complete the model, the following links between the primary tables are needed:

Objects are always in rooms; so, between *PhysicalRoom* and *PhysicalObject* tables, there exists a link. And a physical room may have an indeterminate amount of physical objects. In table *PhysicalObject*, the room associated to each object is also stored.

4.2.3. Tables that allow searches by semantic proximity

As an example of the implementation of the tables, **Figure 7b** shows the design of the database. This database contains several tables that manage important information to help the robot to find objects and relations. With the information of this table, if the robot does not locate a specific object, it can deduce with what other objects it is linked to, helping the robot locate the object. To get additional information about objects, some tables are defined:

- **Interaction**: Objects interact with other objects. This table can handle any type of interaction, although tests have been performed with a limited number of them. The interactions taken into account are: *BE_A*, when an object is a subtype of another object; *IS_USED_WITH*, when an object is used with an object; and *IS_INSIDE_OF*, when usually an object is inside another object.
- **Utility**: All objects have one or more utilities. The tendency is to group objects that have the same utility (or something related) in the same places. Also, concepts are created regarding kinds of room depending on what they are used for.
- **Meaning**: The actions or utilities that an object has often are associated with a specific meaning. The goal of the navigation may also be oriented to emotions or places that give a feeling, such as calm or fun. For example, *read* is an action *quiet*.
- **Characteristic**: Objects may have features to best define the concept or even differentiate them from others. For example, on the one hand, water may be cold, warm, or hot. And that implies differences in location. Cold water may be found in the refrigerator or in a fountain, but hot water would come out from the faucet.

These tables include the semantic information of the objects in the environment that can help the robot to locate on it. For example, considering the utility, it can be said that a *computer* is used to *work*. The robot can go to an office if it needs to locate a computer.

This relational database model provides a simple way to define and manage semantic knowledge using simple queries without the need to define rules as it has been described in previous works (Galindo et al. [35], Galindo et al. [36]).

4.3. Semantic information management

Once the environment and its components are modeled, the next step is to manage the information in order to accomplish navigation tasks. Robots need to localize in the environment and calculate the path to the targets defined with semantical information. In the real robot, semantic targets must be transferred to the topological and geometrical levels in order to complete the movement of the robot.

4.3.1. Semantic target obtention

Similar to the previous navigation system, a way to establish how to reach a target is needed. In the proposed semantic system is defined the concept of *Semantic Position* as a contextual information unit related to the robot's position. *Semantic Position* contains attributes that correspond to a room code (the target room in this case) and to an object code (the target object in this case). It is therefore understood that a position is obtained in relation to a semantic reference point, which can be either objects or rooms. The target of the robot is defined as a *Semantic Position*.

Depending on the information available in the database about the requested destination and the type of destination (object or room), several options can be found in order to establish the semantic path:

- If the destination is a known object in a known room, the robot has all the information to reach the *Semantic Target*.
- If the destination is an unknown room, a semantic exploration routine must be used to manage the semantic information to locate a room with similar semantic information of the destination.
- If the destination is an unknown object in a known room, the robot would try to explore it to get an object with similar semantic information in the room to define it as a *Semantic Target*.
- If the destination is an unknown object in an unknown room, all information of the object and room in the database must be used to get an object in a room which matches with the semantical information of the destination.

To get the *Semantic Target,* several consultations to the database must be done. One query is used to obtain the specific object instances that have already been stored, other query to know the physical room where a specific object is found, and a last query to match a room with a specific type of object.

In addition, this system allows the search of destinations by semantic proximity. For example, using the knowledge of **Figure 8a** if the requested objective is to do something *fun*, the answer would be to go to the *computer-1*. This is because the *computer-1* is an object associated with the

http://dx.doi.org/10.5772/intechopen.79842

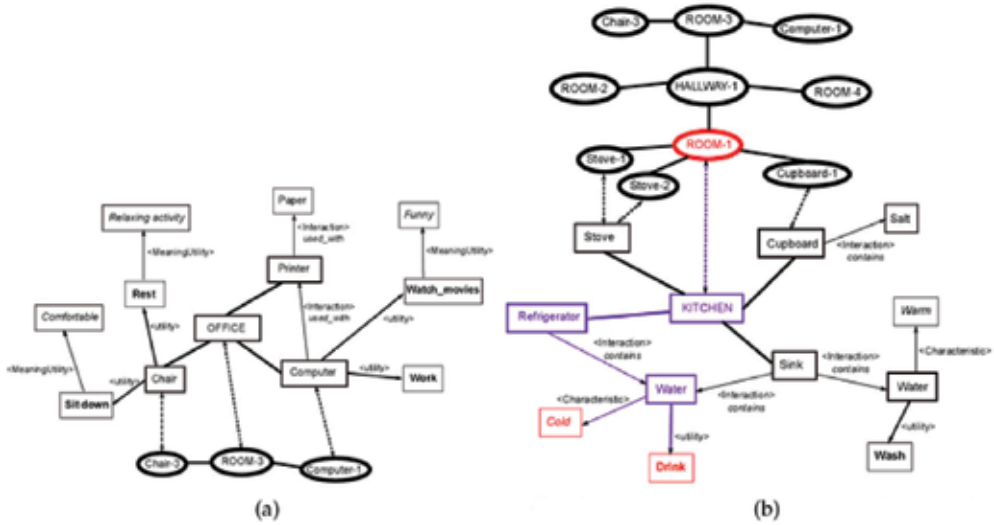


Figure 8. Ontology concepts and semantic relations. (a) Ontology concepts and semantic relations of a office and (b) Ontology concepts and semantic relations of a kitchen.

computer concept. The *computer* concept is associated with the *watching movies* utility and the watching movies utility is associated with the *fun* concept. Using the knowledge of **Figure 8b,** searches for objects with characteristics and actions can be done. If the destination is to *drink something cold*, the system recognizes *cold water* as a valid objective. The *cold water* is related to the *refrigerator*, the *refrigerator* is related to the *kitchen,* and the *kitchen* is related to Room-1. Then, the system goes to Room-1.

4.3.2. Semantic identification of a place

A place, in this case a room, can be identified as a vector of detected objects. The object observed from the environment must be analyzed and with a query we can get a list of the types of room where the object can be found. If an object is found, it defines the room where it is located. For instance, the bed is an object that can only be in a bedroom. If the result of the previous query returns several records, it is because the object is not discriminatory enough to be able to conclude any identification, and more additional semantic information and more queries are needed to get the destination room.

4.4. Example

In this example, the robot is asked for several targets in a home. In the experiment, the robot has the information provided by the previous exploration of the environment. This information is stored in several tables: 1a is the table with physical objects information, 1a is the table with physical objects information and 1b is the table with deducted knowledge.

The objective is to use semantic information to get the semantic target, that is attached to a topological or to a geometrical target to which the robot must move (**Table 1**).

First, the robot is asked for the kitchen. Checking the table *PhysicalConceptualRoom* with a query, the result is *Room-1*. Repeating the query, in the tables, there are no more kitchens in the database, so the process finishes.

In a second experiment, the robot is asked for a chair. In the initial environment, there are three real objects of the CHAIR type. In the query, the robot identifies three chairs: *Chair-1*, *Chair-2*, and *Chair-3*. The robot gets the information of the rooms in which the chairs are and moves to the first option. If it indicates that this is not the correct room, the robot moves to the rest of the options. This process is described in **Figure 9**.

In case the object has no match with the observed physical object, it is necessary to ask again for a chair, after having discarded the chairs from the previous test. The operation sequence is shown in **Figure 10a**. In **Figure 9,** another query on chair is shown, and the robot starts to explore searching in new types of rooms where chairs could be found.

(a) Content of the table physical object			**(b) Content of the table conceptual physical room**	
Physical object			**Conceptual physical room**	
Name	**Conceptual name**	**Room name**	**Conceptual name**	**Physical name**
Chair-1	CHAIR	Room-1	Room-1	KITCHEN
Chair-2	CHAIR	Room-2	Room-2	LIVING ROOM
Chair-3	CHAIR	Room-3	Room-3	OFFICE
Refrigerator-1	REFRIGERATOR	Room-1		
Sofa-1	SOFA	Room-2		
Desk-1	DESK	Room-3		

Table 1. Tables with semantic information.

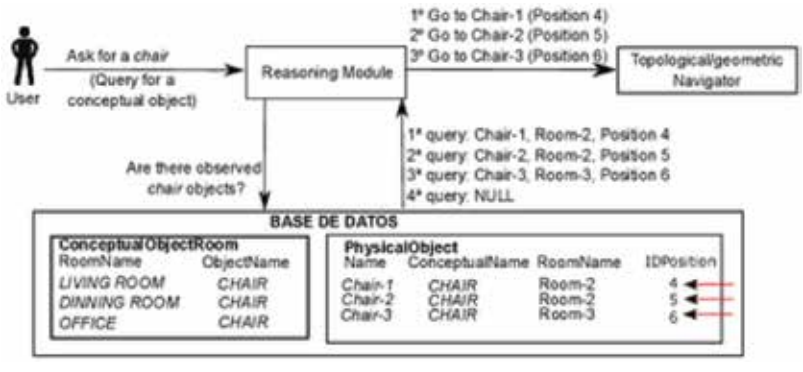
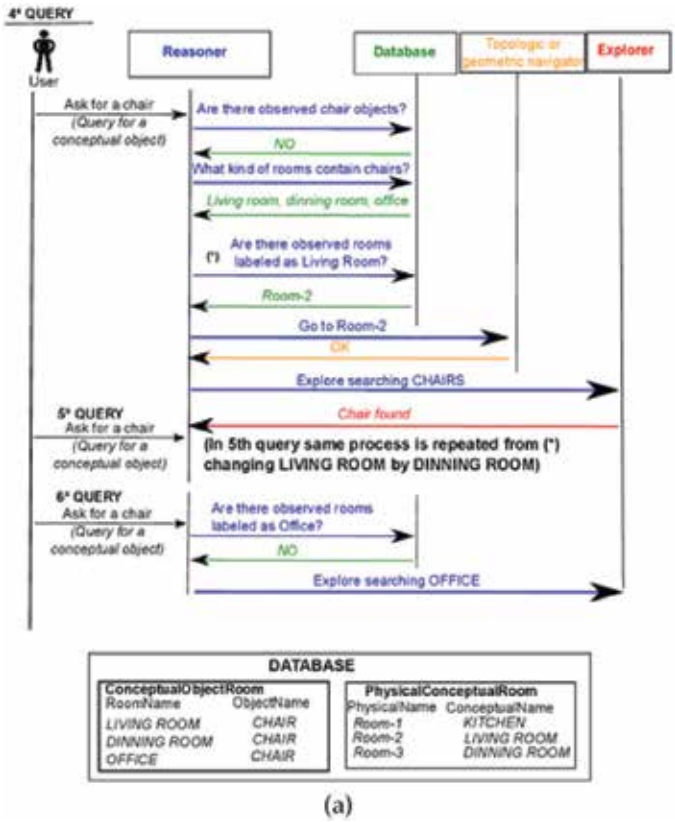


Figure 9. Test sequence with the CHAIR object.

http://dx.doi.org/10.5772/intechopen.79842



ConceptulaObjectRoom	
RoomName	**ObjectName**
KITCHEN	STOVE
KITCHEN	REFRIGERATOR
KITCHEN	SINK
KITCHEN	WASHING MACHINE
KITCHEN	HEATER
LIVING ROOM	SOFA
LIVING ROOM	COFFEE TABLE
LIVING ROOM	CHAIR
LIVING ROOM	TV SET
LIVING ROOM	HEATER
LIVING ROOM	BOOKCASE
DINING ROOM	CHAIR
DINING ROOM	DINING TABLE
DINING ROOM	HEATER
OFFICE	HEATER
OFFICE	CHAIR
OFFICE	BOOKCASE
OFFICE	DESK
OFFICE	COMPUTER
BATHROOM	WASHBASIN
BATHROOM	TOILET
BATHROOM	HEATER

(b)

Figure 10. Ontology concepts and semantic relations. (a) Test sequence with the CHAIR object when there is no match with no observed physical object and (b) Content of the table ConceptualObjectRoom.

5. Perception model for navigation

Talking about autonomous robots implies carrying out movements safely and having a complete knowledge of the environment. These elements define the capabilities of action and interaction between the environment, humans, and robots. Tasks performed by mobile robots such as navigation, localization, planning, among others, can be improved if the perceptual information is considered. So, the general idea is detecting and identifying meaningful elements (objects and scenes) of the environment. There are different ways to obtain the information about the environment. One of them consists of detecting recognizable features of the environment (natural landmarks) using several types of sensors. The detection of artificial landmarks, based on the work of Fischler & Elschlager [37], can be used to acquire a representation of the environment. However, due to the technological advances in 3D sensors (e.g., RGB-D sensors) and according to Huang [38], vision has become the main sensor used for navigation, localization, and scene recognition. Vision provides significant information about

the objects present in a place; at the same time, it is capable of providing semantic information of the scene where it is. Scene recognition is a very well-known challenging issue that deals with how robots understand scenes just like a human does and the appearance variability of real environments. Therefore, regardless of the type of navigation used, whether geometrical, topological, or semantic, place recognition and object identification play a significant role in terms of representation of the environment.

5.1. Object recognition

To perform several tasks in common indoor environments, mobile robots need to quickly and accurately verify and recognize objects, obstacles, etc. One of these crucial tasks is to move safely in an unknown environment. Autonomous robots should be able to acquire and hold visual representations of their environments.

The most important stages in an object recognition model are: feature extraction and prediction. As for feature extraction, the identification of significant aspects of different objects that belong to the same class, independently of the appearance variabilities, such a scaling, rotation, translation, illumination changing, among others, is crucial to obtain a suitable representation of the objects present in a scene. Some techniques are based on local and global descriptors such as the works presented by Bay et al. [39] and Hernández et al. [40], or a combination of both of them (Hernandez-Lopez et al. [41]). Also, in other approaches such as the work presented by Csurka et al. [42], visual vocabularies (e. g., bag of words) are commonly used to create a proper representation of each object.

Regarding prediction, through the vectors created from the extracted features, it is possible to learn these characteristics in order to identify objects that correspond with each class. In the literature, different classification techniques based on machine learning such as nearest neighbor classifier, neural networks, AdaBoost, etc., have been proposed depending on the kind of extracted features (Pontil & Verri [43]). Machine learning is a field of computer science that includes algorithms that improve their performance at a given task considering the experience. In this way, support vector machine (SVM) is one of the most helpful classification algorithms. The aim of SVM is to generate a training model that is capable of predicting the target classes of the test dataset, considering only its attributes.

Generally, an object recognition system that works in real time is divided into two stages: offline and online. Offline stage includes all the processes to reduce the execution time and guarantee the efficiency of the system, which are image preprocessing, segmentation, feature extraction, and training process. Online stage refers to the processes carried out in real time with the goal to ensure the interaction between the robot and the environment. Mainly, the processes included in the online stage are image retrieval and classification.

Object detection can be very useful for a navigation system since it allows the robot to relate what it perceives to the scenes in which it is. For this reason, it is necessary to consider that the designed systems and the sensors are not perfect. Therefore, the object detection model has to incorporate uncertainty information. The uncertainty management is a relevant aspect in an object recognition system because it allows to represent the environment and its elements in a

http://dx.doi.org/10.5772/intechopen.79842

more realistic way. The uncertainty calculation can be determined considering the dependency relations between different factors. Some of the factors are: the accuracy of the model that can be determined empirically and a factor based on the outcomes of the classification algorithm. Also, other factors can include the influence of other elements of the environment, for example the distance during the detection process. Finally, considering these factors makes it possible to obtain a robust object recognition model to serve as an input to a mobile robot.

5.2. Place recognition

The growth of service robotics in recent years has created the needed for developing models that contribute to robots being able to adequately handle information from human environments. A semantic model can improve the high-level tasks of a robot, such as, semantic navigation and human-robot and robot-environment interaction. According to Wang et al. [44], semantic navigation is considered as a system that takes into account semantic information to represent the environment and then to carry out the localization and navigation of the robot.

In view of the above, place recognition deals with the process of recognizing an area of the environment in which there are elements (objects), actions are developed, and robot-environment and human-robot interaction is possible. The scene-understanding issue can be defined as a combination between scene recognition and object detection. There are approaches that try to solve the scene recognition problem through computer vision algorithms, including the creation of complex feature descriptors (Xie et al. [45]), and a combination of feature extraction techniques (Nicosevici & Garcia [46] and Khan et al. [47]), among others. Moreover, if it is desired to get a robust model as close as possible to reality, the incorporation of environmental data and errors of the sensors is needed.

Some approaches are based on machine learning and select support vector machine as classification technique. Commonly, this type of place recognition model is composed by the following processes (**Figure 11**): image preprocessing, that includes the selection of the datasets and the initial preprocessing of the images to obtain the training model; feature extraction, that can implement some techniques such as bag of words, local and global descriptors, among others; training process, where the parameters of the classifier are defined and the model of the scene is generated; prediction process that generates the final results of the possible scene where the robot is located; and ,finally, a last process called reclassification has to be considered in which it is possible to generate a relationship between the place recognition model and the elements (objects) of the environment.

In this way, the influence of the objects in the scene can improve, worsen, and correct the final results on where the robot is located. The process implies the adjustment of the probabilities of being in a place and therefore the management of uncertainties. To do that, it is necessary that the place recognition model and the object recognition system work simultaneously in real time. The uncertainty can be modeled from a set of rules based on learning to determine the probability of co-occurrence of the objects. Also, it is important to incorporate the prior information of the place and object recognition models. Finally, it is possible to apply different

Figure 11. Schematic of a general scene recognition model. Vision data are the initial input of the recognition model. Then, the metric data and object information can be included into the model to update the final outcome of the possible place where the robot can be.

theorems such as Bayes to determine the final relation between objects and scenes, and the last result about where the robot is. The adjusted prediction has to be available as input for relevant tasks such as localization, navigation, scene understanding, and human-robot and robot-environment interaction. All this contributes to the main goal that is to have an autonomous and independent robot, with a wide knowledge of the environment.

6. Conclusion

The aim of this chapter was to describe different approaches for global navigation systems for mobile robots applied to indoor environments. Many researches and current developments are focused on solving specific needs for navigation. Our objective is to merge all these developments in order to classify them and establish a global frame for navigation.

Robot navigation has been tackled from different perspectives leading to a classification into three main approaches: geometric navigation, topological navigation, and semantic navigation. Although the three of them differ in their definition and methods, all of them have the focus on driving a robot autonomously and safely.

In this chapter, different trends and techniques have been presented, all of them inspired by biological models and pursuing human abilities and abstraction models. The geometric representation, closer to the sensor and actuator world, is the best one to perform local navigation and precise path-planning. Topological representation of the environment, which is based on graphs, enables large navigation tasks and uses similar models as humans do. The semantic representation, which is the closest to cognitive human models, adds concepts such as utilities or meanings of the environment elements and establishes complex relations between them. All of these representations are based on the available information of the environment. For this reason, perception plays a significant role in terms of understanding and moving through the environment.

Despite the differences between the environment representations, we consider that the integration of all of them and the proper management of the information is the key to achieve a global navigation system.

Author details

Ramón Barber*, Jonathan Crespo, Clara Gómez, Alejandra C. Hernámdez and Marina Galli

*Address all correspondence to: rbarber@ing.uc3m.es

University Carlos III of Madrid, Spain

References

[1] Levitt TS. Qualitative navigation for mobile robots. International Journal of Artificial Intelligence. 1990;**44**(3):305-360

[2] Bailey T, Durrant-Whyte H. Simultaneous localization and mapping (slam): Part II. IEEE Robotics and Automation Magazine. 2006;**13**(3):108-117

[3] Wang C, Meng L, She S, Mitchell IM, Li T, Tung F, et al. Autonomous mobile robot navigation in uneven and unstructured indoor environments. In: Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE; 2017. pp. 109-116

[4] Pfrunder A, Borges PV, Romero AR, Catt G, Elfes A. Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3d lidar. In: Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE; 2017. pp. 2601-2608

[5] Fernández-Madrigal J-A, Galindo C, González J. Assistive navigation of a robotic wheelchair using a multihierarchical model of the environment. Integrated Computer-Aided Engineering. 2004;**11**(4):309-322

[6] Ko DW, Kim YN, Lee JH, Suh IH. A scene-based dependable indoor navigation system. In: Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. IEEE; 2016. pp. 1530-1537

[7] Kuipers BJ, Levitt TS. Navigation and mapping in large scale space. AI Magazine. 1988; **9**(2):25

[8] Giralt G, Chatila R, Vaisset M. An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots, Autonomous Robot Vehicles. New York: Springer; 1990. pp. 420-443

[9] Mataric MJ. A distributed model for mobile robot environment-learning and navigation. Massachusetts Institute of Technology Cambridge. MA, USA; 1990

[10] LaValle SM. Planning Algorithms. Cambridge University Press; 2006

[11] De Berg M, Van Kreveld M, Overmars M, Schwarzkopf OC. Computational Geometry. Berlin Heidelberg: Springer; 2000. pp. 1-17

[12] Hwang YK, Ahuja N. A potential field approach to path planning. IEEE Transactions on Robotics and Automation. 1992;**8**(1):23-32

[13] Poty A, Melchior P, Oustaloup A. Dynamic path planning for mobile robots using fractional potential field. In: Control, Communications and Signal Processing, 2004. First International Symposium on. IEEE; 2004. pp. 557-561

[14] Lee D-T, Drysdale RL III. Generalization of voronoi diagrams in the plane. SIAM Journal on Computing. 1981;**10**(1):73-87

[15] Breu H, Gil J, Kirkpatrick D, Werman M. Linear time euclidean distance transform algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1995;**17**(5):529-533

[16] Alt H, Cheong O, Vigneron A. The voronoi diagram of curved objects. Discrete & Computational Geometry. 2005;**34**(3):439-453

[17] Gang L, Wang J. Prm path planning optimization algorithm research. (n.d.). Wseas Transactions on Systems and Control. 2016;**11**. E-ISSN: 2224-2856

[18] Kuffner JJ, LaValle SM. Rrt-connect: An efficient approach to single-query path planning. In: Robotics and Automation, 2000. Proceedings of ICRA'00. IEEE International Conference on, Vol. 2. IEEE; 2000. pp. 995-1001

[19] LaValle SM. Rapidly-exploring random trees: A new tool for path planning. 1998

[20] Valero-Gomez A, Gomez JV, Garrido S, Moreno L. The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories. IEEE Robotics and Automation Magazine. 2013;**20**(4):111-120

[21] Thrun S, Bücken A. Integrating grid-based and topological maps for mobile robot navigation. In: Proceedings of the National Conference on Artificial Intelligence. 1996. pp. 944-951

[22] Vale A, Ribeiro M. Environment mapping as a topological representation. In: Proceedings of the 11th International Conference on Advanced Robotics-ICAR. 2003. pp. 1-3

[23] Kuipers BJ, Byun YT. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Robotics and Autonomous Systems. 1991;**8**(1–2):47-63

[24] Thrun S. Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence. 1998;**99**(1):21-71

[25] Simhon S, Dudek G. A global topological map formed by local metric maps. In: IEEE/RSJ International Conference on Intelligent Robotic Systems. 1998. pp. 1708-1714

[26] Amigoni F, Gallo A. A multi-objective exploration strategy for mobile robots. In: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE; 2005. pp. 3850-3855

[27] Arvanitakis I, Giannousakis K, Tzes A. Mobile robot navigation in unknown environment based on exploration principles. In: Control Applications (CCA), 2016 IEEE Conference on. IEEE; 2016. pp. 493-498

[28] Mataric MJ. Behaviour-based control: Examples from navigation, learning, and group behaviour. Journal of Experimental & Theoretical Artificial Intelligence. 1997;**11**:323-336

[29] Skiena S. Dijkstra's Algorithm. Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. MA: Addison-Wesley, Reading; 1990. pp. 225-227

[30] Egido V, Barber R, Boada MJL, Salichs MA. A planner for topological navigation based on previous experiences. IFAC Proceedings Volumes; 2004;**37**(8):615-620. ISSN 1474-6670. https://doi.org/10.1016/S1474-6670(17)32046-3

[31] Vasudevan S, Siegwart R. Bayesian space conceptualization and place classification for semantic maps in mobile robotics. Robotics and Autonomous Systems (RAS). 2008;**56**: 522-537

[32] Vasudevan S, Siegwart R. A bayesian conceptualization of space for mobile robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2007

[33] Duda RO, Hart PE, Stork DG. Pattern Classification. New York, NY, USA: Wiley Interscience; 2000

[34] Kollar T, Roy N. Utilizing object-object and object-scene context when planning to find things. In: IEEE International Conference on Robotics and Automation (ICRA). 2009

[35] Galindo C, Fernï£¡ndez-Madrigal J-A, Gonzï£¡lez J, Saffiotti A. Robot task planning using semantic maps, Robotics and Autonomous Systems. 2008;**56**(11):955-966

[36] Galindo C, Saffiotti A, Coradeschi S, Buschka P, Fernï£¡ndez-Madrigal JA, Gonzï£¡lez J. Multi-hierarchical semantic maps for mobile robotics. In: Proceedings of IROS. 2005

[37] Fischler MA, Elschlager RA. The representation and matching of pictorial structures. IEEE Transactions on Computers. 1973;**100**(1):67-92

[38] Huang T. Computer vision: Evolution and promise. In: International conference; 5th, High technology: Imaging science and technology. Chiba, Japan; 1996. pp.13-20

[39] Bay H, Tuytelaars T, Van Gool L. Surf: Speeded up robust features. In: European Conference on Computer Vision. Springer; 2006. pp. 404-417

[40] Hernández AC, Gómez C, Crespo J, Barber R. Object detection applied to indoor environments for mobile robot navigation. Sensors. 2016;**16**(8):1180

[41] Hernandez-Lopez J-J, Quintanilla-Olvera A-L, López-Ramírez J-L, Rangel-Butanda F-J, Ibarra-Manzano M-A, Almanza-Ojeda D-L. Detecting objects using color and depth segmentation with kinect sensor. Procedia Technology. 2012;**3**:196-204

[42] Csurka G, Dance C, Fan L, Willamowski J, Bray C. Visual categorization with bags of keypoints. In: Workshop on statistical learning in computer vision, ECCV, Vol. 1; Prague. 2004. pp. 1-2

[43] Pontil M, Verri A. Support vector machines for 3d object recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1998;**20**(6):637-646

[44] Wang L, Zhao L, Huo G, Li R, Hou Z, Luo P, et al. Visual semantic navigation based on deep learning for indoor mobile robots. Complexity. 2018;**2018**. Article ID: 1627185

[45] Xie L, Tian Q, Wang M, Zhang B. Spatial pooling of heterogeneous features for image classification. IEEE Transactions on Image Processing. 2014;**23**(5):1994-2008

[46] Nicosevici T, Garcia R. Automatic visual bag-of-words for online robot navigation and mapping. IEEE Transactions on Robotics. 2012;**28**(4):886-898

[47] Khan SH, Bennamoun M, Sohel F, Togneri R. Geometry driven semantic labeling of indoor scenes. In: European Conference on Computer Vision. Springer; 2014. pp. 679-694

Chapter 6

Intelligent Robotic Perception Systems

Cristiano Premebida, Rares Ambrus and
Zoltan-Csaba Marton

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79742

Abstract

Robotic perception is related to many applications in robotics where sensory data and artificial intelligence/machine learning (AI/ML) techniques are involved. Examples of such applications are object detection, environment representation, scene understanding, human/pedestrian detection, activity recognition, semantic place classification, object modeling, among others. Robotic perception, in the scope of this chapter, encompasses the ML algorithms and techniques that empower robots to learn from sensory data and, based on learned models, to react and take decisions accordingly. The recent developments in machine learning, namely deep-learning approaches, are evident and, consequently, robotic perception systems are evolving in a way that new applications and tasks are becoming a reality. Recent advances in human-robot interaction, complex robotic tasks, intelligent reasoning, and decision-making are, at some extent, the results of the notorious evolution and success of ML algorithms. This chapter will cover recent and emerging topics and use-cases related to intelligent perception systems in robotics.

Keywords: robotic perception, machine learning, advanced robotics, artificial intelligence

1. Introduction

In robotics, perception is understood as a system that endows the robot with the ability to perceive, comprehend, and reason about the surrounding environment. The key components of a perception system are essentially sensory data processing, data representation (environment modeling), and ML-based algorithms, as illustrated in **Figure 1**. Since *strong AI* is still far from being achieved in real-world robotics applications, this chapter is about *weak AI*, i.e., standard machine learning approaches [1].

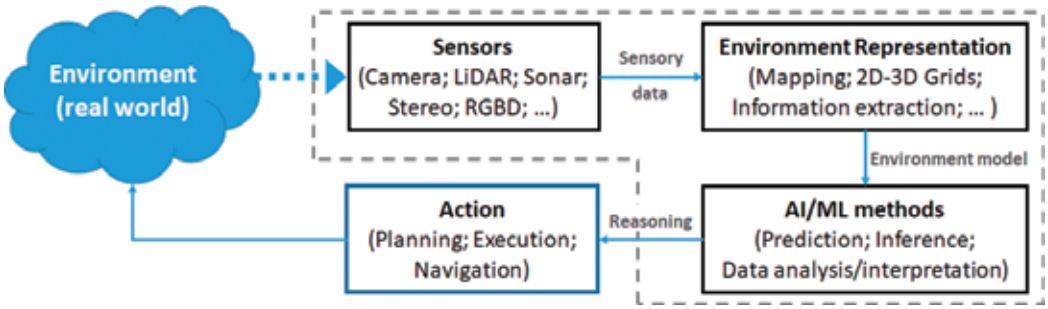


Figure 1. Key modules of a typical robotic perception system: sensory data processing (focusing here on visual and range perception); data representations specific for the tasks at hand; algorithms for data analysis and interpretation (using AI/ML methods); and planning and execution of actions for robot-environment interaction.

Robotic perception is crucial for a robot to make decisions, plan, and operate in real-world environments, by means of numerous functionalities and operations from occupancy grid mapping to object detection. Some examples of robotic perception subareas, including autonomous robot-vehicles, are obstacle detection [2, 3], object recognition [4, 5], semantic place classification [6, 7], 3D environment representation [8], gesture and voice recognition [9], activity classification [10], terrain classification [11], road detection [12], vehicle detection [13], pedestrian detection [14], object tracking [3], human detection [15], and environment change detection [16].

Nowadays, most of robotic perception systems use machine learning (ML) techniques, ranging from classical to deep-learning approaches [17]. Machine learning for robotic perception can be in the form of unsupervised learning, or supervised classifiers using handcrafted features, or deep-learning neural networks (e.g., convolutional neural network (CNN)), or even a combination of multiple methods.

Regardless of the ML approach considered, data from sensor(s) are the key ingredient in robotic perception. Data can come from a single or multiple sensors, usually mounted onboard the robot, but can also come from the infrastructure or from another robot (e.g., cameras mounted on UAVs flying nearby). In multiple-sensors perception, either the same modality or multimodal, an efficient approach is usually necessary to combine and process data from the sensors before an ML method can be employed. Data alignment and calibration steps are necessary depending on the nature of the problem and the type of sensors used.

Sensor-based environment representation/mapping is a very important part of a robotic perception system. Mapping here encompasses both the acquisition of a metric model and its semantic interpretation, and is therefore a synonym of environment/scene representation. This semantic mapping process uses ML at various levels, e.g., reasoning on volumetric occupancy and occlusions, or identifying, describing, and matching optimally the local regions from different time-stamps/models, i.e., not only higher level interpretations. However, in the majority of applications, the primary role of environment mapping is to model data from exteroceptive sensors, mounted onboard the robot, in order to enable reasoning and inference regarding the real-world environment where the robot operates.

Robot perception functions, like localization and navigation, are dependent on the environment where the robot operates. Essentially, a robot is designed to operate in two categories of

http://dx.doi.org/10.5772/intechopen.79742

environments: indoors or outdoors. Therefore, different assumptions can be incorporated in the mapping (representation) and perception systems considering indoor or outdoor environments. Moreover, the sensors used are different depending on the environment, and therefore, the sensory data to be processed by a perception system will not be the same for indoors and outdoors scenarios. An example to clarify the differences and challenges between a mobile robot navigating in an indoor versus outdoor environment is the ground, or terrain, where the robot operates. Most of indoor robots assume that the ground is regular and flat which, in some manner, facilitates the environment representation models; on the other hand, for field (outdoors) robots, the terrain is quite often far from being regular and, as consequence, the environment modeling is itself a challenge and, without a proper representation, the subsequent perception tasks are negatively affected. Moreover, in outdoors, robotic perception has to deal with weather conditions and variations in light intensities and spectra.

Similar scenario-specific differences exist in virtually all use-cases of robotic vision, as exemplified by the 2016 Amazon Picking Challenge participants' survey [18], requiring complex yet robust solutions, and therefore considered one of the most difficult tasks in the pick-and-place application domain. Moreover, one of the participating teams from 2016 benchmarked a pose estimation method on a warehouse logistics dataset, and found large variations in performance depending on clutter level and object type [2]. Thus, perception systems currently require expert knowledge in order to select, adapt, extend, and fine-tune the various employed components.

Apart from the increased training data sizes and robustness, the end-to-end training aspect of deep-learning (DL) approaches made the development of perception systems easier and more accessible for newcomers, as one can obtain the desired results directly from raw data in many cases, by providing a large number of training examples. The method selection often boils down to obtaining the latest pretrained network from an online repository and fine-tuning it to the problem at hand, hiding all the traditional feature detection, description, filtering, matching, optimization steps behind a relatively unified framework. Unfortunately, at the moment an off-the-shelf DL solution for every problem does not exist, or at least no usable pretrained network, making the need for huge amounts of training data apparent. Therefore, large datasets are a valuable asset for modern AI/ML. A large number of datasets exist for perception tasks as well, with a survey of RGB-D datasets presented by Firman [5] (up-to-date list available online: http://www.michaelfirman.co.uk/RGBDdatasets/), and even tools for synthetically generating sensor-based datasets, e.g., the work presented by Handa et al. [4] which is available online: http://robotvault.bitbucket.org/. However, the danger is to overfit to such benchmarks, as the deployment environment of mobile robots is almost sure to differ from the one used in teaching the robot to perceive and understand the surrounding environment. Thus, the suggestions formulated by Wagstaff [19] still hold true today and should be taken to heart by researchers and practitioners.

As pointed out recently by Sünderhauf et al. [17], robotic perception (also designated robotic vision in [17]) differs from traditional computer vision perception in the sense that, in robotics, the outputs of a perception system will result in decisions and actions in the real world. Therefore, perception is a very important part of a complex, embodied, active, and goal-driven robotic system. As exemplified by Sünderhauf et al. [17], robotic perception has to translate images (or scans, or point-clouds) into actions, whereas most computer vision applications take images and translate the outputs into information.

2. Environment representation

Among the numerous approaches used in environment representation for mobile robotics, and for autonomous robotic-vehicles, the most influential approach is the occupancy grid mapping [20]. This 2D mapping is still used in many mobile platforms due to its efficiency, probabilistic framework, and fast implementation. Although many approaches use 2D-based representations to model the real world, presently 2.5D and 3D representation models are becoming more common. The main reasons for using higher dimensional representations are essentially twofold: (1) robots are demanded to navigate and make decisions in higher complex environments where 2D representations are insufficient; (2) current 3D sensor technologies are affordable and reliable, and therefore 3D environment representations became attainable. Moreover, the recent advances in software tools, like ROS and PCL, and also the advent of methods like Octomaps, developed by Hornung et al. [21], have been contributing to the increase in 3D-like environment representations.

The advent and proliferation of RGBD sensors has enabled the construction of larger and evermore detailed 3D maps. In addition, considerable effort has been made in the semantic labeling of these maps, at pixel and voxels levels. Most of the relevant approaches can be split into two main trends: methods designed for online and those designed for offline use. Online methods process data as it is being acquired by the mobile robot, and generate a semantic map incrementally. These methods are usually coupled with a SLAM framework, which ensures the geometric consistency of the map. Building maps of the environment is a crucial part of any robotic system and arguably one of the most researched areas in robotics. Early work coupled mapping with localization as part of the simultaneous localization and mapping (SLAM) problem [22, 23]. More recent work has focused on dealing with or incorporating time-dependencies (short or long term) into the underlying structure, using either grid maps as described in [8, 24], pose-graph representations in [25], and normal distribution transform (NDT) [16, 26].

As presented by Hermans et al. [27], RGBD data are processed by a random forest-based classifier and predict semantic labels; these labels are further regularized through the conditional random field (CRF) method proposed by Krahenbuhl and Koltun [28]. Similarly, McCormac et al. [29] use the elastic fusion SLAM algorithm proposed by Whelan et al. [30] to fuse CNN predictions about the scene in a geometrically consistent map. In the work of Sünderhauf et al. [6], a CNN is used to incrementally build a semantic map, with the aim of extending the number of classes supported by the CNN by complementing it with a series of one-vs-all classifiers which can be trained online. A number of semantic mapping approaches are designed to operate offline, taking as input a complete map of the environment. In the methods described by Ambrus et al. [31, 32] and Armeni et al. [33], large-scale point clouds of indoor buildings are processed, and then, after segmenting the input data, the method's outputs are in the form of a set of "rooms." Ambrus et al. [31, 32] use a 2D cell-complex graph-cut approach to compute the segmentation with the main limitation that only single floor buildings can be processed, while Armeni et al. [33] process multifloor structures by detecting the spaces between the walls, ceilings, etc., with the limitation that the building walls have to be axis-aligned (i.e., the Manhattan world assumption). Similarly, in the work proposed by Mura et al. [34], a large point cloud of an indoor structure is processed by making use of a 3D cell-complex structure

http://dx.doi.org/10.5772/intechopen.79742

and outputting a mesh containing the semantic segmentation of the input data. However, the main limitation in [34] is that the approach requires knowledge of the positions from which the environment was scanned when the input data were collected.

The recent work presented by Brucker et al. [7] builds on the segmentation of Ambrus et al. [31, 32] and explores ways of fusing different types of information, such as presence of objects and cues of the types of rooms to obtain a semantic segmentation of the environment. The aim of the work presented by Brucker et al. [7] is to obtain an intuitive and human-like labeling of the environment while at the same time preserving as many of the semantic features as possible. Also, Brucker et al. [7] use a conditional random field (CRF) or the fusion of various heterogeneous data sources and inference is done using Gibbs sampling technique.

Processing sensory data and storing it in a representation of the environment (i.e., a map of the environment) has been and continues to be an active area in robotics research, including autonomous driving system (or autonomous robotic-vehicles). The approaches covered range from metric representations (2D or 3D) to higher semantic or topological maps, and all serve specific purposes key to the successful operation of a mobile robot, such as localization, navigation, object detection, manipulation, etc. Moreover, the ability to construct a geometrically accurate map further annotated with semantic information also can be used in other applications such as building management or architecture, or can be further fed back into a robotic system, increasing the awareness of its surroundings and thus improving its ability to perform certain tasks in human-populated environments (e.g., finding a cup is more likely to be successful if the robot knows a priori which room is the kitchen and how to get there).

3. Artificial intelligence and machine learning applied on robotics perception

Once a robot is (self) localized, it can proceed with the execution of its task. In the case of autonomous mobile manipulators, this involves localizing the objects of interest in the operating environment and grasping them. In a typical setup, the robot navigates to the region of interest, observes the current scene to build a 3D map for collision-free grasp planning and for localizing target objects. The target could be a table or container where something has to be put down, or an object to be picked up. Especially in the latter case, estimating all 6 degrees of freedom of an object is necessary. Subsequently, a motion and a grasp are computed and executed. There are cases where a tighter integration of perception and manipulation is required, e.g., for high-precision manipulation, where approaches like visual servoing are employed. However, in every application, there is a potential improvement for treating perception and manipulation together.

Perception and manipulation are complementary ways to understand and interact with the environment and according to the common coding theory, as developed and presented by Sperry [35], they are also inextricably linked in the brain. The importance of a tight link between perception and action for artificial agents has been recognized by Turing [36], who suggested to equip computers “with the best sense organs that money can buy” and let them learn from gathered experiences until they pass his famous test as described in [37].

The argument for embodied learning and grounding of new information evolved, considering the works of Steels and Brooks [38] and Vernon [39], and more recently in [40], robot perception involves planning and interactive segmentation. In this regard, perception and action reciprocally inform each other, in order to obtain the best results for locating objects. In this context, the localization problem involves segmenting objects, but also knowing their position and orientation relative to the robot in order to facilitate manipulation. The problem of object pose estimation, an important prerequisite for model-based robotic grasping, uses in most of the cases precomputed grasp points as described by Ferrari and Canny [41]. We can categorize this topic in either template/descriptor-based approaches or alternatively local feature/patch-based approaches. In both cases, an ever-recurring approach is that bottom-up data-driven hypothesis generation is followed and verified by top-down concept-driven models. Such mechanisms are assumed, as addressed by Frisby and Stone [42], to be like our human vision system.

The approaches presented in ([43–45] make use of color histograms, color gradients, depth or normal orientations from discrete object views, i.e., they are examples of vision-/camera-based perception for robots. Vision-based perception systems typically suffer from occlusions, aspect ratio influence, and from problems arising due to the discretization of the 3D or 6D search space. Conversely, in the works of [46–48], they predict the object pose through voting or a PnP algorithm [49]. The performance usually decreases if the considered object lacks texture and if the background is heavily cluttered. In the works listed above, learning algorithms based on classical ML methods and deep-learning (e.g., CNN) have been employed.

The importance of mobile manipulation and perception areas has been signaled by the (not only academic) interest spurred by events like the Amazon Robotics (formerly Picking) Challenge and the workshop series at the recent major computer vision conferences associated with the SIXD Challenge (http://cmp.felk.cvut.cz/sixd/workshop_2018/). However, current solutions are either heavily tailored to a specific application, requiring specific engineering during deployment, or their generality makes them too slow or imprecise to fulfill the tight time-constraints of industrial applications. While deep learning holds the potential to both improve accuracy (i.e., classification or recognition performance) and also to increase execution speed, more work on transfer learning, in the sense of generalization improvement, is required to apply models learned in real-world and also in unseen (new) environment. Domain adaptation and domain randomization (i.e., image augmentations) seem to be important directions to pursue, and should be explored not only for vision/camera cases, but also for LiDAR-based perception cases.

Usually, in traditional mobile robot manipulation use-cases, the navigation and manipulation capabilities of a robot can be exploited to let the robot gather data about objects autonomously. This can involve, for instance, observing an object of interest from multiple viewpoints in order to allow a better object model estimation, or even in-hand modeling. In the case of perception for mobile robots and autonomous (robot) vehicles, such options are not available; thus, its perception systems have to be trained offline. However, besides AI/ML-based algorithms and higher level perception, for autonomous driving applications, environment representation (including multisensor fusion) is of primary concern [50, 51].

The development of advanced perception for (full) autonomous driving has been a subject of interest since the 1980s, having a period of strong development due to the DARPA

http://dx.doi.org/10.5772/intechopen.79742

Challenges (2004, 2005, and 2007) and the European ELROB challenges (since 2006), and more recently, it has regained considerable interest from automotive and robotics industries and academia. Research in self-driving cars, also referred as autonomous robot-cars, is closely related to mobile robotics and many important works in this field have been published in well-known conferences and journals devoted to robotics. Autonomous driving systems (ADS) comprise, basically, perception (including sensor-fusion and environment modeling/representation), localization, and navigation (path planning, trajectory following, control) and, more recently, cooperation (V2X-based communication technologies). However, the cornerstone of ADS is the perception system because it is involved in most of the essential and necessary tasks for safe driving such as the "segmentation," detection/recognition, of: road, lane-markings, pedestrians, and other vulnerable road users (e.g., cyclists), other vehicles, traffic signals, crosswalks, and the numerous other types of objects and obstacles that can be found on the roads. In addition to the sensors (e.g., cameras, LIDAR, Radar, "new" solid-state LiDAR technology) and the models used in ADS, the common denominator in a perception system consists of AI/ML algorithms, where deep learning is the leading technique for semantic segmentation and object detection [50].

One of current trends in autonomous vehicles and robotics is the promising idea of incorporating cooperative information, from connected environment/infrastructure, into the decision loop of the robotic perception system. The rationale is to improve robustness and safety by providing complementary information to the perception system, for example: the position and identification of a given object or obstacle on the road could be reported (e.g., broadcasted through a communication network) in advance to an autonomous car, moments before the object/obstacle are within the onboard sensor's field/range of view.

4. Case studies

4.1. The Strands project

The EU FP7 Strands project [52] is formed by a consortium of six universities and two industrial partners. The aim of the project is to develop the next generation of intelligent mobile robots, capable of operating alongside humans for extended periods of time. While research into mobile robotic technology has been very active over the last few decades, robotic systems that can operate robustly, for extended periods of time, in human-populated environments remain a rarity. Strands aims to fill this gap and to provide robots that are intelligent, robust, and can provide useful functions in real-world security and care scenarios. Importantly, the extended operation times imply that the robotic systems developed have to be able to cope with an ever-increasing amount of data, as well as to be able to deal with the complex and unstructured real world (**Figure 2**).

Figure 3 shows a high level overview of the Strands system (with more details in [52]): the mobile robot navigates autonomously between a number of predefined waypoints. A task scheduling mechanism dictates when the robot should visit which waypoints, depending on the tasks the robot has to accomplish on any given day. The perception system consists, at the lowest level, of a module which builds local metric maps at the waypoints visited by the robot.

Figure 2. The Strands project (image from http://strands.acin.tuwien.ac.at/).

These local maps are updated over time, as the robot revisits the same locations in the environment, and they are further used to segment out the dynamic objects from the static scene. The dynamic segmentations are used as cues for higher level behaviors, such as triggering a data acquisition and object modeling step, whereby the robot navigates around the detected object to collect additional data which are fused into a canonical model of the object [53]. The data can

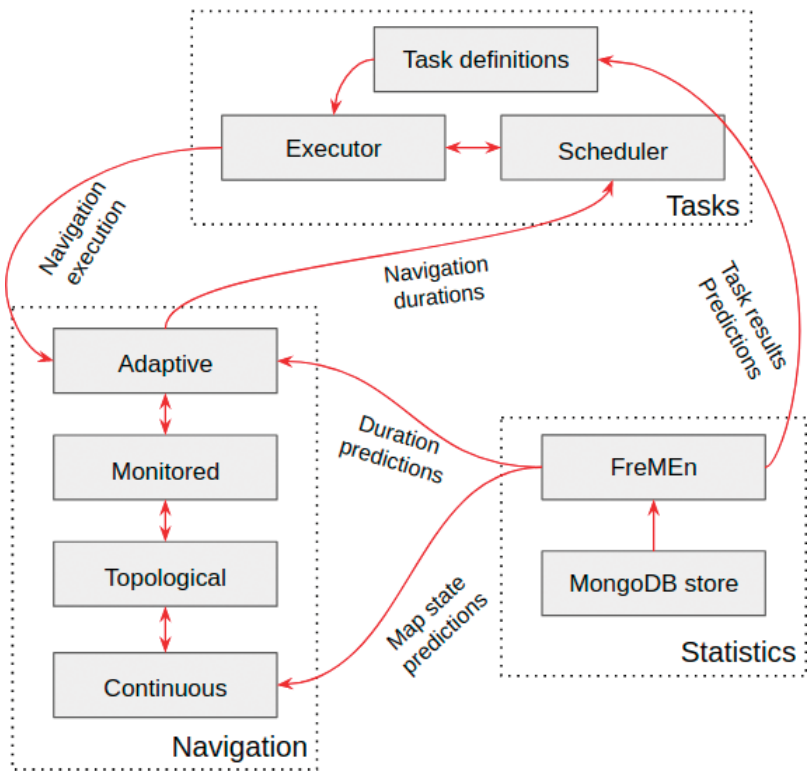


Figure 3. The Strands system—Overview.

further be used to generate a textured mesh through which a convolutional neural network can be trained which can successfully recognize the object in future observations [31, 32]. The dynamics detected in the environment can be used to detect patterns, either through spectral analysis (i.e., by applying a Fourier transform on the raw detection data), as described in [54], or as part of a multitarget tracking system based on a Rao-Blackwellized particle filter.

In addition to the detection and modeling of objects, the Strands perception system also focuses on the detection of people. Beyer et al. [55] present a method to continuously estimate the head-pose of people, while in [15] laser and RGB-D are combined to reliably detect humans and to allow human-aware navigation approaches which make the robot more socially acceptable. Beyer et al. [56] propose a CNN-based system which uses laser scanner data to detect objects; the usefulness of the approach is demonstrated in the case scenario, where it is used to detect wheelchairs and walkers.

Robust perception algorithms that can operate reliably for extended periods of time are one of the cornerstones of the Strands system. However, any algorithm deployed on the robot has to be not only robust, but also able to scale as the robot makes more observations and collects more information about the world. One of the key parts that would enable the successful operation of such a robotic system is a perception stack that is able to continuously integrate observations about the world, extract relevant parts as well as build models that understand and are able to predict what the environment will look like in the future. This spatio-temporal understanding is crucial, as it allows a mobile robot to compress the data acquired during months of autonomous operation into models that can be used to refine the robot's operation over time. Modeling periodicities in the environment and integrating them into a planning pipeline is further investigated by Fentanes et al. [57], while Santos et al. [58] build spatio-temporal models of the environment and use them for exploration through an information-theoretic approach which predicts the potential gain of observing particular areas of the world at different points in time.

4.2. The RobDREAM project

Advanced robots operating in complex and dynamic environments require intelligent perception algorithms to navigate collision-free, analyze scenes, recognize relevant objects, and manipulate them. Nowadays, the perception of mobile manipulation systems often fails if the context changes due to a variation, e.g., in the lightning conditions, the utilized objects, the manipulation area, or the environment. Then, a robotic expert is needed who needs to adjust the parameters of the perception algorithm and the utilized sensor or even select a better method or sensor. Thus, a high-level cognitive ability that is required for operating alongside humans is to continuously improve performance based on introspection. This adaptability to changing situations requires different aspects of machine learning, e.g., storing experiences for life-long learning, generating annotated datasets for supervised learning through user interaction, Bayesian optimization to avoid brute-force search in high-dimensional data, and a unified representation of data and meta-data to facilitate knowledge transfer.

The RobDREAM consortium automated and integrated different aspects of these. Specifically, in the EU's H2020 RobDREAM project, a mobile manipulator was used to showcase the

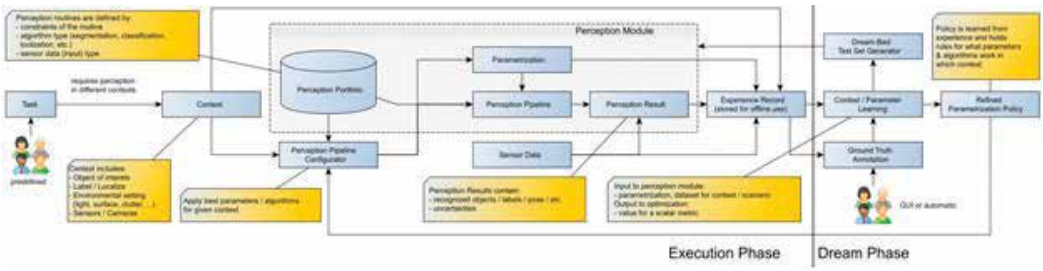


Figure 4. Schematics of the RobDREAM approach (image based on deliverables of http://robdream.eu/).

intuitive programming and simplified setup of robotic applications enabled by automatically tuning task execution pipelines according to user-defined performance criteria.

As illustrated in **Figure 4**, this was achieved by a semantically annotated logging of perceptual episodic memories that can be queried intuitively in order to analyze the performance of the system in different contexts. Then, a ground truth annotation tool can be used by the user to mark satisfying results, or correct unsatisfying ones, where the suggestions and interactive capabilities of the system reduced the cognitive load of this often complicated task (especially when it comes to 6 DoF pose annotations), as shown in user studies involving computer vision expert and nonexpert users alike.

These annotations are then used by a Bayesian optimization framework to tune the off-the-shelf pipeline to the specific scenarios the robot encounters, thereby incrementally improving the performance of the system. The project did not focus only on perception, but on other key technologies for mobile manipulation as well. Bayesian optimization and other techniques were used to adapt the navigation, manipulation, and grasping capabilities independently of each other and the perception ones. However, the combinatorial complexity of the joint parameter space of all the involved steps was too much even for such intelligent meta-learners. The final industrially relevant use-case demo featured the kitting and mounting of electric cabinet board elements, for which a pose-annotated database was built using two RBD-D cameras and released to the public (http://www.dlr.de/rm/thr-dataset).

4.3. The SPENCER project

When deploying robots in scenarios where they need to share the environment and interact with a large number of people, it is increasingly important that their functionalities are "socially aware." This means that they respect the personal space (and also privacy) of encountered persons, does not navigate s.t. to cut up cues or groups, etc. Such functionalities go beyond the usual focus of robotics research groups, while academics focusing on user experience typically do not have the means to develop radically new robots. However, the EU's FP7 program funded such an interdisciplinary project, called SPENCER, driven by an end-user in the aviation industry.

http://dx.doi.org/10.5772/intechopen.79742

Since around 80% of passenger traffic at different hubs, including Schiphol in Amsterdam, is comprised of passengers who are transferring from one flight to the other, KLM is interested in an efficient management of their movements. For example, when transfer times are short, and finding one's way in a big airport is difficult due to language and alphabet barriers, people are at risk to losing their connection. In such, and similar cases, robotic assistants that can be deployed and booked flexibly can possibly help alleviate some of the problem. This use-case was explored by the SPENCER demonstrator for smart passengers' flow management and mobile information provider, but similar solutions are required in other domains as well (**Figure 5**).

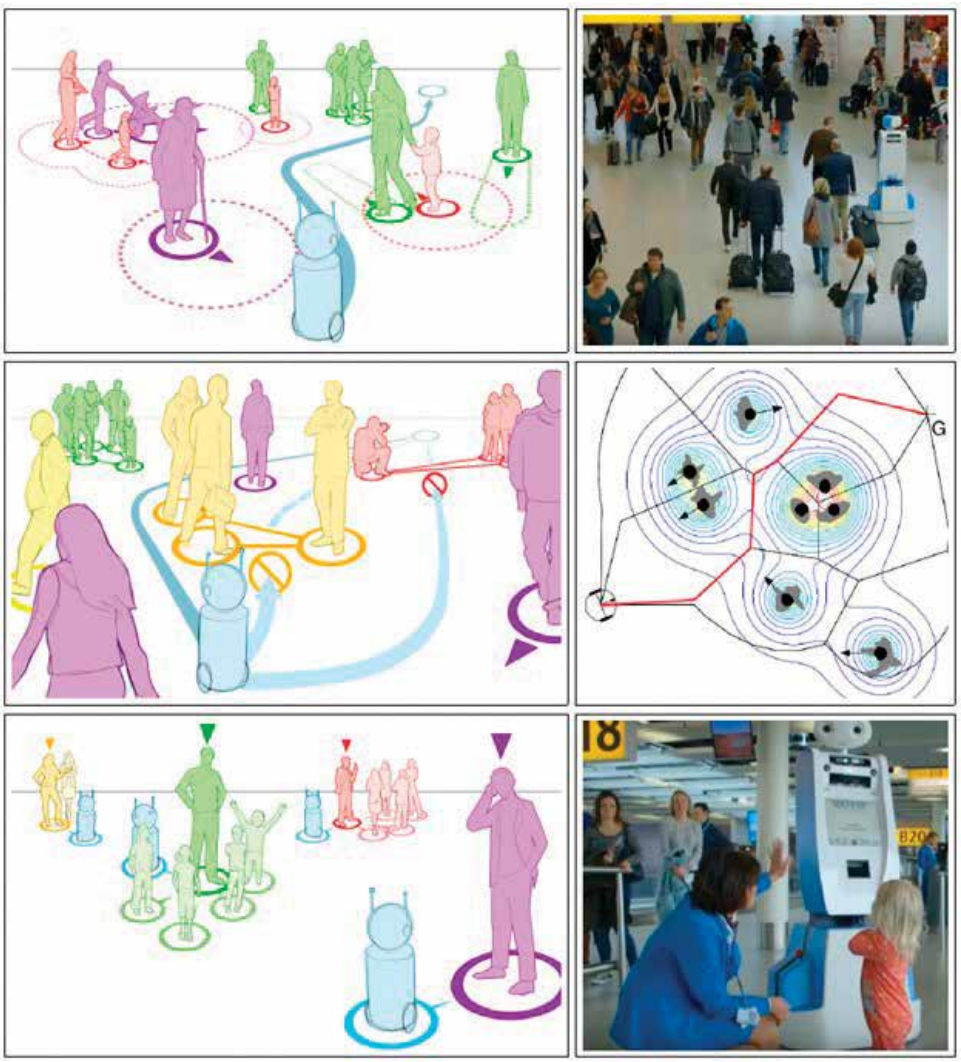


Figure 5. Concept and results of the SPENCER project (images from http://www.spencer.eu/).

The SPENCER consortium integrated the developed technologies onto a robot platform whose task consists in picking up short-transfer time passenger groups at their gate of arrival, identifying them with an onboard boarding pass reader, guiding them to the Schengen barrier and instructing them to use the priority track [59]. Additionally, the platform was equipped with a KLM information kiosk and provides services to passengers in need of help.

In crowded environments such as airports, generating short and safe paths for mobile robots is still difficult. Thus, social scene understanding and long-term prediction of human motion in crowds is not sufficiently solved but highly relevant for all robots that need to quickly navigate in human environments, possibly under temporal constraints. Social scene understanding means, in part, that a reliable tracking and prediction of people's motion with low uncertainty is available, and that is particularly hard if there are too many occlusions and too many fast changes of motion direction. Classical path planning approaches often result in an overconstrained or overly cautious robot that either fails to produce a feasible and safe path in the crowd, or plans a large and suboptimal detour to avoid people in the scene.

4.4. The AUTOCITS project

The AUTOCITS (https://www.autocits.eu/) project will carry out a comprehensive assessment of cooperative systems and autonomous driving by deploying real-world Pilots, and will study and review regulations related to automated and autonomous driving. AUTOCITS, cofinanced by the European Union through the Connecting Europe Facility (CEF) Program, aims to facilitate the deployment of autonomous vehicles in European roads, and to use connected/cooperative intelligent transport systems (C-ITS) services to share information between autonomous vehicles and infrastructure, by means of V2V and V2I communication technology, to improve safety and to facilitate the coexistence of autonomous cars in real-world traffic conditions. The AUTOCITS Pilots, involving connected and autonomous vehicles (including autonomous shuttles, i.e., low-speed robot-vehicles), will be deployed in three major European cities in "the Atlantic Corridor of the European Network": Lisbon (Portugal), Madrid (Spain), and Paris (France).

A number of technologies are involved in AUTOCITS, ranging from the onboard and roadside units (OBU, RSU) to the autonomous driving systems that equip the cars. Today, the autonomous and/or automated driving technology we see on the roads belongs to the levels 3 or 4 (with respect to the SAE's levels of automation in vehicles). In AUTOCITS, the Pilot's deployment will be of level 3 to 4. In this context, it is important to say that level 5 cars (i.e., 100% self-driving or full-automated cars: the driving wheels would be unnecessary) operating in real-world roads and streets are still far from reality.

We can say that the perception system is in charge of all tasks related to object and event detection and response (OEDR). Therefore, a perception system—including of course its software modules—is responsible for sensing, understanding, and reasoning about the autonomous car's surroundings. Within a connected and cooperative environment, connected cars would leverage and complement onboard sensor data by using information from vehicular communication systems (i.e., V2X technology): information from other connected vehicles, from infrastructure, and road users (and vice-versa).

5. Conclusions and remarks

So just how capable is current perception and AI, and how close did/can it get to human-level performance? Szeliski [60] in his introductory book to computer vision argued that traditional vision struggled to reach the performance of a 2-year old child, but today's CNNs reach super-human classification performance on restricted domains (e.g., in the ImageNet Large Scale Visual Recognition Challenge: http://www.image-net.org/challenges/LSVRC/).

The recent surge and interest in deep-learning methods for perception has greatly improved performance in a variety of tasks such as object detection, recognition, semantic segmentation, etc. One of the main reasons for these advancements is that working on perception systems lends itself easily to offline experimentation on publicly available datasets, and comparison to other methods via standard benchmarks and competitions.

Machine learning (ML) and deep learning (DL), the latter has been one of the most used keywords in some conferences in robotics recently, are consolidated topics embraced by the robotics community nowadays. While one can interpret the filters of CNNs as Gabor filters and assume to be analogous to functions of the visual cortex, currently, deep learning is a purely nonsymbolic approach to AI/ML, and thus not expected to produce "strong" AI/ML. However, even at the current level, its usefulness is undeniable, and perhaps, the most eloquent example comes from the world of autonomous driving which brings together the robotics and the computer vision community. A number of other robotics-related products are starting to be commercially available for increasingly complex tasks such as visual question and answering systems, video captioning and activity recognition, large-scale human detection and tracking in videos, or anomaly detection in images for factory automation.

Author details

Cristiano Premebida[1]*, Rares Ambrus[2] and Zoltan-Csaba Marton[3]

*Address all correspondence to: cpremebida@isr.uc.pt

1 Institute of Systems and Robotics (ISR-UC), Coimbra, Portugal

2 Toyota Research Institute, Los Altos, California, USA

3 German Aerospace Center, Köln, Germany

References

[1] Russell SJ, Norvig P. Artificial Intelligence: A Modern Approach. 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2; 2003

[2] Rennie C, Shome R, Bekris KE, Souza AF. A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place. IEEE Robotics and Automation Letters. July 2016;**1**(2), pp. 1179-1185

[3] Bore N, Jensfelt P, Folkesson J. Multiple object detection, tracking and long-term dynamics learning in large 3D maps. CoRR, https://arxiv.org/abs/1801.09292. 2018

[4] Handa A, Patraucean V, Badrinarayanan V, Stent S, Cipolla R. Understanding real world indoor scenes with synthetic data. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 4077-4085

[5] Firman M. RGBD datasets: Past, present and future. Firman 2016 RGBDDP, In: 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) on Large Scale 3D Data: Acquisition, Modelling and Analysis; 2016. 661-673

[6] Sünderhauf N, Dayoub F, McMahon S, Talbot B, Schulz R, Corke P, Wyeth G, Upcroft B, Milford M. Place categorization and semantic mapping on a mobile robot. In: IEEE International Conference on Robotics and Automation (ICRA); Stockholm; 2016. pp. 5729-5736

[7] Brucker M, Durner M, Ambrus R, Csaba Marton Z, Wendt A, Jensfelt P, Arras KO, Triebel R. Semantic labeling of indoor environments from 3D RGB maps. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2018

[8] Saarinen J, Andreasson H, Lilienthal AJ. Independent Markov chain occupancy grid maps for representation of dynamic environment. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2012. pp. 3489-3495

[9] Fong T, Nourbakhsh I, Dautenhahn K. A survey of socially interactive robots. Robotics and Autonomous Systems. 2003;**42**(3-4):143-166

[10] Diego R. Faria, Mario Vieira, Premebida C, Nunes U. Probabilistic human daily activity recognition towards robot-assisted living. In: Proceedings of the IEEE RO-MAN'15; Japan; 2015

[11] Manduchi R, Castano A, Talukder A, Matthies L. Obstacle detection and terrain classification for autonomous off-road navigation. Autonomous Robots. 2005;**18**(1):81-102

[12] Fernandes R, Premebida C, Peixoto P, Wolf D, Nunes U. Road detection using high resolution LIDAR. In: IEEE Vehicle Power and Propulsion Conference, IEEE-VPPC; 2014

[13] Asvadi A, Garrote L, Premebida C, Peixoto P, Nunes UJ. Multimodal vehicle detection: Fusing 3D LIDAR and color camera data. Pattern Recognition Letters. Elsevier. 2017

[14] Premebida C, Nunes U. Fusing LIDAR, camera and semantic information: A context-based approach for pedestrian detection. The International Journal of Robotics Research. 2013;**32**(3):371-384

[15] Dondrup C, Bellotto N, Jovan F, Hanheide M. Real-time multisensor people tracking for human-robot spatial interaction. IEEE International Conference on Robotics and Automation (ICRA); 2015

[16] Andreasson H, Magnusson M, Lilienthal A. Has something changed here? Autonomous difference detection for security patrol robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2007. pp. 3429-3435

[17] Sünderhauf N, Brock O, Scheirer W, Hadsell R, Fox D, Leitner J, Upcroft B, Abbeel P, Burgard W, Milford M, Corke P. The limits and potentials of deep learning for robotics. The International Journal of Robotics Research. 2018;**37**(4-5):405-420

[18] Correll N, Bekris KE, Berenson D, Brock O, Causo A, Hauser K, Okada K, Rodriguez A, Romano JM, Wurman PR, et al. IEEE Transactions on Automation Science and Engineering. 2016

[19] Wagstaff K. Machine learning that matters. In: Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML); 2012. pp. 529-536

[20] Moravec H, Elfes A. High resolution maps from wide angle sonar. In: Proceedings of IEEE International Conference on Robotics and Automation; 1985. pp. 116-121

[21] Hornung A, Wurm KM, Bennewitz M, Stachniss C, Burgard W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots. 2013

[22] Thrun S, Burgard W, Fox D. Probabilistic Robotics. The MIT Press; 2005. ISBN:0262201623

[23] Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I, Leonard JJ. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics. 2016;**32**(6):1309-1332

[24] Biber P, Ducket T. Experimental analysis of sample-based maps for long-term SLAM. The International Journal of Robotics Research. 2009;**28**:20-33

[25] Walcott-Bryant A, Kaess M, Johannsson H, Leonard JJ. Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2012. pp. 1871-1878

[26] Saarinen J, Stoyanov T, Andreasson H, Lilienthal AJ. Fast 3D mapping in highly dynamic environments using normal distributions transform occupancy maps. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2013. pp. 4694-4701

[27] Hermans A, Floros G, Leibe B. Dense 3D semantic mapping of indoor scenes from RGB-D images. In: IEEE International Conference on Robotics and Automation (ICRA), Hong Kong; 2014. pp. 2631-2638

[28] Krahenbuhl P, Koltun V. Efficient inference in fully connected CRFs with Gaussian edge potentials. Advances in Neural Information Processing Systems. 2016;**24**:109-117

[29] McCormac J, Handa A, Davison A, Leutenegger S. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In: IEEE International Conference on Robotics and Automation (ICRA), Singapore; 2017. pp. 4628-4635

[30] Whelan T, Leutenegger S, Salas-Moreno RF, Glocker B, Davison AJ. ElasticFusion: Dense SLAM without a pose graph. Robotics: Science and Systems. 2015

[31] Ambrus R, Claici S, Wendt A. Automatic room segmentation from unstructured 3-D data of indoor environments. IEEE Robotics and Automation Letters. 2017;**2**(2):749-756

[32] Ambrus R, Bore N, Folkesson J, Jensfelt P. Autonomous meshing, texturing and recognition of object models with a mobile robot. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC. 2017. pp. 5071-5078

[33] Armeni I, Sener O, Zamir AR, Jiang H, Brilakis I, Fischer M, Savarese S. 3D semantic parsing of large-scale indoor spaces. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV; 2016. pp. 1534-1543

[34] Mura C, Mattausch O, Pajarola R. Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. Computer Graphics Forum. 2016;**35**:179-188

[35] Sperry RW. Neurology and the mind-body problem. American Scientist. 1952;**40**:291-312

[36] Turing AM. Intelligent machinery. Journal of Machine Intelligence. 1970;**5**, different sources cite 1947 and 1948 as the time of writing

[37] Turing AM. Computing machinery and intelligence. Mind. 1950;**LIX**:433-460

[38] Steels L, Brooks RA, editors. The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc.; 1995

[39] Vernon D. Cognitive vision: The case for embodied perception. In: Image and Vision Computing. Elsevier. 1 January 2008;**26**(1):127-140

[40] Bohg J, Hausman K, Sankaran B, Brock O, Kragic D, Schaal S, Sukhatme G. Interactive perception: Leveraging action in perception and perception in action. IEEE Transactions on Robotics. 2017;**33**:1273-1291

[41] Ferrari C, Canny J. Planning optimal grasps. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA); Vol. 3; 1992. pp. 2290-2295

[42] Frisby JP, Stone JV. Seeing objects, ch. 8. In: Seeing: The Computational Approach to Biological Vision. MIT Press; 2010. pp. 178-179

[43] Ulrich M, Wiedemann C, Steger C. Cad-based recognition of 3d objects in monocular images. In: IEEE International Conference on Robotics and Automation (ICRA); Vol. 9; 2009. pp. 1191-1198

[44] Hinterstoisser S, Lepetit V, Ilic S, Holzer S, Bradski G, Konolige K, Navab N. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: Asian conference on computer vision; Springer; 2012. pp. 548-562

[45] Tjaden H, Schwanecke U, Schomer E. Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In: Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE; 2017. pp. 124-132

[46] Brachmann E, Krull A, Michel F, Gumhold S, Shotton J, Rother C. Learning 6d object pose estimation using 3d object coordinates. In: European Conference on Computer Vision. Springer; 2014. pp. 536-551

[47] Krull A, Michel F, Brachmann E, Gumhold S, Ihrke S, Rother C. 6-DOF model based tracking via object coordinate regression. In: Asian Conference on Computer Vision. Springer; 2014. pp. 384-399

[48] Kehl W, Milletari F, Tombari F, Ilic S, Navab N. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In: European Conference on Computer Vision. Springer; 2016. pp. 205-220

[49] Lepetit V, Moreno-Noguer F, Fua P. Epnp: An accurate o(n) solution to the pnp problem. International Journal of Computer Vision. 2009;**81**(2)

[50] Janai J, Guney F, Behl A, Geiger A. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. arXiv:1704.05519v1. 2018

[51] Tas OS, Salscheider NO, Poggenhans F, Wirges S, Bandera C, Zofka MR, Strauss T, Zollner JM, Stiller C. Making Bertha Cooperate—Team AnnieWAY's entry to the 2016 Grand Cooperative Driving Challenge. IEEE Transactions on Intelligent Transportation Systems. 2018;**19**(4):1262-1276

[52] Hawes N, Burbridge C, Jovan F, Kunze L, Lacerda B, Mudrova L, Young J, Wyatt J, Hebesberger D, Kortner T, Ambrus R, Bore N, Folkesson J, Jensfelt P, Beyer L, Hermans A, Leibe B, Aldoma A, Faulhammer T, Zillich M, Vincze M, Chinellato E, Al-Omari M, Duckworth P, Gatsoulis Y, Hogg DC, Cohn AG, Dondrup C, Pulido Fentanes J, Krajnik T, Santos JM, Duckett T, Hanheide M. The STRANDS project: Long-term autonomy in everyday environments. IEEE Robotics and Automation Magazine. 2017;**24**:146-156

[53] Fäulhammer T, Ambruş R, Burbridge C, Zillich M, Folkesson J, Hawes N, Jensfelt P, Vincze M. Autonomous learning of object models on a mobile robot. IEEE Robotics and Automation Letters. 2017;**2**(1):26-33

[54] Krajnik T, Fentanes JP, Cielniak G, Dondrup C, Duckett T. Spectral analysis for long-term robotic mapping. In: IEEE International Conference on Robotics and Automation (ICRA), Hong Kong. 2014. pp. 3706-3711

[55] Beyer L, Hermans A, Leibe B. Biternion nets: Continuous head pose regression from discrete training labels. German Conference on Pattern Recognition. 2014:157-168

[56] Beyer L, Hermans A, Leibe B. DROW: Real-time deep learning-based wheelchair detection in 2-D range data. IEEE Robotics and Automation Letters. 2017;**2**(2):585-592

[57] Fentanes JP, Lacerda B, Krajnik T, Hawes N, Hanheide M. Now or later? Predicting and maximising success of navigation actions from long-term experience. In: IEEE International Conference on Robotics and Automation (ICRA), Seattle, USA. 2015. pp. 1112-1117

[58] Santos JM, Krajník T, Fentanes JP, Duckett T. Lifelong information-driven exploration to complete and refine 4-D spatio-temporal maps. IEEE Robotics and Automation Letters. 2016;**1**(2):684-691

[59] Triebel R, Arras K, Alami R, Beyer L, Breuers S, Chatila R, Chetouani M, Cremers D, Evers V, Fiore M, Hung H, Islas Ramírez OA, Joosse M, Khambhaita H, Kucner T, Leibe B, Lilienthal AJ, Linder T, Lohse M, Magnusson M, Okal B, Palmieri L, Rafi U, van Rooij M, Zhang L. SPENCER: A socially aware service robot for passenger guidance and help in busy airports. Field and Service Robotics: Results of the 10th International Conference. 2016. 607-622

[60] Szeliski R. Computer vision: Algorithms and applications ser. In: Texts in Computer Science. New York, NY, USA: Springer-Verlag New York, Inc.; 2010

Chapter 7

Online Mapping-Based Navigation System for Wheeled Mobile Robot in Road Following and Roundabout

Mohammed A. H. Ali and Musa Mailah

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79412

Abstract

A road mapping and feature extraction for mobile robot navigation in road roundabout and road following environments is presented in this chapter. In this work, the online mapping of mobile robot employing the utilization of sensor fusion technique is used to extract the road characteristics that will be used with path planning algorithm to enable the robot to move from a certain start position to predetermined goal, such as road curbs, road borders, and roundabout. The sensor fusion is performed using many sensors, namely, laser range finder, camera, and odometry, which are combined on a new wheeled mobile robot prototype to determine the best optimum path of the robot and localize it within its environments. The local maps are developed using an image's preprocessing and processing algorithms and an artificial threshold of LRF signal processing to recognize the road environment parameters such as road curbs, width, and roundabout. The path planning in the road environments is accomplished using a novel approach so called Laser Simulator to find the trajectory in the local maps developed by sensor fusion. Results show the capability of the wheeled mobile robot to effectively recognize the road environments, build a local mapping, and find the path in both road following and roundabout.

Keywords: LRF, vision system, odometry, laser simulator, road roundabout, road following, 2D map, local map

1. Introduction

Autonomous navigation can be defined as the ability of the mobile robot to determine its position within the reference frame environment using suitable sensors, plan its path mission through the terrain from the start toward the goal position using high planner techniques and

perform the path using actuators, all with a high degree of autonomy. In other words, the robot during navigation must be able to answer the following questions [1]:

- Where have I been? It is solved using cognitive maps.
- Where am I? It is determined by the localization algorithm.
- Where am I going? It is done by path planning.
- How can I go there? It is performed by motion control system.

Autonomous navigation comprises many tasks between the sensing and actuating processes that can be further divided into two approaches: behavior-based navigation and model-based navigation. The behavior-based navigation depends on the interaction of some asynchronous sets of behaviors of the robot and the environment such as: build maps, explore, avoid obstacles, and follow right/left and goal seeking, which are fused together to generate suitable actions for the actuators. The model-based navigation includes four subtasks that are performed synchronously step by step: perception, localization, path planning, and motion control. This is the common navigation systems that have been used and found in the literature.

The model-based navigation will be explained in details, since it is implemented in the proposed navigation system. In the perception task of this navigation model, the sensors are used to acquire and produce enormous information and good observation for the environments, where the robot navigates. There is a range of sensors that can be chosen for the autonomous navigation depending on the task of mobile robot, which generally can be classified into two groups: absolute position measurements and relative position measurements [2]. The European robotics platform (EUROP) marks the perception process in the robotics system as the main problem which needs further research solutions [1].

The localization process can be perceived as the answer to the question: where am I? In order to enable the robot to find its location within the environments, two types of information are needed. First, a-priori information is given to the robot from maps or cause-effect relationships in an initialization phase. Second, the robot acquires information about the environment through observation by sensors. Sensor fusion techniques are always used to combine the initial information and position measurements of sensors to estimate the location of the robot instantly [1].

In the path planning process, the collision-free path between start and target positions is determined continuously. There are two types of path planning: (a) global planning or deliberative technique and (b) local path planning or sensor-based planning. In the former, the surrounding terrain of the mobile robot is known totally, and then the collision-free path is determined offline, while in the latter, the surrounding terrain of the mobile robot is partially or totally unknown, and the robot uses the sensor data for planning the path online in the environments [1]. The path planning consists of three main steps, namely, the environment's modeling, determination of the collision-free trajectory from the initial to target positions, and searching continuously for the goal [1].

In motion control, the mobile robot should apply a steering strategy that attempts to prevent slippage and reduce position errors. The control algorithm has to guarantee a zero steady-state

orientation or position error. The kinematic and dynamic analysis of the mobile robot is used to find the parameters of the controllers for continuously avoiding the obstacles and moving toward the target position through the predefined path [1]. The autonomous navigation system can be classified according to the environmental features as follows:

i. Structured environments: it is usually found in indoor applications, where the environments are totally known.

ii. Semi-structured environments: the environments are partially known.

iii. Unstructured environments: outdoor applications are almost unstructured, where the environments are totally unknown.

There are many factors that can make outdoor autonomous navigation very complex in comparison to the indoor navigation:

i. The borders of surrounding terrain in the indoor application are clear and distinct. However, in contrary to that, they are mostly "faded" in outdoor setting.

ii. The robot algorithm should be able to discover and deal with many things that are not in the initial planning.

iii. The locomotion of robot will be different depending on the terrain roughness.

iv. Outdoor autonomous navigation should have adequate robustness and reliable enough because the robot typically works and meddles with people and encounters unexpected moving obstacles.

v. Weather changes can affect the mobile robot sensor measurement, electronic devices, and actuators. For example, the inhomogeneous of the light will affect the camera when capturing the image of the scene, the sunray affects the laser measurement, and ultrasonic measurement becomes unavailable.

Although the outdoor navigation in urban area is a topic involving a wide range and level of difficulties, it is attractive and forms a challenge to the researchers who are motivated to discover solution(s) to the problems via their proposed developed technique. Until today, there is as yet a robot or vehicle, which is able to drive fully autonomously on the roads of urban buildings, taking into account a high level of robust interaction with its environments [3, 4]. This vehicle is expected to operate in a dangerous situation, since it always passes through crowded area, must follow printed or signal traffic lights and more often than not, it is in touch with the people.

Most navigation systems on the roads are performed using two or more sensors to obtain the parameters of the surrounding environment, taking into consideration various situations and conditions. Sensor fusion technique is the common method that has been used in navigation of mobile robot in road environments. In this model, the methods for extracting the features of roads can be done with reference to either behavior-based navigation or model-based navigation as described in the following paragraphs.

Global positioning system (GPS) is used as the main sensor and combined with odometry and LRF for trajectory tracking, obstacle avoiding and localization in curbed roads [5] or navigating

mobile robot between urban buildings [6]. GPS can be combined with LRF and dead reckoning for navigating vehicles in roads using beacons and landmarks [7], combined with camera vision for localizing mobile robot in color marked roads [8], combined with 3D laser scanner for building compact maps to be used in a miscellaneous navigation applications of mobile robot [9], combined with IMU, camera vision, and sonar altimeter for navigating unmanned helicopter through urban building [10] or combined with LRF and inertial sensors for leading mobile robot ATRV-JR in paved and rugged terrain [11, 12]. GPS can be combined with INS and odometry sensor for helping GPS to increase its positioning accuracy of vehicles [13]. GPS is combined with odometry and GIS (geographical information system) for road matching-based vehicle navigation [14]. GPS can also be combined with LIDAR (light detection and ranging) and INS for leading wheelchair in urban building by 3D map-based navigation [15]. GPS is combined with a video camera and INS for navigating the vehicles by lane detection of roads [16]. GPS is combined with INS for increasing position estimation of vehicles [17]. GPS is combined with INS and odometry for localizing mobile robot in urban buildings [18]. Camera video with odometry is used for navigating land vehicle, road following by lane signs and obstacles avoiding [19]. Omni-directional infrared vision system can be used for localizing patrol mobile robot in an electrical station environment [20]. 3D map building and urban scenes are used in mobile robot navigation by fusing stereo vision and LRF [21].

LRF can be combined with cameras and odometry for online modeling of road boundary navigation [6], for enhancing position accuracy during mobile robot navigation [22] or for correcting trajectory navigation of mobile robot [23]. Also, it can be combined with compass and odometry for building map-based mobile robot navigation [1]. It can be combined with color CCD camera for modeling roads and driving mobile robot in paved and unpaved road [24], for crossing roads through landmark detection [25], for obstacle detecting and avoiding [26] or for road recognition process [27]. It can be combined with sonar and camera vision for navigating mobile robot when crossing roads [1]. 3D LRF is combined with two color camera, INS, and odometry for cross-country navigation of ADAM mobile robot through natural terrain [1]. It can be combined with IMU, CCD camera, and odometry for guiding tractor vehicle in Citrus Grove [1]. It can be combined with camera, LRF, sonar, and compass for building 3D map of an urban environment [1].

Priori map can be used for navigation by mapping precisely environment landmarks [1]. Dead reckoning is used for estimating position accurately by feature detection in mobile robot [1]. A hybrid methodology between the teleoperating and autonomous navigation system has been applied in a curbed road environment as explained in [5]. A combination of differential global positioning system (DGPS) and odometry with extended *Kalman* filter is used to localize the mobile robot in a road environment. LRF is used for obstacle avoidance by finding the suitable path to pass through the road curbs and to estimate the position of the road curbs during trajectory tracking. The main setback in this work is that the robot cannot work autonomously in road environments.

Jose et al. [7] presented a robust outdoor navigation system using LRF, and dead reckoning is proposed for navigating vehicles using beacons and landmarks on the roads. The system is studied considering several cylindrical or V-shaped objects (beacons) that can be detected

using LRF in a road environment. The information filter (derived from *Kalman* filter) is used to gather data from sensors and build map and determine the localization of robot. The results show that the accuracy of the online mapping is very close to DGPS, which has accuracy equal to 1 cm. Donghoon et al. [8] proposed a navigation system method based on fusing two localization systems is proposed. The first one uses a camera vision system with particle filter, while the second uses two GPS with *Kalman* filter. The printed marks on the road like arrows, lines, and cross are effectively detected through the camera system. By processing the camera's data using the hyperbola-pair lane detection, that is, *Delaunay* triangulation method for point extraction and mapping based on sky view of points, the robot is able to determine the current position within its environment. To eliminate the camera drawback like lens distortion and inaccuracy in the long run, GPS system data are used to complement and validate the camera's data. The errors between the predefined map and real-time measurements are found to be 0.78 m in x direction and 0.43 m in y direction.

Cristina et al. [9] suggested a compact 3D map building for a miscellaneous navigation application of mobile robot in real time performed through the use of the GPS and 3D laser scanner. A 3D laser scanner senses the environment surface with sufficient accuracy to obtain a map, which is processed to get a qualitative description of the traversable environment cell. The 3D local map is built using 2D *Voronoi* diagram model, while the third dimension is determined through the data from laser scanner for the terrain reversibility. The previous map is gathered with the GPS information to build a global map. This system is able to detect the slopes of navigated surfaces using *Brezets* algorithm and the roughness of the terrain using normal vector deviation for the whole region. Although the proposed algorithm is very complicated, the result shows that it is able to extract an area as big as 40×20 m in an outdoor environment for 0.89 s.

2. Platform overview

A new platform for an autonomous wheeled mobile robot navigation system has been designed and developed in our labs. It is comprised of the following parts as shown in **Figure 1**:

- Two motors, type: DC-brush with power *120 W* and model DKM.
- One spherical Castor wheel.
- 4 m range LRF; model *HOKUYO URG-04LX-UG01.*
- High resolution WiFi camera, model *JVC-GC-XA1B*.
- Two optical rotary odometry; model, *B106.*
- Motors drivers, types *SmartDrive 40.*
- Five cards for the interface free-based controllers system (*IFC*): interface power, abbreviated as *IFC-IP00;* interface computer, abbreviated as *IFC-IC00*; interface brushless, abbreviated as *IFC-BL02*; and interface brush motors, abbreviated as *IFCBH02*.

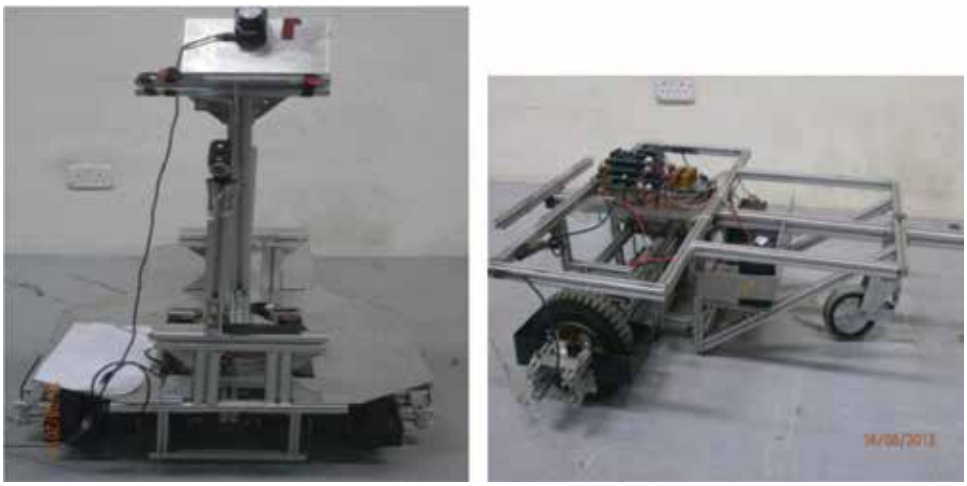


Figure 1. The developed platform used for experiments.

- The platform has an additional part such as battery with model *NP7-12* lead acid and the aluminum profile and sheet-based chassis as shown in **Figure 1**.

3. Sensor modeling and feature extraction

The data coming from each sensor in the platform were prepared in such a way that it enables for the extraction of the road features autonomously while navigating on the road. It is described in the following section.

3.1. Camera

JVC-GC-XA1B camera is utilized to figure out the environments in front of the robot with good resolution (760 × 320) and speed equal to 30 frame/s. The sequences of pictures are extracted from the live video using image processing tool boxes in MATLAB. An algorithm has been developed that can take the image sequences from the video and apply multiple operations to get a local map from the image and perform further calculation for road following and roundabout detection. In general, the image sequences processing algorithm consists of three main parts:

i. Preprocessing of the image for depth processing.

ii. Processing of the image and development of the environment local map.

iii. Post processing algorithms to perform the following subtasks:

- Roundabout detection based on LS approach.
- Road following in the roads without curbs.

The first and second steps will be explained here, while the third one will be explained in next section.

(i) Preprocessing of the image for depth processing.

A preprocessing algorithm is utilized to capture the environments of the road from the live video and prepare it in an appropriate form to be processed in the next processes. The main operations that have been used at this step can be briefly described as follows:

- Constructing the video input object: the video input object represents the connection between MATLAB and a particular image acquisition device.

  vid = videoinput('winvideo',3);

- Preview of the WiFi video: it creates a video preview window that displays live video data for video input object. The window also displays the timestamp and video resolution of each frame and the current status of video input object.

  preview (vid);

- Setting the brightness of live video

  The brightness of image's sequences is adjusted, since the aperture of camera is not automatic. Following command is used:

  set (vid.source, 'Brightness', 35);

- Start acquiring frames

  This function helps to reserve the image device for the exclusive use in this program and locks the configuration to others applications. Thus, certain properties become read only while running.

  Start(vid);

- Acquiring the image frames to MATLAB workspace

  It returns the data, which contains the number of frames specified in the (Frames per Trigger) property of the video input object.

  data = getdata(vid);

- Crop image

  It allows for cutting the interesting regions in the images. In the proposed algorithm, the bottom half of the image is cropped, since it contains the nearest area of the road to the robot; also to save time needed for calculation in comparison with whole image.

  GH = imcrop(data, [1 u1 io(2) io(1)]);

- Convert from RGB into grayscale

  The purpose is to make image useful for the edge detection and removing the noise operations.

  IS = rgb2gray(GH);

- Remove image acquisition object from memory

  It is used to free memory at the end of an image acquisition session and start a new frame acquisition.

  delete(vid).

(ii) Processing of the image and development of the environments local map.

It includes some operations that allow to extract the edges of the road and roundabout from the images with capability to remove the noises and perform filtrations.

The following operations are applied for edge detection and noise filtering:

1. 2D *Gaussian* filters: It is used for smoothing the detection of the edge. The point-spread function (PSF) for 2D continuous space *Gaussian* is given by:

$$g(x,y) = \frac{1}{2\pi\sigma^2}.e^{-\left(\frac{(x-x_0)^2}{2\sigma^2}+\frac{(y-y_0)^2}{2\sigma^2}\right)} \tag{1}$$

σ is the standard deviation of the *Gaussian* distribution.

The following *Gaussian* filter was applied to the image in MATLAB:

PSF = fspecial('*gaussian*', 3, 3).

2. Multidimensional images (*imfilter):* It allows the user to filter one array with another array according to the specified options. In the proposed image processing, a symmetric option is used to filter the image by the *Gaussian* filter as follows:

I = imfilter(*IS*, *PSF*, '*symmetric*', '*conv*').

IS is the gray value image of the scene, and *conv* indicates the linear convolution operation of the image in which each output pixel is the weighted sum of neighboring input pixels.

3. *Canny, Prewitt,* and *Sobel* filters for edge detection: these filters were used to find the edges of curbed road in the image. They are derivative-based operations for edge detection:

Canny filter: It finds the edges by looking for local maxima of the gradient of the image matrix. It passes through the following steps: smoothing the image by blurring the operation in order to prepare for removing the noise; finding the gradients of the edges, where the large magnitude gradients of the image should be marked; nonmaximum suppression operation which allow only the local maxima to be marked as edges; double thresholding applied to the potential edges; and edge tracking by hysteresis, where the final edges are determined by removing all edges that are not connected to certain strong edges (large magnitudes). This filter is implemented with threshold coming from *imfilter* as follows:

BW = edge (*I*, '*canny*', (*graythresh*(*I*) * .1)).

Prewitt gradient filter:

It finds the edges using the *Prewitt* approximation of the derivative. Each component of the filter takes the derivative in one direction and smoothies in the orthogonal direction using a *uniform* filter. This filter is implemented with threshold coming from *imfilter* as follows:

$$BW = edge\ (I, \text{'prewitt'}, (graythresh(I) * .1)).$$

Sobel gradient filter:

It finds the edges using the *Sobel* approximation of the derivative. Each filter takes the derivative in one direction and smoothies in the orthogonal direction using a triangular filter. This filter is implemented with the threshold coming from *imfilter* as follows:

$$BW = edge\ (I, \text{'sobel'}, (graythresh(I) * .1)).$$

4. Morphological operations

The morphological methodology is applied to improve the shape of the lines representing the road curb, which are extracted from edges in the images using the above-mentioned filters. Morphology technique includes a various set of image processing enhancing operations that process the images based on shapes. Morphological operations implement a structuring element to the input image and then carry out the output images with a similar size. The morphological operation compares the current pixel in the output image with the input one and its neighbors to calculate the value of each output image pixels. This is done by selecting a certain shape and size of the neighbors that are sensitive to the input image's shapes.

Two operations are used to perform the morphological method:

Morphological structuring element (*strel*): it is used to define the areas that will be applied using morphological operations. Straight lines with 0° and 90° are the shapes used for the images which actually represent the road curbs.

$$se90 = strel(\text{'line'}, 3, 90).$$

$$se0 = strel(\text{'line'}, 3, 0).$$

Dilation of image (*imdilate*): dilation and erosion are two common operations in morphology. With dilation operation, the pixels in the image have to be summed to the object boundaries. The numbers of the added pixels are depended on the element size and its structure that is utilized to process the input images. In dilation process, the status of the pixels in the output images is figured out by implementing a certain rule and equations to the corresponding input pixels and all its neighbors. In the developed algorithm, the *imdilate* is used with a range that comes from the *strel* operation as follows:

$$BW1 = imdilate(BWC, [se90\ se0]).$$

BWC is the binary image coming from the edges filters.

5. 2D order-statistic filtering (*ordfil2*)

It is also called Min/Max of the *Median* filter. It uses a more general approach for filtration, which allows for specifying and choosing the rank order of the filter as:

$$f2 = ordfilt2(x, 5, [1\ 1\ 1; 1\ 1\ 1; 1\ 1\ 1]).$$

Where, the second parameter specifies the rank order chosen, and the third parameter specify the mask with the neighbors specified by the nonzero elements in it.

6. Removing of small objects from binary image (*bwareaopen*)

This function is helpful to remove all connected components (objects) that have less than a certain threshold numbers of pixels from the binary image. The connectivity between the pixels can be defined at any direction of image; in MATLAB, it can be defined by default to 4-connected neighborhood or 8-connected neighborhood for the images. In the proposed algorithm, it is adjusted as follows:

$$BW2 = bwareaopen(BW1, 1200).$$

7. Filling of image regions and hole operation (*imfill*)

Due to the object in images are not clear, especially at the borders, this function is used to fill the image region that represents an object. This tool is very useful in the proposed algorithm to detect the roundabout intersection. One can describe the region filling operation as follows:

$$X_f = \left(X_{f-1} \oplus B\right) \cap A^C \quad f = 1, 2, 3, .. \tag{2}$$

X_f is any point inside the interested boundary, B is the symmetric structuring element, ∩ is the intersection area between A and B, and A^C is the complement of set A. The above-mentioned equation is performed in MATLAB by the following command:

$$P = imfill(BW2, \text{'holes'}).$$

The developed image sequences after processing looked like the one shown in **Figures 2** and **3**, which are used later on for Laser Simulator-based roundabout detection. Also, **Figures 8–11** portrayed in Section 4 show the capability of the algorithm to develop the local map in the indoor and outdoor road environments.

3.2. Laser range finder

LRF measurements are used to localize the robot in environments and for building 2D local map. As previously mentioned, this device can scan an area with 240° at 100 ms/scan. The surrounding environments in the LRF measurements look like the one shown in **Figure 4**.

Two coordinate systems are used to characterize the measurements of LRF as depicted in **Figure 5**, which are: LRF measuring coordinate and LRF environment coordinate systems. Since the LRF measurements are scanned starting from the origin point, the calculation is done based on the right triangle as depicted in **Figure 5**.

If the laser measurement of components in y_L direction is compared at points a, b, and c, one can find:

$$\begin{aligned} y_{\mathrm{Lc}} &= L_{\mathrm{c}}, \quad x_{\mathrm{Lc}} = 0 \\ y_{\mathrm{La}} &= L_{\mathrm{a}} \sin(\beta) = L_c, \quad \mathrm{x_{Lc}} = L_{\mathrm{a}} \cos(\beta) \\ y_{\mathrm{Lb}} &= L_{\mathrm{b}} \sin(\beta), \quad \mathrm{x_{Lb}} = L_{\mathrm{b}} \cos(\beta) \end{aligned} \tag{3}$$

L_a, L_b, and L_c are the length of laser measurements at point a, b, and c, respectively, as shown in **Figure 5**. β is the angle between the measurement point and the platform coordinate system which can be calculated from the scan area angle $\boldsymbol{L}_\theta$ = (0–240°). y_{La} and y_{Lc} represent the

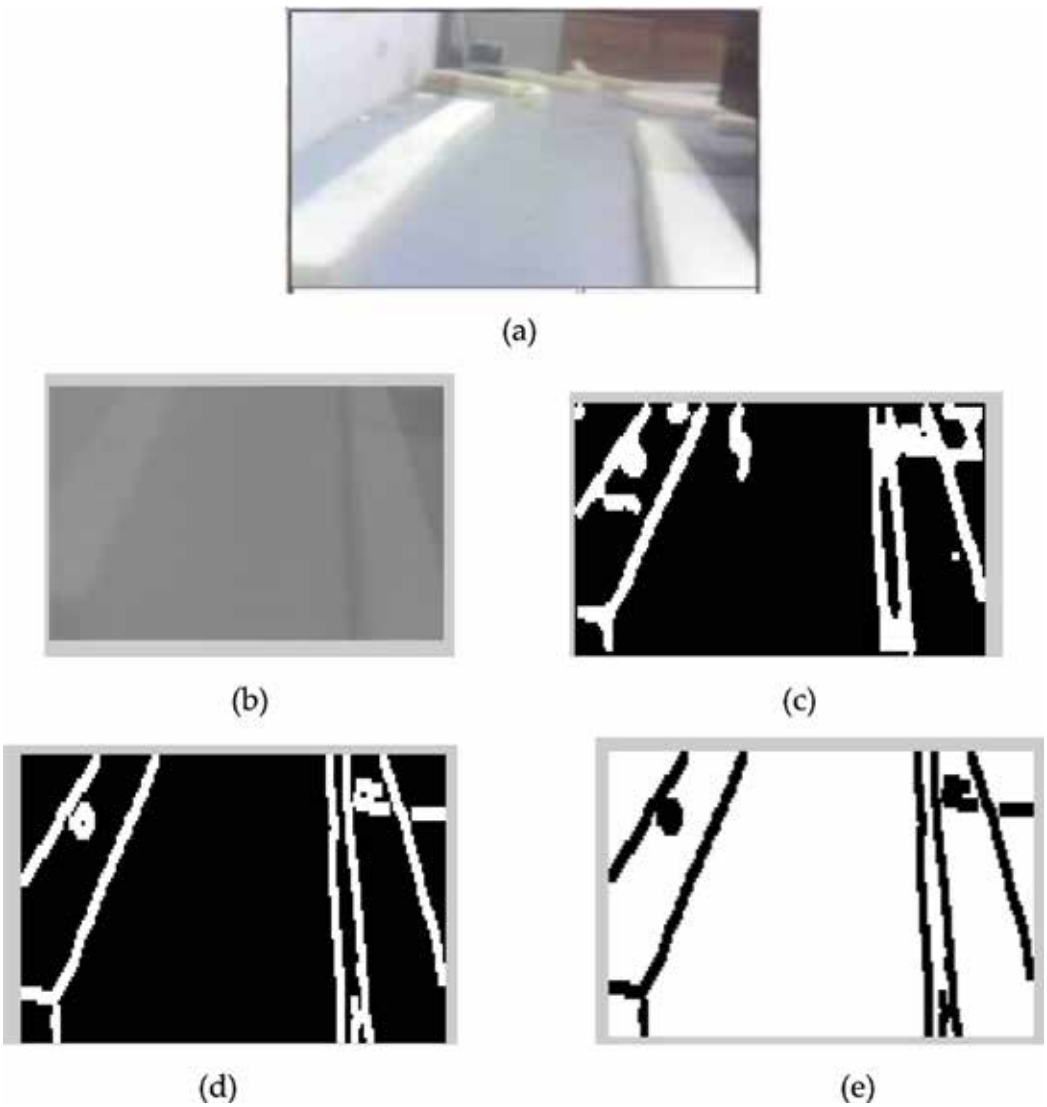


Figure 2. Image sequence processing where no roundabout can be detected, (a) original image, (b) curbed image in a gray, (c) after edge detection algorithm, (d) final processed image, and (e) developed image for local map.

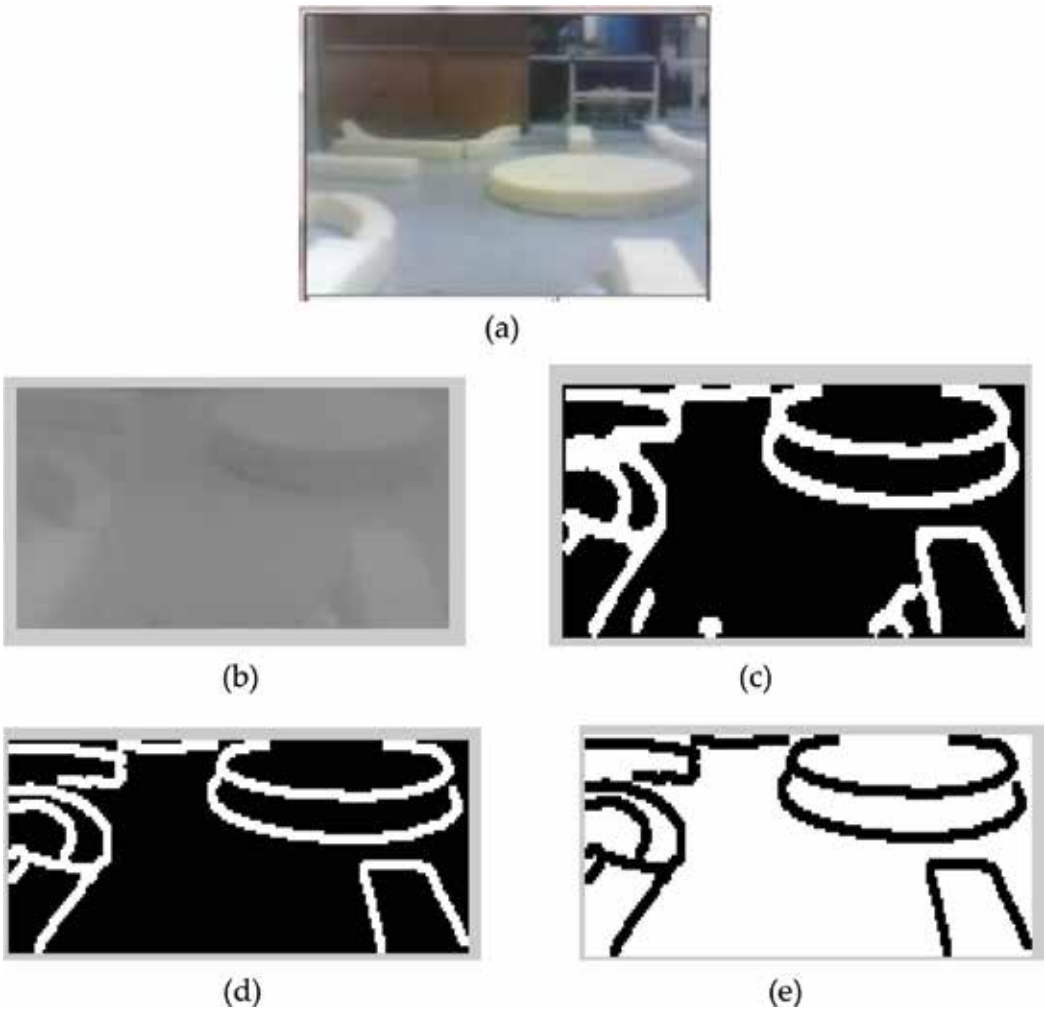


Figure 3. Image sequence processing where roundabout is detected, (a) original image, (b) curbed image in a gray value, (c) with edge detection algorithm, (d) final image, and (e) developed image for local map.

parameters of the road, and y_{Lb} represents the curb. It is found that y_{La} and y_{Lc} have the same length in y direction; however, the length of y_{Lb} is different and can be written as follows:

$$y_{\mathrm{Lb}} = y_{\mathrm{Lc}} - \frac{Z_{\mathrm{Rb}}}{\sin(\rho)} \Rightarrow y_{\mathrm{Lc}} - y_{\mathrm{Lb}} = \frac{Z_{\mathrm{Rb}}}{\sin(\rho)} \tag{4}$$

$Z_{\mathrm{Rb}} = h_{\mathrm{c}}$, h_{c} is idetified as the road curbed height which is known as a threshold in the program. ρ is defined as the LRF rays and the floor. For obstacle detection, two successive scan measurements (*i* and *ii*) have to be compared with each other in y_L direction in the location between e and d as illustrated in **Figure 5** to find the obstacles ahead of the robot as shown in Eqs. (5)–(7):

http://dx.doi.org/10.5772/intechopen.79412

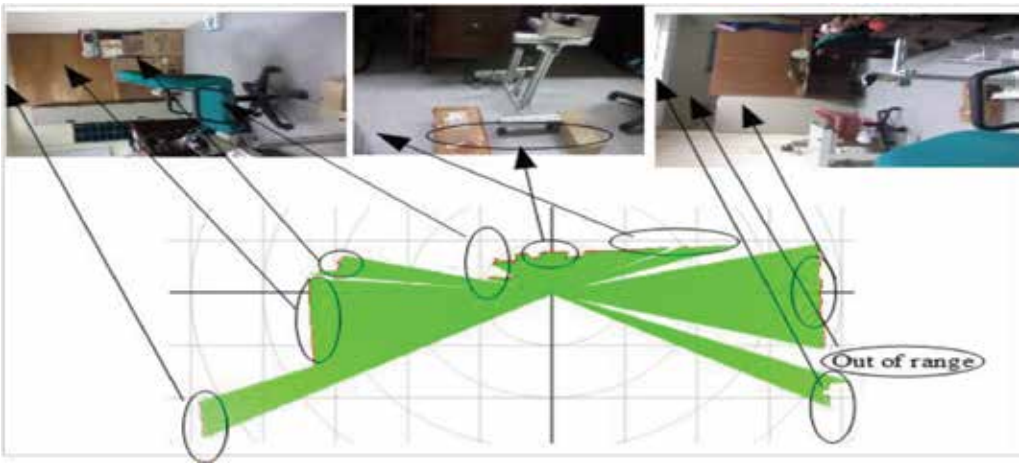


Figure 4. Laser measurements and the location of objects in its environment at the left, middle, and right side.

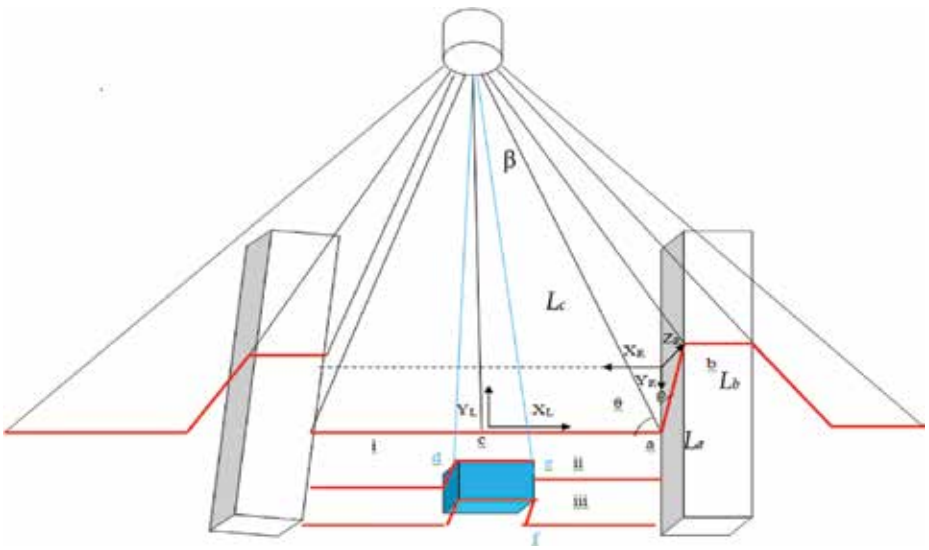


Figure 5. Principle of laser measurement calculation.

$$y_c - y_d = \frac{Z_{Rd}}{\sin(\rho)} \text{ and } x_d = -L_d \cos(\beta) \tag{5}$$

$$y_c - y_e = \frac{Z_{Re}}{\sin(\rho)} \text{ and } x_e = -L_e \cos(\beta), \tag{6}$$

The width of the obstacle can be calculated as:

$$W_{ob} = x_e - x_d \tag{7}$$

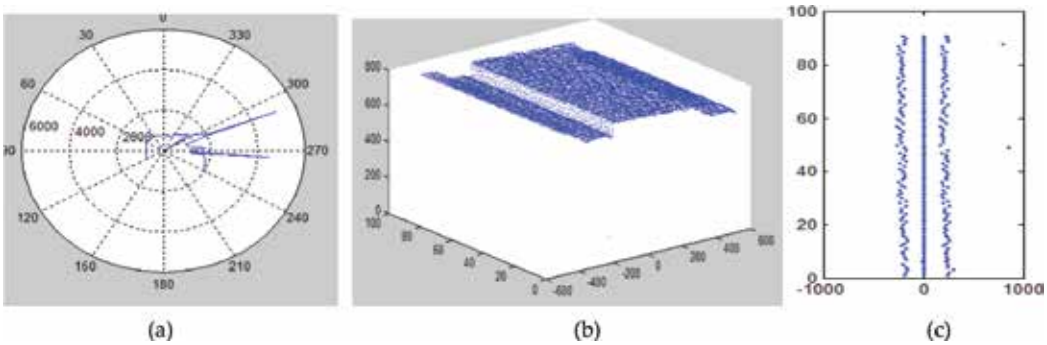


Figure 6. Laser measurements for the experimental setup, (a) one scan of laser measurement (mm), (b) road with curbs in 3D (mm), and (c) path generation (mm).

From previous calculation, one can define the parameters that will be used later for road discovering by LRF as shown in **Figure 5** as:

- Road fluctuations (height of objects with reference to the laser device) as follows:

$$r_{fn} = r_n \cos(\beta) \tag{8}$$

- Road width (side distance with reference to the laser device) as follows:

$$r_{wn} = r_n \sin(\beta) \tag{9}$$

r_n is the dimension of LRF signals for n-th LRF measurements.

- Curb threshold: rf_0 in $\beta = 0^\circ$ is used as reference and the other fluctuation measurements (r_{fn}) on the left and right side in the same scan are compared with this base line. If the deviation between the reference point and other measurement values exceeds the predefined threshold t_h, this point will be considered as a road curb. Otherwise, it will be considered as a road. This operation is repeated with all measurements as follows:

$$r_{fn} - r_{f0} \geq \pm t_h \tag{10}$$

The LRF driver for reading the data from USB connection in real-time is developed in MATLAB. It includes some functions to identify, configure and get data from the sensors as further detailed out in Appendix B. The algorithm for detecting the curbs of road based on the previous mentioned Eqs. (3)–(10) is developed and implemented in the road environments as shown in **Figure 6**. The LRF is the main sensor for the calculation of the robot path within the environments, which produces the results through using the LRF to generate the equations for path planning as shown in Sections 4.1 and 4.2.

3.3. Odometry

Two rotary encoders (*B106*) were used to estimate the position of robot, which are connected through dual brush card pins (*IFC-BH02*) to the two differential wheels. The full rotation of

http://dx.doi.org/10.5772/intechopen.79412

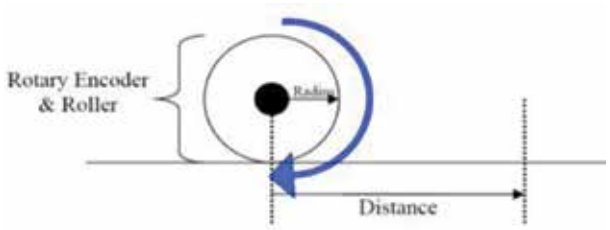


Figure 7. Calculation of the linear position from the rotary encoder.

encoders is 500 pulses/rotations, and the linear position of the wheels is calculated from the encoder rotation as shown in **Figure 7** using the following expression:

$$C = \frac{2\pi\, r\, P_{\text{cur}}}{P_{\text{fr}}} \tag{11}$$

C is the accumulative path of robot, r is the radius of wheel, P_{cur} is the number of pulses in the current position, and P_{fr} is the number of pulses for one full rotation of the encoder.

Two encoders were used in the proposed system, and the accumulative path will be calculated as an average value of both encoders as follows:

$$C_{av} = \frac{C_1 + C_2}{2} \tag{12}$$

4. Navigation system results and discussion

The proposed WMR is able to navigate autonomously on the road following and negotiates effectively the curvature of a standard roundabout. The robot is able to find the path starting from the predefined start pose until it reaches the pregoal position using the LS approach for the local map that was identified by the robot sensors such as camera, the LRF, and odometry measurements. Normally, the start and goal positions are determined using the DGPS system, and the robot uses this information to detect the direction of the roundabout exit. Since the proposed robot is navigating in a relatively small area (about 10–30 m), it is not useful to use GPS. Instead one can simply inform the robot about the roundabout exit direction, goal position before it starts to move. The goal position can be determined as how much distance the robot should travel after passing through the roundabout, for example, the goal is located at eastern part of the exit of the roundabout (i.e., 270°) with a distance equals to 20 m. The path that the robot will follow in the environment can be adjusted in the program of the navigation system as in middle, left or right. It is desired that the robot should navigate in the middle of the road for all the experiments. The velocity of the robot during navigation is adjusted in the range of 7–10 cm/s.

During navigation on the road, our Laser Simulator approach is used to detect the roundabout when it is present in the local map identified by the camera [27–29]. The sensor fusion including LRF and encoders is used to estimate the position of robot within the environments.

Figure 8. Image sequences processing where no roundabout can be detected using LS, (a) image from preprocessing and processing step, and (b) applying the LS (continues line in the middle).

Figure 9. Image sequences processing where roundabout is detected using LS, (a) image from preprocessing and processing step, and (b) applying the LS (discontinues line in the middle).

In general, two conditions are used to detect a roundabout; noncurbs detection on the left and right and discovering ellipse in the image as shown in **Figures 8** and **9**.

4.1. Road following

In the road following area, the camera, encoders, and laser are combined together to find the collision-free path. The camera is used for roundabout detection when it figures out using Laser Simulator. LRF is utilized for detecting the road curbs and localizing the robot in the environment. The encoder's measurements are used for estimating the current position of robot within environments. The generation of robot path is described in the following paragraphs.

4.2. Roundabout navigation results

The results show the capability of the roundabout path planning algorithm to find the right trajectory with a small deviation in comparison to the optimum path as shown in **Figure 10**.

http://dx.doi.org/10.5772/intechopen.79412

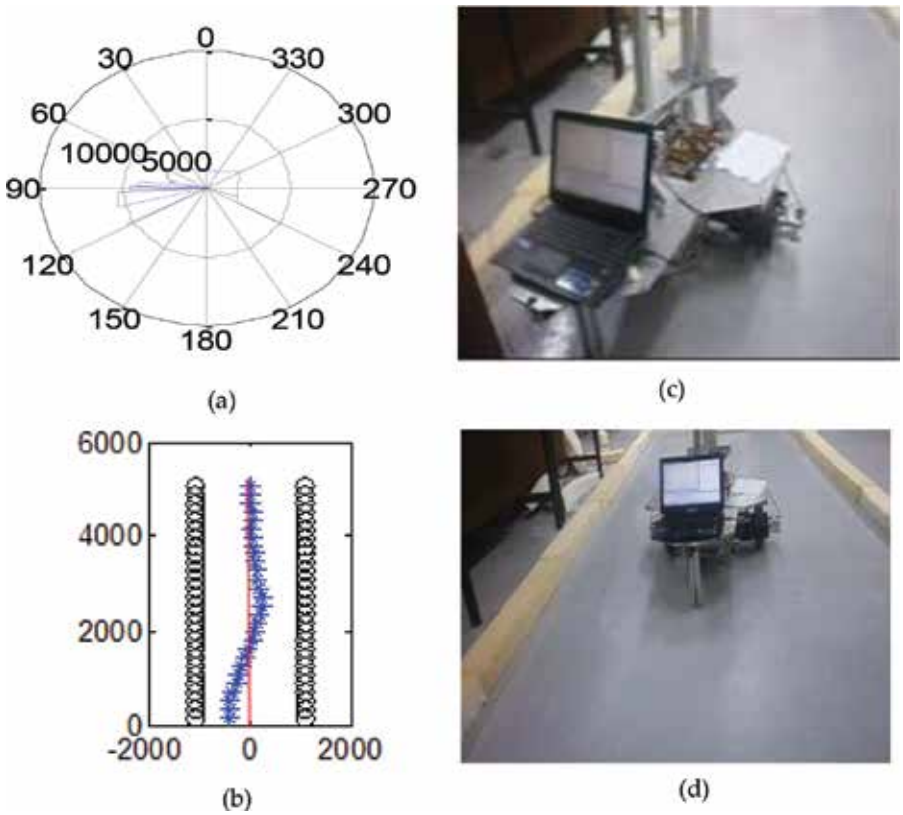


Figure 10. Correction of the robot path in the road following when the start location of the robot is near to left curb, (a) laser measurements, (b) path determination: the blue with (*) is the laser measurements, black with (o) is road curbs and roundabout, and red with (−) is the optimum path, (c) images of the robot in the beginning of movement, and (d) images of the robot when correcting its path.

Figure 11 shows how the camera, encoders, and laser range finder are used for path planning and execution from start to goal position. It also shows the local map built from camera sequence frames, where the developed view presents the curbs of the road environment of road in images. The camera's local map is determined for each image in the sequences of video frames as shown in **Figure 11(b)** and **(c)**, and the LS is applied to find the roundabout. **Figure 11(c)** shows the last image, where the roundabout is detected.

By applying the algorithm of cameras, laser range finder, and odometry described in Section 3, one can determine the path of robot as in **Figure 11(d)**. It is noticed that the part of roundabout in the entrance and exit areas is not occurred in the roundabout environment, due to that the laser still does not reach that area, but it is calculated based on a mixed algorithm of camera and laser range finder. The road following curbs are shown in **Figure 11(d)**, where the entrance to the roundabout is located on the left side, and the exit is located on the right. The path of the robot looked smooth especially with the indoor environments in comparison with the optimum path (continuous red line); however, there is a deviation in the outdoor measurements in

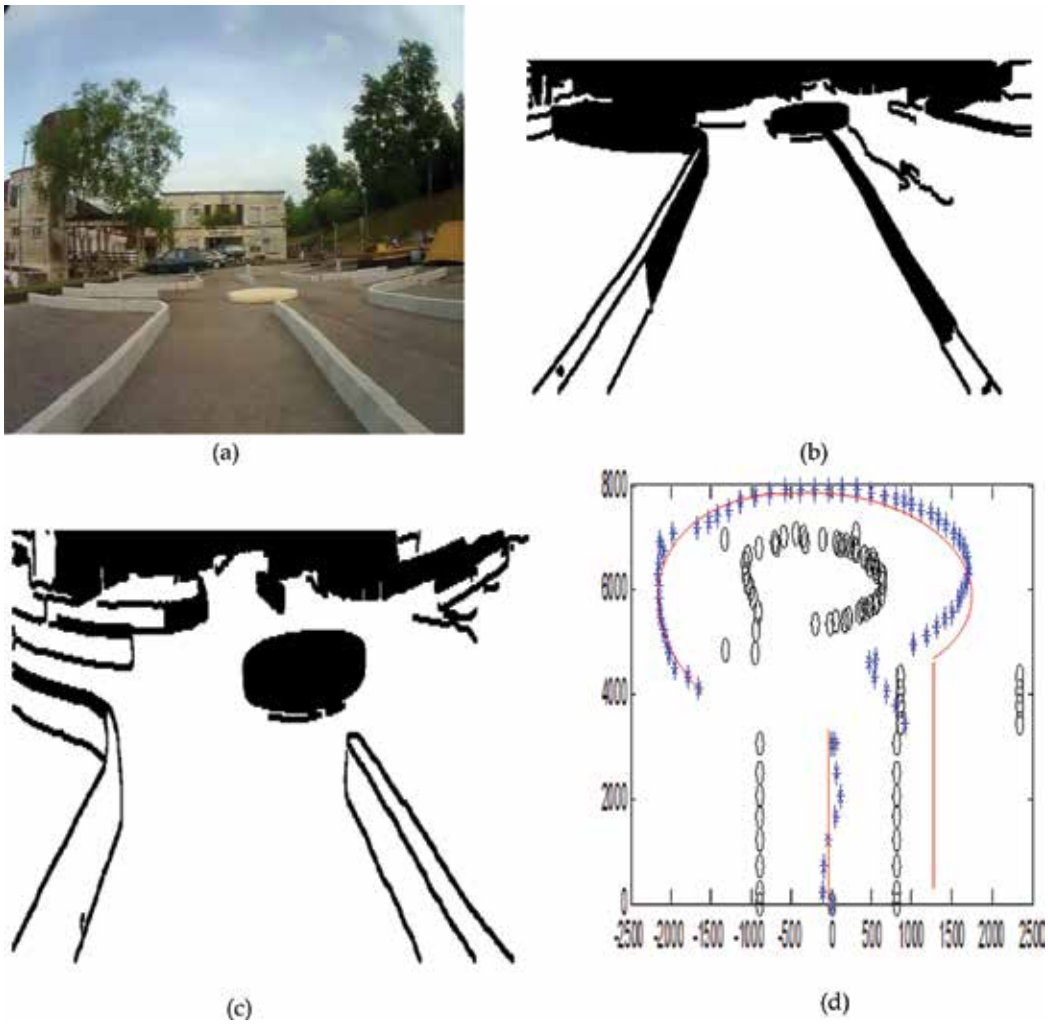


Figure 11. Camera sequence images when navigating in outdoor with 360° rotation, (a) image when start moving, (b) camera's local map when robot starts to move, (c) camera's local map when robot detects the roundabout, and (d) path of robot during navigation in a roundabout in (mm) with 360° rotation: blue with (*) is the path, black with (O) is for the road environments, and red with (−) for the optimum path.

the area of entrance and exit of roundabout due to the distance between the curbs and the roundabout center.

5. Conclusion

A practical implementation of the WMR online path navigation using laser simulator (LS) and sensor fusion in the road environments is presented in this thesis. The mobile robot is supposed to navigate on the roads in real time and reach the predetermined goal while

discovering and detecting the roundabout. A complete modeling and path determination of the roundabout intersection environments for autonomous mobile robot navigation is derived and presented in this chapter. The LS approach was used for road roundabout environment detection; whereas, the sensor fusion involves the laser range finder (LRF), and odometry was used to generate the path of the robot and localize it within its terrain. The local maps were built using both the camera and LRF to estimate the road border parameters such as road width, curbs, and roundabout in 2D. The algorithm of the path generation is fully derived within the local maps and subsequently implemented in two ways; first, considering the direct use of the sensor fusion data for path planning without applying localization and control algorithms and second, taking into account a complete model-based navigation including localization, path planning, and control schemes.

Author details

Mohammed A. H. Ali[1]* and Musa Mailah[2]

*Address all correspondence to: hashem@ump.edu.my

1 Faculty of Manufacturing Engineering, Universiti Malaysia Pahang, Pekan, Malaysia

2 Faculty of Mechanical Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia

References

[1] Ali MAH. Autonomous mobile robot navigation and control in the road following and roundabout environments incorporating laser range finder and vision system [PhD thesis]. UTM; 2014

[2] Ali MA, Mailah M, Yussof WA, Hamedon ZB, Yussof ZB, Majeed AP. Sensors fusion based online mapping and features extraction of mobile robot in the road following and roundabout. IOP Conference Series: Materials Science and Engineering. 2016;**114**(2016):1-12

[3] Miguel A. VIRTUOUS: Vision-based road transport-system for unmanned operation on urban-like scenarios. IEEE Transactions on Intelligent Transportation Systems. 2004;**5**(1): 69-83

[4] Matsushita Y, Miura J. On-line road boundary modeling with multiple sensory features, flexible road model and particle filter. Robotics and Autonomous Systems. 2011;**59**(1):274-284

[5] Seung H, Chi W, Sung C, Minyong P. Outdoor navigation of a mobile robot using differential GPS and curb detection. In: Proceedings of IEEE International Conference on Robotics and Automation; April 10–14. Roma, Italy: IEEE; 2007. pp. 3414-3419

[6] Kazunori O, Takashi T, Bunji S, Shoichi M, Shin Y. Outdoor navigation of a mobile robot between buildings based on DGPS and Odometry data fusion. In: Proceedings of IEEE

International Conference on Robotics and Automation; September 14–19. Taipei, Taiwan: IEEE; 2003. pp. 1978-1984

[7] Jose G, Eduardo N, Stephan B. Autonomous navigation and map building using laser range sensors in outdoor applications. Journal of Robotic Systems. 2000;**17**(10):565-583

[8] Donghoon L, Sukyung S, Kwangwoong Y, Jonghwan P, Hogil L. Sensor fusion localization system for outdoor mobile robot. In: ICROS-SICE International Joint Conference; August 18–21, 2009. Fukuoka, Japan: SICE; 2009. pp. 1384-1387

[9] Cristina C, Dolores B, Luis M. Compact modeling technique for outdoor navigation. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans. 2008; **38**(1):9-24

[10] Florian A, Franz A, Sven L, Lukas G, Joerg D. An unmanned helicopter for autonomous flights in urban terrain. In: German Workshop on Robotics; June 9–10. Braunschweig, Germany: Aerospace Center (DLR). pp. 1-11

[11] Panzieri S, Pascucci F, Ulivi G. An outdoor navigation system using GPS and inertial platform. In: Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics: IEEE/ASME; 2001. pp. 1346-1351

[12] Thrapp R, Westbrook C, Subramanian D. Robust localization algorithms for an autonomous campus tour guide. In: Proceedings of IEEE International Conference on Robotics and Automation. May 21–26, 2001. Seoul, Korea: IEEE; 2001. pp. 2065-2071

[13] Kazuyuki K, Cheok K, Kajiro W, Fumio M. Accurate differential global positioning system via fuzzy logic Kalman filter sensor fusion technique. IEEE Transactions on Industrial Electronics. 1998;**45**(3):510-528

[14] Cherif S, El-Najjar M, François C. Multi-sensor fusion method using dynamic Bayesian network for precise vehicle localization and road matching. In: Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence. October 29–31, 2007. Patras, Greece: IEEE; 2007. pp. 146-151

[15] Chao G, Michael S, John R. Towards autonomous wheelchair systems in urban environments. In: Howard A et al., editors. Field and Service Robotics. Vol. 7. Heidelberg-Berlin: Springer-Verlag; 2010. pp. 13-23

[16] Goldbeck J, Huertgen B, Ernsta S, Kelch L. Lane following combining vision and DGPS. Journal of Image and Vision Computing. 2000;**18**:425-433

[17] Nebot E, Sukkarieh S, Hugh D. Inertial navigation aided with GPS information. In: IEEE Conference on Mechatronics and Machine Vision in Practice. September 23–25, 1997. Toowoomba, Australia: IEEE; 1997. pp. 169-174

[18] Eric N, Jacques G, Tarbouchi M, Umar I, Aboelmagd, N. Enhanced mobile robot outdoor localization using INS/GPS integration. In: International Conference on Computer Engineering and Systems ICCES 2009. December 14–16, 2009. Cairo, Egypt: IEEE Xplore; 2009. pp. 127-132

[19] Campbel N, Pout M, Priestly M, Dagless E, Thomas BT. Autonomous road vehicle navigation. Engineering Applications of Artificial Intelligence. 1994;**7**(2):177-190

[20] Jing-Chuan W, Shiyu H, Zhang X, Weidong C. Mobile robot localization in outdoor environments based on infrared vision. In: International Conference on Intelligent Computing for Sustainable Energy and Environment LSMS/ ICSEE 2010. September 17–20, 2010. Wuxi, China: Springer-Verlag; 2010. pp. 33-41

[21] Locchi L, Kanolige K. Visually realistic mapping of a planar environment with stereo. In: Proceedings of Seventh International Symposium on Experimental Robotics (ISER). December 10–13, 2000. Honolulu, Hawaii: Springer; 2000. pp. 521-532

[22] Castellanos J, Martfinez M, Neira J, Tadus D. Simultaneous map building and localization for mobile robot: A multi-sensor fusion approach. In: Proceedings of IEEE International Conference on Robotics and Automation. May 16–20, 1998. Leuven, Belgium: IEEE; 1998. pp. 1244-1249

[23] Leonimer F, Jose F. Trajectory planning for non-holonomic mobile robot using extended Kalman filter. Mathematical Problems in Engineering. 2010;**2010**:979205, 22 pp

[24] Hong T, Christopher R, Tommy C, Michael S. Road detection and tracking for autonomous mobile robots. In: Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense, Sensing, Simulation and controls. April 1–5. Orlando, FL: SPIE; 2002. pp. 1-8

[25] Aneesh C, Yuta S. Road-crossing landmarks detection by outdoor mobile robots. Journal of Robotics and Mechatronics. 2010;**22**(6):708-717

[26] Labayrade R, Royere C, Gruyer D, Aubert D. Cooperative fusion for multi-obstacles detection with use of stereovision and laser scanner. Autonomous Robots. 2005;**19**(2):117-140

[27] Ali M, Mailah M, Tang HH. A novel approach for visibility search graph based path planning. In: 13th Int. Conf. Robot., Control Manuf. Syst.; Kuala-Lumpur, Malaysia, April 2–4, 2013. pp. 44-49

[28] Ali M, Mailah M, Tang HH. Implementation of laser simulator search graph for detection and path planning in roundabout environments. WSEAS Transactions on Signal Processing. 2014;**10**(2014):118-126

[29] Ali MAH, Yusoff WABW, Hamedon ZB, Yusssof ZBM, Mailah M. Global mobile robot path planning using laser simulator. In: 2nd IEEE International on Robotics and Manufacturing Automation (ROMA); Ipoh: Malaysia; 2016

Chapter 8

Path Tracking of a Wheeled Mobile Manipulator through Improved Localization and Calibration

Tao Song, Fengfeng (Jeff) Xi and Shuai Guo

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79598

Abstract

This chapter focuses on path tracking of a wheeled mobile manipulator designed for manufacturing processes such as drilling, riveting, or line drawing, which demand high accuracy. This problem can be solved by combining two approaches: improved localization and improved calibration. In the first approach, a full-scale kinematic equation is derived for calibration of each individual wheel's geometrical parameters, as opposed to traditionally treating them identical for all wheels. To avoid the singularity problem in computation, a predefined square path is used to quantify the errors used for calibration considering the movement in different directions. Both statistical method and interval analysis method are adopted and compared for estimation of the calibration parameters. In the second approach, a vision-based deviation rectification solution is presented to localize the system in the global frame through a number of artificial reflectors that are identified by an onboard laser scanner. An improved tracking and localization algorithm is developed to meet the high positional accuracy requirement, improve the system's repeatability in the traditional trilateral algorithm, and solve the problem of pose loss in path following. The developed methods have been verified and implemented on the mobile manipulators developed by Shanghai University.

Keywords: mobile manipulator, localization, tracking, path following

1. Introduction

Recently, mobile manipulators have been used in various industries including aerospace or indoor decoration engineering, which requires a large workspace [1–4]. The said mobile manipulator consists of an industrial manipulator mounted on a mobile platform to perform various manufacturing tasks such as drilling/riveting in aerospace industry or baseline

drawing in decoration engineering. Wheeled mobile platforms with Mecanum wheels that can easily navigate through crowded spaces due to their omnidirectionality with a zero turning radius are commonly used. Path tracking is one of the important issues for mobile manipulators, in particular for performing manufacturing tasks. This chapter addresses this issue from the aspect of localization and calibration.

Localization is a key functionality of mobile manipulator in order to track and determine its position around the environment [5]. Many methods are proposed to address this issue [6]. They can be divided into two categories: absolute localization and relative localization.

Absolute localization relies on detecting and recognizing different features in the environment to obtain the position and posture. The features can be normally divided into two types: artificial landmarks and natural landmarks. Compared with the natural landmarks, artificial landmarks have advantages of high recognition, which will lead to high accuracy. There is no cumulative error problem when a localization method based on artificial landmarks is used. The key challenge is to identify and extract the needed information from the raw data of landmarks. For the relative localization, dead reckoning and inertial navigation are commonly carried out to obtain the systems' position. It does not have to perceive the external environment, but the drift error accumulates over time.

Researchers have proposed several solutions, such as fuzzy reflector-based localization [7] and color reflector-based self-localization [8]. The main drawbacks of these two methods are that the anti-interference ability is poor and the computation is huge. To solve these problems pertaining to the localization method based on artificial landmarks, Madsen and Andersen [9] proposed a method using three reflectors and the triangulation principle. Betke and Gurvits [10] proposed a multichannel localization method with the three-dimensional localization principle and the least squares method. Because of the unavoidable errors in position and angle measurement of reflectors, the use of only a trilateral or triangular method will not achieve high accuracy [11]. Nevertheless, there are still many challenges for mobile manipulator working in industrial environments such as aerospace manufacturing or decoration engineering, which requires high maneuverability and high accuracy at the same time. The stationary industrial manipulator has high repeated localization accuracy, which the mobile manipulator cannot achieve. This chapter focuses on the improvement of path tracking of a mobile manipulator through enhanced localization combined with calibration. Calibration is required for the system kinematic model established based on nominal geometry parameter to improve motion accuracy. Muir and Neuman [12] proposed a kinematic error model for Mecanum wheeled platform and applied actuated inverse and sensed forward solutions to the kinematic control. Wang and Chang [13] carried out error analysis in terms of distribution among Mecanum wheels. Shimada et al. [14] presented a position-corrective feedback control method with vision system on Mecanum wheeled mobile platform. Qian et al. [15] conducted a more detailed analysis on the installation angle of rollers. An improved calibration method is presented in this chapter to improve the tracking accuracy of a mobile manipulator.

2. System modeling

The wheeled mobile manipulator, as shown in **Figure 1**, is built with a manipulator onto a wheeled mobile platform with four Mecanum wheels. This system aims to carry out fuselage

http://dx.doi.org/10.5772/intechopen.79598

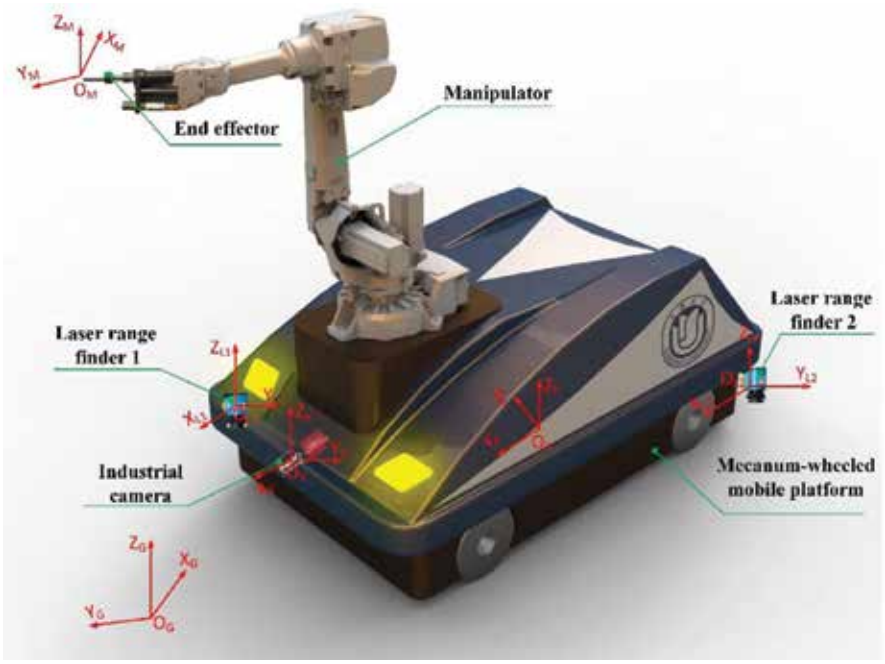


Figure 1. The wheeled mobile manipulator for fuselage drilling/riveting.

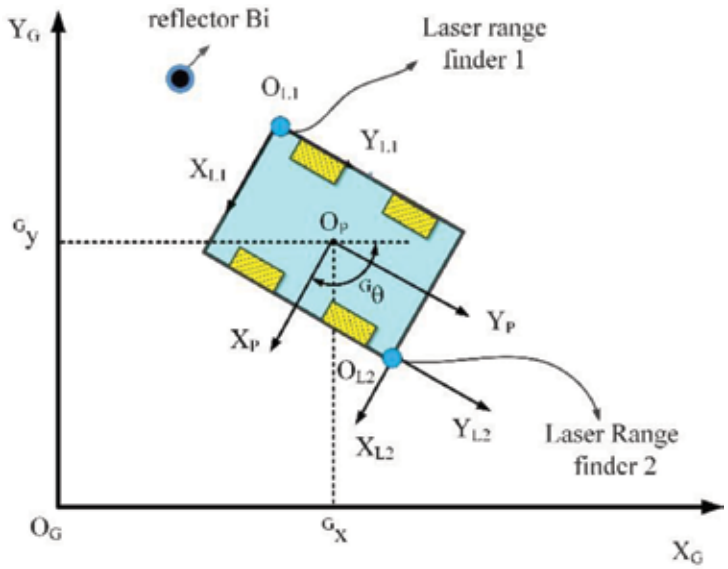


Figure 2. The position and posture of system.

drilling/riveting tasks at assembly stations in an adaptive and flexible way in the aerospace industry.

This system needs to localize in real time during machining process. The global frame $\{X_G, Y_G, Z_G\}$ is attached to the ground in the environment. The platform frame $\{X_P, Y_P, Z_P\}$ is

attached to the center of the mobile platform. The tool frame $\{X_M, Y_M, Z_M\}$ is attached to the tool of the manipulator. Two laser range finders are equipped and their frames $\{X_{L1}, Y_{L1}, Z_{L1}\}$ and $\{X_{L2}, Y_{L2}, Z_{L2}\}$ are attached to the left-front and right-rear of the mobile platform, respectively. The vision frame $\{X_V, Y_V, Z_V\}$ is attached to the industrial camera.

The position and posture of the system in the global frame can be defined as

$$^{G}\mathbf{P} = [^{G}x, \ ^{G}y, \ ^{G}\theta] \tag{1}$$

where ^{G}x and ^{G}y are the positions in two directions, respectively, and $^{G}\theta$ is the azimuth angle, as shown in **Figure 2**.

3. Accuracy analysis of a wheeled mobile manipulator

3.1. Problem formulation

Figure 3 shows the kinematic model of the mobile platform with Mecanum wheels. According to kinematic analysis [16], the motion equation can be obtained as

$$\mathbf{V} = \mathbf{D}_0\mathbf{W} \tag{2}$$

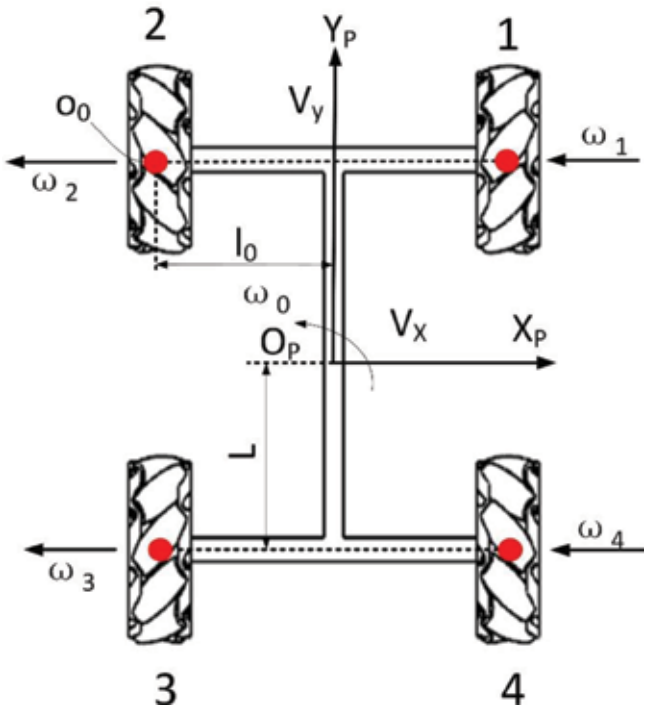


Figure 3. The kinematic model of a Mecanum wheeled mobile platform.

where $\mathbf{V} = [V_X, V_Y, \omega_0]^T$, $\mathbf{D_0} = \frac{1}{4}\begin{bmatrix} -R_0 & R_0 & -R_0 & R_0 \\ R_0 & R_0 & R_0 & R_0 \\ \frac{R_0}{l_0+L} & -\frac{R_0}{l_0+L} & -\frac{R_0}{l_0+L} & \frac{R_0}{l_0+L} \end{bmatrix}$, $\mathbf{W} = [\omega_1, \omega_2, \omega_3, \omega_4]^T$.

$[V_X, V_Y, \omega_0]^T$ is defined as the linear and angular speeds of the platform; L is the half distance from the front axle to the rear axle as shown in **Figure 3**; l_0 is the transverse distance from the wheel centers to the platform center line; R_0 is the radius of the Mecanum wheel; and ω_1, ω_2, ω_3, and ω_4 are angular velocities of the four wheels, respectively.

The Mecanum wheel and its roller are shown in **Figure 4**. The roller is fitted on the edge of the wheel at a certain angle (generally 45°) with its central axis. Each roller can rotate freely around the central axis. The wheel relies on the friction between the roller and the ground to move. The material of roller's outer rim is usually rubber, which will deform under pressure between ground and the wheel.

Figure 5 shows the distribution of roller deformation. F and T are the force and driving torque on the roller, respectively. The radiuses of wheels reduce differently under different pressures. The deformation zone and the position of the wheel center change with the action of driving torque and the shifting [17]. Furthermore, with such deformation, R_0 and l_0 will change.

As shown in **Figure 6**, based on the kinematic analysis and consideration on deformation of the roller, Eq. (2) can be revised as follows: [18].

$$\mathbf{V} = \mathbf{D}_1\mathbf{W} \tag{3}$$

where $\mathbf{D}_1 = \frac{1}{4}\begin{bmatrix} -R_1 & R_2 & -R_3 & R_4 \\ R_1 & R_2 & R_3 & R_4 \\ \frac{R_1}{l_1+L} & -\frac{R_2}{l_2+L} & -\frac{R_3}{l_3+L} & \frac{R_4}{l_4+L} \end{bmatrix}$; here $\mathbf{D}_1$ is defined by R_1, R_2, R_3, and R_4, which are individual radiuses of the four Mecanum wheels, respectively. In order to improve the system motion accuracy, the relative errors ΔR_i between R_i and R_0, and relative errors Δl_i between l_i and l_0 ($i = 1, 2, 3, 4$) should be obtained to revise matrix $\mathbf{D}_1$ (**Figure 6**).

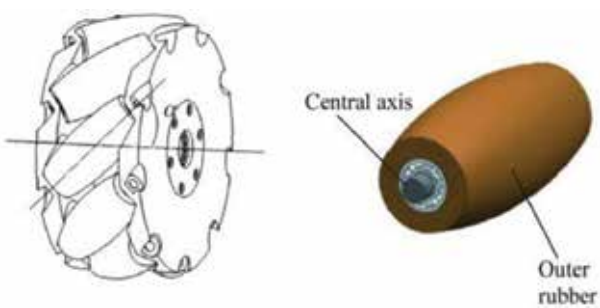


Figure 4. Mecanum wheel and roller.

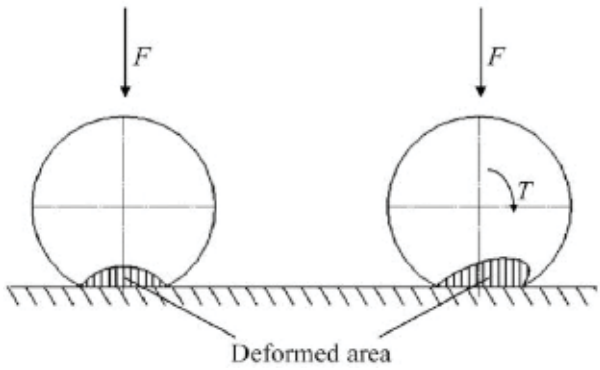


Figure 5. Roller deformation.

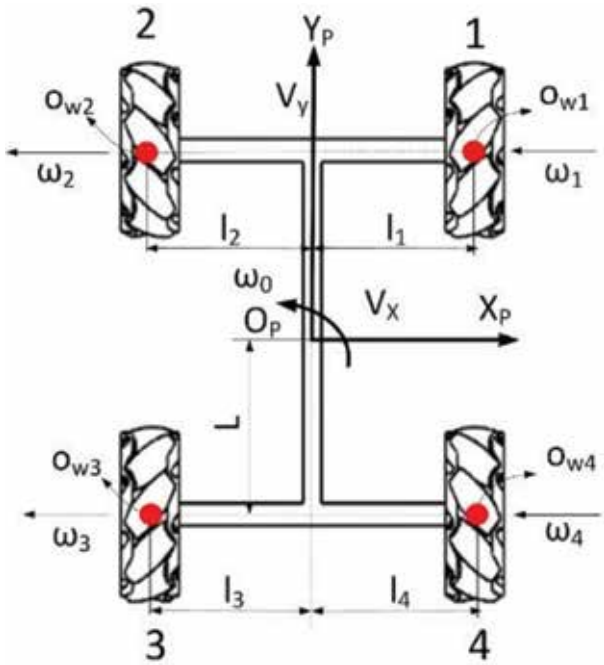


Figure 6. Motion model of the mobile platform.

3.2. Error modeling

By multiplying time t, Eq. (3) becomes a displacement equation as

$$\mathbf{X} = \mathbf{D_1}\boldsymbol{\theta} \tag{4}$$

where $\mathbf{X} = \mathbf{V}t$ is $[X, Y, \theta]^T$. $\boldsymbol{\theta} = \mathbf{W}t$ is $[\theta_1, \theta_2, \theta_3, \theta_4]^T$. Individual geometric parameters including l_1, l_2, l_3, l_4, R_1, R_2, R_3, and R_4 are variables in Eq. (4). The relative errors can be determined from

$$\mathbf{X} = \frac{1}{4}\begin{bmatrix} -R_1\theta_1 & R_2\theta_2 & -R_3\theta_3 & R_4\theta_4 \\ R_1\theta_1 & R_2\theta_2 & R_3\theta_3 & R_4\theta_4 \\ \dfrac{R_1}{l_1+L}\theta_1 & -\dfrac{R_2}{l_2+L}\theta_2 & -\dfrac{R_3}{l_3+L}\theta_3 & \dfrac{R_4}{l_4+L}\theta_4 \end{bmatrix} \tag{5}$$

leading to

$$\Delta\mathbf{X} = \mathbf{G}\Delta\mathbf{B} \tag{6}$$

where $\mathbf{\Delta X} = [\Delta X, \Delta Y, \Delta\theta]^T$, $\mathbf{\Delta B} = [\Delta R_1, \Delta R_2, \Delta R_3, \Delta R_4, \Delta l_1, \Delta l_2, \Delta l_3, \Delta l_4]^T$, and

$$\mathbf{G} = \begin{bmatrix} -\theta_1 & \theta_2 & -\theta_3 & \theta_4 & 0 & 0 & 0 & 0 \\ \theta_1 & \theta_2 & \theta_3 & \theta_4 & 0 & 0 & 0 & 0 \\ \dfrac{\theta_1}{M_1} & -\dfrac{\theta_2}{M_2} & -\dfrac{\theta_3}{M_3} & \dfrac{\theta_4}{M_4} & -\dfrac{R_1\theta_1}{{M_1}^2} & \dfrac{R_2\theta_2}{{M_2}^2} & \dfrac{R_3\theta_3}{{M_3}^2} & -\dfrac{R_4\theta_4}{{M_4}^2} \end{bmatrix}$$

$$R_1 = R_0 + \Delta R_1, l_1 = l_0 + \Delta l_1, R_2 = R_0 + \Delta R_2, l_2 = l_0 + \Delta l_2$$

$$R_3 = R_0 + \Delta R_3, l_3 = l_0 + \Delta l_3, R_4 = R_0 + \Delta R_4, l_4 = l_0 + \Delta l_4$$

With the least squares method, the geometric errors can be solved as

$$\Delta\mathbf{B} = \left(\mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{G}^T\Delta\mathbf{X} \tag{7}$$

ΔR_i is generally defined as the tolerance resulting from manufacturing or assembly. With the deformation of the roller and Δl_i limited to [–h, h] and [–j, j] (h = 3mm, j = 5mm) respectively, l_i and R_i can be defined as:

$$l_i = 2jr + (l_0 - j) \tag{8}$$

$$R_i = 2hr + (R_0 - h) \tag{9}$$

where l_i and R_i are in $[l_0 - j, l_0 + j]$ and $[R_0 - h, R_0 + h]$, respectively, and r is a random number between 0 and 1. The angular speeds of all wheels are set to 20rad/s, and the time t is set to 1 s.

The rank of matrix **G** in Eq. (7) is generally 3, so **G** is a full rank matrix and **ΔB** can be obtained. The following steps can be carried out. First, the displacement error measurement is analyzed. In order to obtain **ΔB**, it is needed to obtain the displacement error matrix **ΔX** first.

The system is moved in the Y_G direction and stopped every 2 s to obtain its position and posture $\left[{}^GX_k, {}^GY_k, {}^G\theta_k\right]$. The principle of measurement is shown in **Figure 7** and $\Delta\mathbf{X}$ can be calculated as

$$\mathbf{\Delta X} = \begin{bmatrix} \left({}^GX_1 + {}^GX_2 + {}^GX_3 + {}^GX_4\right)/10 - X_T \\ \left({}^GY_1 + {}^GY_2 + {}^GY_3 + {}^GY_4\right)/10 - Y_T \\ \left({}^G\theta_1 + {}^G\theta_2 + {}^G\theta_3 + {}^G\theta_4\right)/10 - \theta_T \end{bmatrix} \tag{10}$$

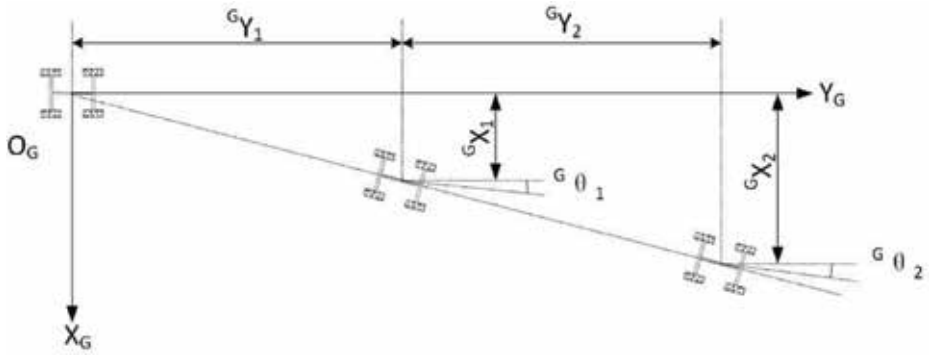


Figure 7. The principle of measuring displacement.

According to Eq. (2), X_T, Y_T, and θ_T are theoretical values which can be obtained as follows: $X_T = V_x t$, $Y_T = V_y t$, $\theta_T = 0$. Finally, through the experiment, the displacement errors can be obtained.

$$\Delta \mathbf{X} = \begin{bmatrix} 3\text{mm} \\ 1\text{mm} \\ 0^{\circ} \end{bmatrix} \tag{11}$$

Substituting Eq. (11) in to Eq. (7), $\Delta\mathbf{B}$ can be determined.

The Monte Carlo analysis is applied to deal with 50 samples. While $\Delta\mathbf{X}$ has been given in Eq. (11), the corresponding $\Delta\mathbf{B}$ should also satisfy its own tolerance, i.e., $-3 \le \Delta R_i \le 3$ and $-5 \le \Delta L_i \le 5$.

The results are given in **Figure 8**. These data are averaged to form a new result: $\Delta\mathbf{B} = [-0.541, 1.237, -0.605, 1.055, 0.458, 0.683, -0.048, -0.683]^T$. A process capability index is then used to estimate the value of $\Delta\mathbf{B}$. Process capability refers to the ability of a process to meet the required quality. It is a measure of the minimum fluctuation of the internal consistency of the process itself in the most stable state. When the process is stable, the quality characteristic value of the product is in the range $[\mu - 3\sigma, \mu + 3\sigma]$, where μ is the ensemble average of the quality characteristic value of the product, and σ is the standard deviation of the quality characteristic value of the product.

There are two kinds of situations for which the process capability index has to be solved. First, $\mu = M$ as shown in **Figure 9**: U_{SL} is the maximum error of quality and L_{sl} is the minimum error of quality; $M = (U_{SL} + L_{SL})/2$; μ is the average value of process machining; and C_p indicates the process capability index, and $C_p = (U_{SL} - L_{SL})/6\sigma$.

Second, $\mu \neq M$ as shown in **Figure 10**: C_{pk} indicates the process capability index, and $C_{pk} = (U_{SL} - L_{SL})/6\sigma - |M - \mu|/3\sigma$. Only process capability index greater than 1 is valid. As $-3 \le \Delta R_i \le 3$ and $-5 \le \Delta L_i \le 5$, $M = 0$ and $\mu \neq M$, the following results can be obtained:

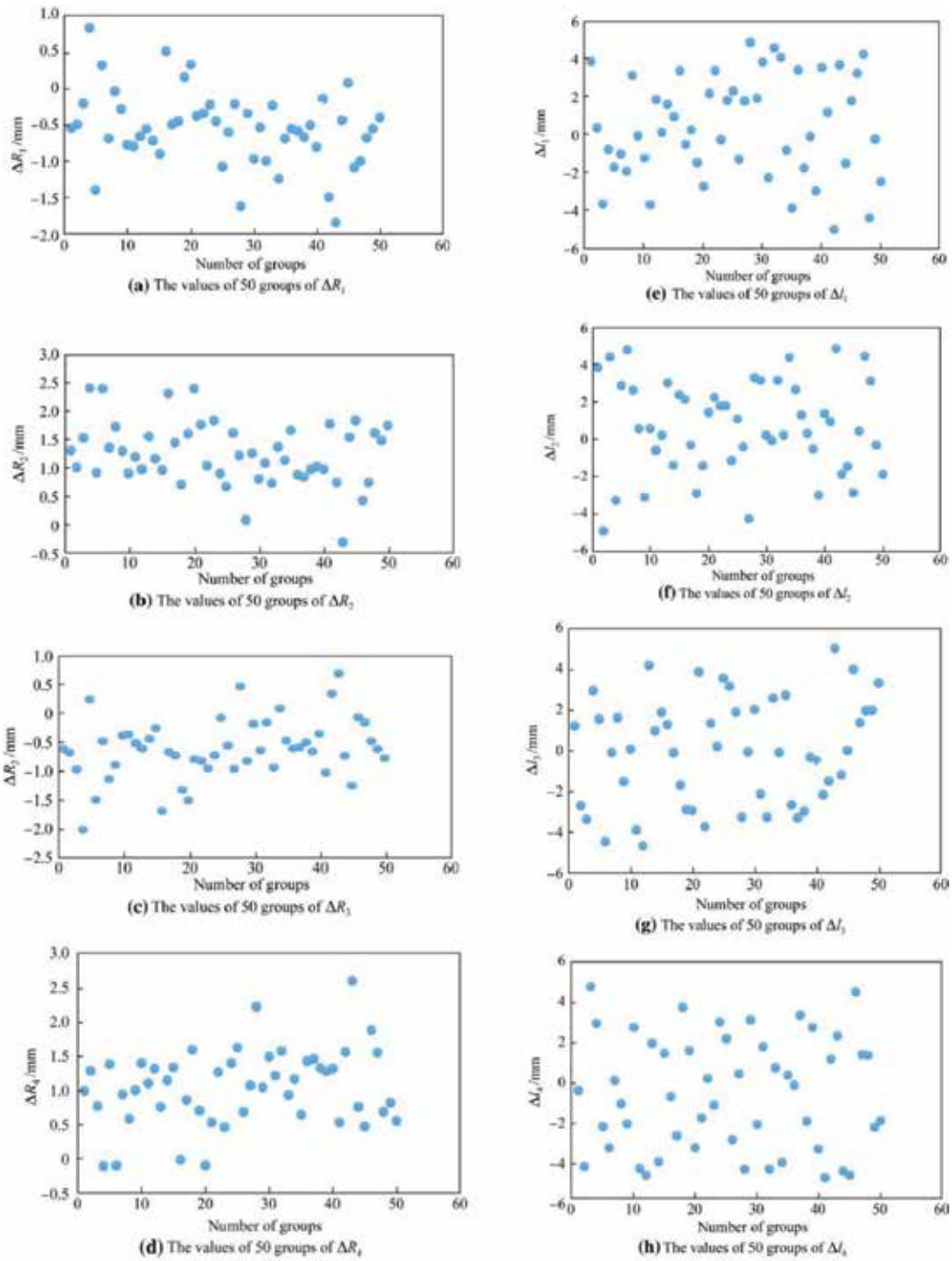


Figure 8. The values obtained with the Monte Carlo method.

$C_{pk}(\Delta R_1) = 1.58$, $C_{pk}(\Delta l_1) = 0.57$, $C_{pk}(\Delta R_2) = 1.06$, $C_{pk}(\Delta l_2) = 0.58$, $C_{pk}(\Delta R_3) = 1.54$, $C_{pk}(\Delta l_3) = 0.63$, $C_{pk}(\Delta R_4) = 1.17$, and $C_{pk}(\Delta l_4) = 0.5$.

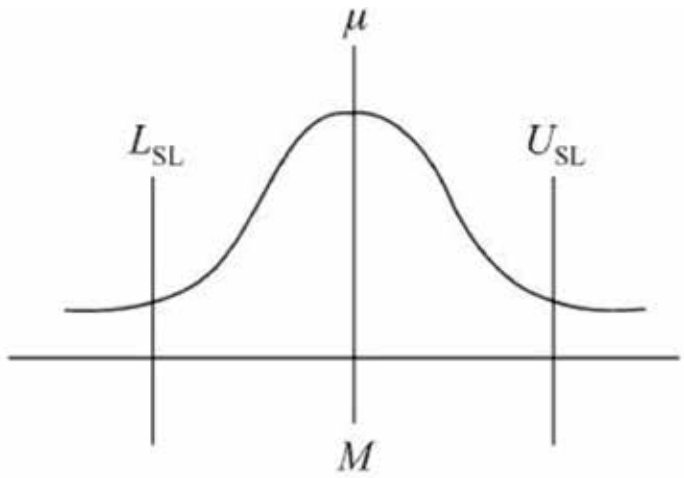


Figure 9. $\mu = M$.

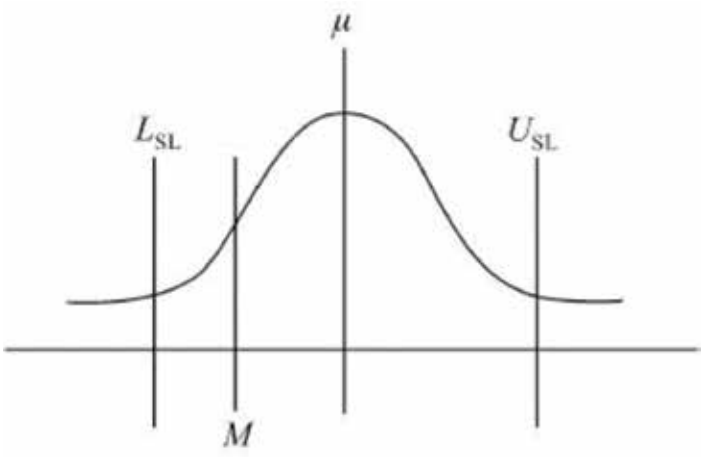


Figure 10. $\mu \neq M$.

The values of ΔR_1, ΔR_2, ΔR_3, and ΔR_4 are closer to the real values than the values of Δl_1, Δl_2, Δl_3, and Δl_4. Now, taking $\Delta \mathbf{B_1}(\mathbf{a}) = [-0.541, 1.237, -0.605, 1.055, 0, 0, 0, 0]^T$, $\Delta \mathbf{B_1}(\mathbf{b}) = [0, 0, 0, 0, 0.458, 0.683, -0.048, -0.683]^T$ in Eq. (5), it can be seen that $\Delta \mathbf{X_1} = [-0.4119, 4, -0.0003]^T$, $\Delta \mathbf{X_1}(\mathbf{b}) = [0, 0, -0.0003]^T$. According to the change rate relative to Eq. (10), the influence of ΔR_1, ΔR_2, ΔR_3, and ΔR_4 is much bigger than that of Δl_1, Δl_2, Δl_3, and Δl_4. Although the values of Δl_1, Δl_2, Δl_3, and Δl_4 are not accurate enough for the real values, $\Delta \mathbf{B_1}$ is valid.

The interval analysis is also carried out. In Eq. (6), the value of $\mathbf{G}$ is uncertain because of the change of ΔR_i and Δl_i in $[-3, 3]$ and $[-5, 5]$, respectively. Now, the increment of ΔR_i and Δl_i is set

to 1 mm; there are 7 values of ΔR_i, and 11 values of Δl_i, so there are $7^4 \times 11^4$ groups of combinations of matrix **G**. Take all the combinations of matrix **G** to Eq. (6) to obtain **ΔB** and exclude the cases for which ΔR_i and Δl_i are not in $[-3, 3]$ and $[-5, 5]$, respectively. Thus, 277 groups of **ΔB** can be obtained. As shown in **Figure 11**, the average values of ΔR_1, ΔR_2, ΔR_3, ΔR_4, Δl_1, Δl_2, Δl_3, and Δl_4, lead to a new $\mathbf{\Delta B_2} = [-0.233, 1.467, -0.932, 0.824, 0.2, -0.213, -0.068, -0.039]^T$, which is close to $\mathbf{\Delta B_1}$.

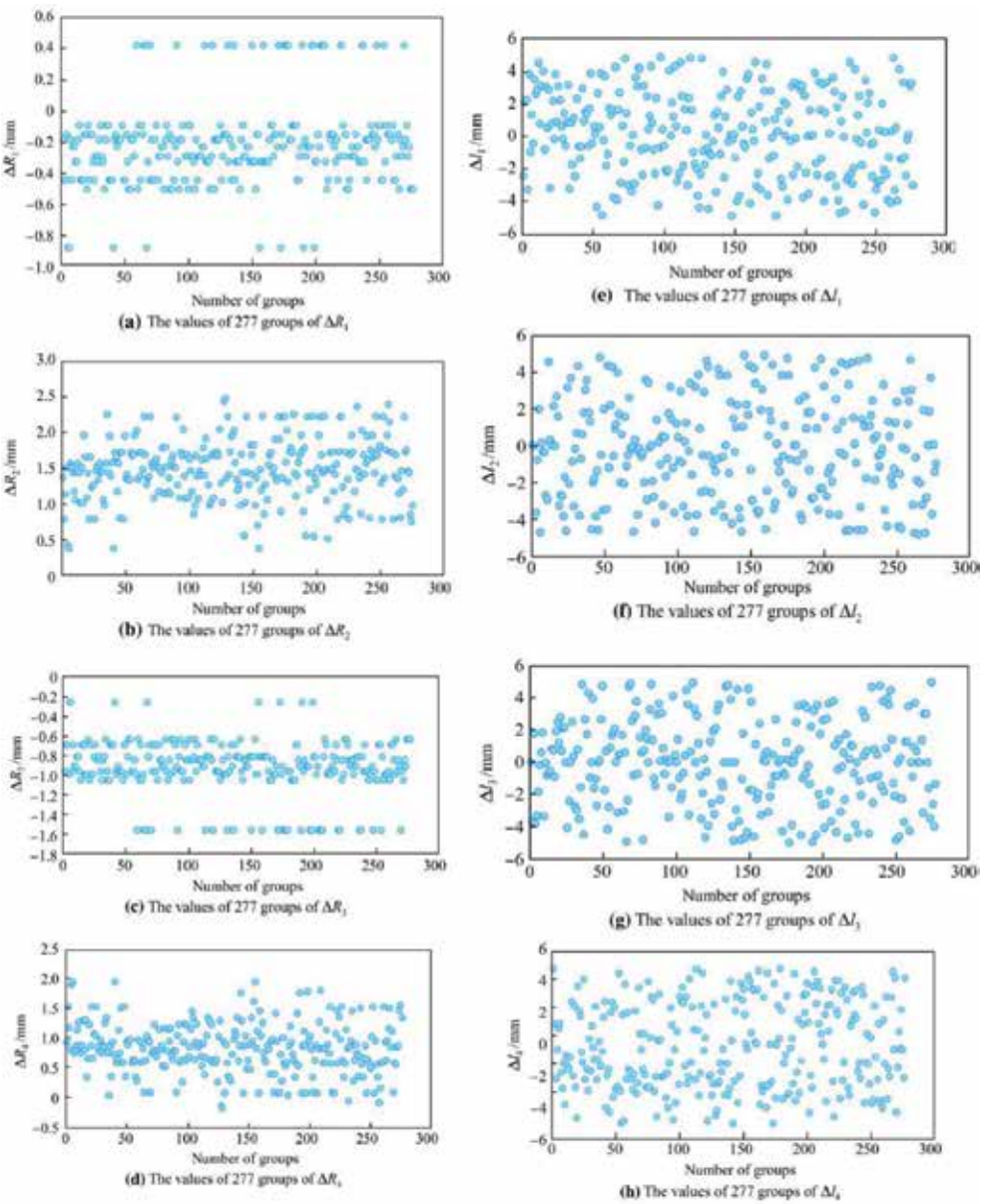


Figure 11. The values obtained in interval analysis.

The same method is used to solve the process capability index of $\mathbf{\Delta B_2}$. One obtains: $C_{pk}(\Delta R_1) = 3.34$, $C_{pk}(\Delta l_1) = 0.68$, $C_{pk}(\Delta R_2) = 1.22$, $C_{pk}(\Delta l_2) = 0.59$, $C_{pk}(\Delta R_3) = 2.5$, $C_{pk}(\Delta l_3) = 0.63$, $C_{pk}(R_4) = 1.73$, and $C_{pk}(\Delta l_4) = 0.6$. The values of ΔR_1, ΔR_2, ΔR_3, and ΔR_4 in $\mathbf{\Delta B_2}$ are closer to the real ones than the values of Δl_1, Δl_2, Δl_3, and Δl_4 in $\mathbf{\Delta B_2}$. As the influence of ΔR_1, ΔR_2, ΔR_3, and ΔR_4 is bigger than that of Δl_1, Δl_2, Δl_3, and Δl_4, $\mathbf{\Delta B_2}$ is valid.

The result is verified as well. $\mathbf{\Delta B_1}$ is used to revise the parameters of $\mathbf{D_1}$ in Eq. (2) as

$$\mathbf{D}_1 = \frac{1}{4}\begin{bmatrix} -186.959 & 188.737 & -186.895 & 188.555 \\ 186.959 & 188.737 & 186.895 & 188.555 \\ 0.1369 & -0.1382 & -0.1369 & 0.1382 \end{bmatrix}$$

By setting two sets of four wheel speeds $\mathbf{W}_1 = [10\pi/21, 10\pi/21, 10\pi/21, 10\pi/21]^{\mathrm{T}}$ and $\mathbf{W}_2 = [4\pi/7, 4\pi/7, 4\pi/7, 4\pi/7]^{\mathrm{T}}$, the displacement errors can be computed as $\mathbf{S}_1 = (\mathbf{D}_1 - \mathbf{D}_0)\mathbf{W}_2 t$

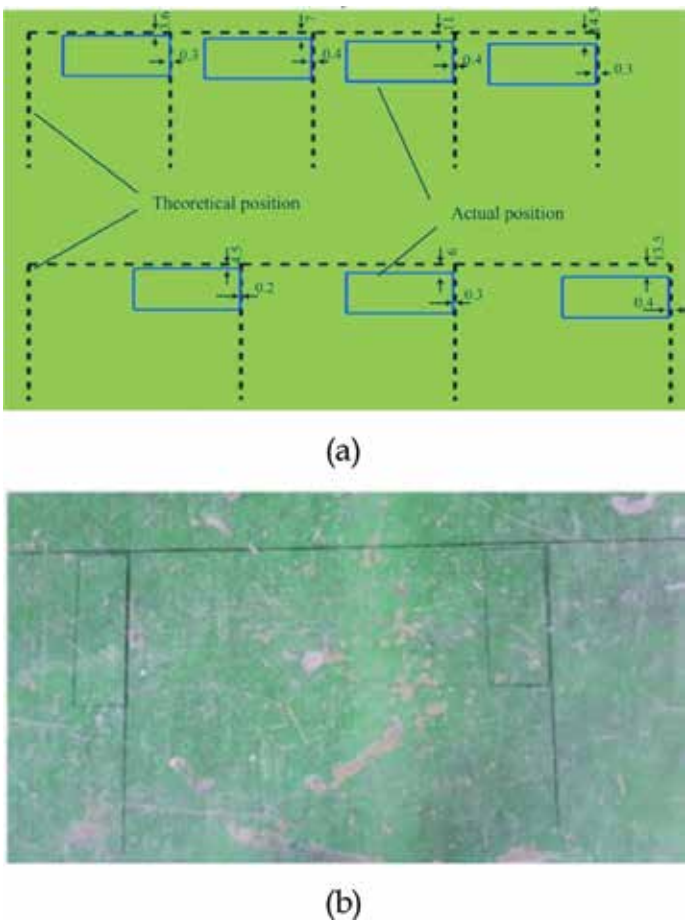


Figure 12. (a) Schematic diagram of measuring and (b) the actual measuring.

and $\mathbf{S}_2 = (\mathbf{D}_2 - \mathbf{D}_0)\mathbf{W}_2 t$. The results of the two correction methods are almost identical in theory.

The experiment with four movements is shown in **Figure 12(a)**, while the actual movement is shown in **Figure 12(b)**. It can be found that $\mathbf{S}_1$ and $\mathbf{S}_2$ are close to the measured displacement errors, so both $\Delta\mathbf{B_1}$ and $\Delta\mathbf{B_2}$ are satisfied to revise the matrix $\mathbf{D_1}$.

4. Localization

4.1. System configuration

The localization component of mobile manipulator includes two laser range finders and a number of reflectors that are of cylindrical shape placed in the environment. Each reflector is covered by a reflective tape with high reflectivity (**Figure 13**).

4.2. Dynamic tracking

To achieve accurate localization, the system should have the ability to perceive external information through extracting the reflector features. In this research, as shown in **Figure 13**, there are n reflectors, and the feature of each reflector $B_i (i = 1, 2, \ldots, n)$ is extracted from the raw data. The feature extraction algorithm consists of three steps: (i) filtering and clustering, (ii) identification, and (iii) feature extraction.

The first step is filtering and clustering. The raw data obtained by each finder are a set of discrete data sequences $\{\{(\gamma, \varnothing), \lambda\}_i \mid i = 1, 2, \ldots, n\}$. γ is the distance from the target point to the finder. $\varnothing$ is the polar angle. λ_i is the intensity value of the ith data point. In the process of data analysis, the outlier points that are contaminated by noise will be filtered out.

The density of the collected data points is proportional to the distance from the target point to the laser range finder. To improve the efficiency of the feature extraction process, an adaptive clustering method is adopted as given by Eq. (12). Unless the distance between two data points is less than the threshold δ, these data points are clustered for one reflector.

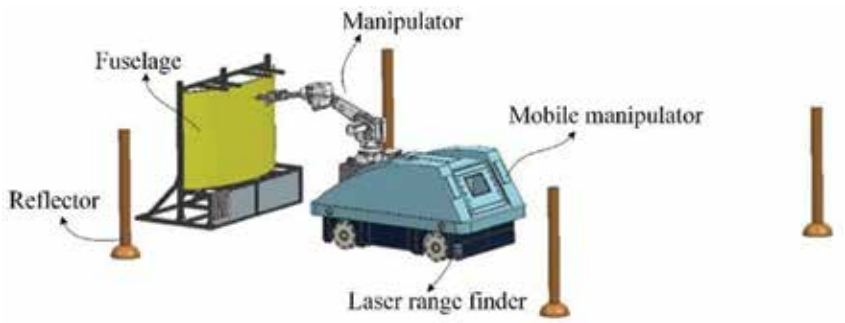


Figure 13. Localization system for the mobile manipulator.

$$\delta = \gamma_{i-1}\left(\left(\sin \Delta\varnothing\right)/\left(\sin\left(\beta - \Delta\varnothing\right)\right)\right) + 3\sigma_\gamma \tag{12}$$

where γ_{i-1} is the distance value of the $(i-1)$th data point, $\Delta\varnothing$ is the angle resolution of the laser range finder, β is an auxiliary constant parameter, and σ_γ is the measurement error. The values of parameters σ_γ and β are given as 0.01 m and 10°, respectively.

The second step is identification. After clustering, the data set can be correlated to each observable reflector. Each reflector data set is then used to calculate the position of the reflector in the laser range finder frame [19]. Let λ_δ be the reflected intensity threshold and $[D - D_\delta,\ D + D_\delta]$ be the diameter range of a reflector, where D is a nominal reflector diameter and D_δ is the tolerance. The values of λ_δ and D_δ are selected based on the actual situation. Considering that $\mathbf{W_c}$ represents a set after clustering, i.e., $\mathbf{W}_c = \left\{\{(\gamma, \varnothing), \lambda\}_i | i = m, \ldots, n\right\}$, for each reflector, the measured diameter D_c is calculated as $D_c = \sqrt{\gamma_n^2 + \gamma_m^2 - 2\gamma_n\gamma_m \cos\left(\varnothing_n - \varnothing_m\right)}$, where $(\gamma, \varnothing)_m$ and $(\gamma, \varnothing)_n$ are the beginning and end data of a reflector set, respectively. When the set $\mathbf{W_c}$ satisfies the following two conditions

$$\begin{cases} \lambda_{max} \geq \lambda_\delta \\ D_c \in [D_{min}, D_{max}] \end{cases} \tag{13}$$

then the set $\mathbf{W}_c$ is established for all reflectors.

The third step is feature extraction. The feature extraction algorithm of a reflector extracts its central position $\left(\gamma_{L,B}, \beta_{L,B}\right)$ in the laser range finder frame $\{X_L, Y_L, Z_L\}$, where $\gamma_{L,B}$ and $\beta_{L,B}$ are the distance and azimuth of each reflector, respectively. In the process of extracting the feature of the circular reflector, only a small portion of the entire circle was scanned with noise, so the accuracy of the fitting result would be low if a general least square circle fitting method is used. Since the radius of the reflector is known, the known radius is used as a constraint to improve the accuracy.

First, the value of $\beta_{L,B}$ is obtained from $(n - m)$ consecutive data points of the reflector set

$${}^L\beta_B = \frac{1}{(n-m)}\sum_{i=m}^{n} {}^L\beta_i \tag{14}$$

As shown in **Figure 14**, M_i represents the ith data and $\overline{O_{L2}B}$ is a line between the reflector center B and the laser range finder center O_{L2}; the projected angle of the line $\overline{O_{L2}B}$ in the laser range finder frame is ${}^L\beta_B$, while the angle between the line $\overline{O_{L2}B}$ and the line $\overline{O_{L2}M_i}$ is θ_i. The distance from M_i to B is approximately the radius of the reflector, i.e., $\left|\overline{BM_i}\right| = D/2$.

$$\theta_i = \varnothing_i - {}^L\beta_B \tag{15}$$

$$\left|\overline{O_{L2}B}\right|_i = \left|\overline{O_{L2}M_i}\right|\cos|\theta_i| + \left|\overline{BM_i}\right|\cos\left(\arcsin\left(\left(2\cdot\left|\overline{O_{L2}M_i}\right|\sin\theta_i\right)/D\right)\right) \tag{16}$$

$${}^L\gamma_B = \frac{1}{n-m}\sum_{i=m}^{n}\left|\overline{O_{L2}B}\right|_i,\ \ i = 1, 2, \cdots m \tag{17}$$

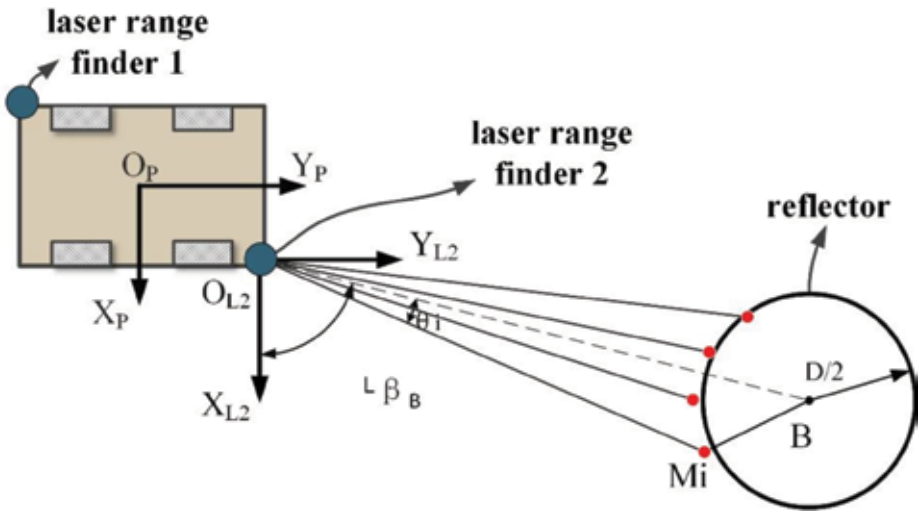


Figure 14. Extraction of the center of the cylindrical reflector.

Furthermore, as mentioned before, the position of the finder in the platform frame is $\left({}^{P}x_{L2}, {}^{P}y_{L2}\right)$. The position of reflector center B $\left({}^{P}\gamma_B, {}^{P}\beta_B\right)$ in the platform frame can be expressed as

$$\begin{cases} {}^{P}\gamma_B = \sqrt{\left({}^{P}x_{L2} + {}^{P}\gamma_B \cos {}^{P}\beta_B\right)^2 + \left({}^{P}y_{L2} + {}^{P}\gamma_B \sin {}^{P}\beta_B\right)^2} \\ {}^{P}\beta_B = \arctan({}^{P}y_{L2} - {}^{P}\gamma_B \sin {}^{P}\beta_B)/\left({}^{P}x_{L2} - {}^{P}\gamma_B \cos {}^{P}\beta_B\right) \end{cases} \tag{18}$$

The optimal triangulation localization algorithm based on angle measurement is carried out. A number of experiments were carried out on the reflector feature extraction algorithm before proposing a localization algorithm for the system. A single reflector was placed in the measuring range of the finder. The finder performed repeated scanning when it was placed at difference places. Two different feature extraction algorithms were tested to calculate the position of the reflector, and the results are shown in **Figure 15**. Algorithm A is the feature extraction algorithm proposed in this chapter, while algorithm B is the least square circle fitting method without a radius constraint. Apparently, the former one yields a better result.

After extracting the reflector center, the positions of all the reflectors $G = \left\{ \left({}^{L}\gamma_B {}^{L}\beta_B\right)_i | i = 1, 2, \ldots, n \right\}$ can be obtained and used to calculate the distance measurement range $R\gamma$ and the angler measurement range R_β of the reflector, as given below:

$$R_\gamma = \left({}^{L}\gamma_B\right)_{max} - \left({}^{L}\gamma_B\right)_{min} \tag{19}$$

$$R_\beta = \left(\left(\left({}^{L}\beta_B\right)_{max} - \left({}^{L}\beta_B\right)_{min} \right) \pi \cdot \frac{1}{n} \sum_{i=1}^{n} {}^{L}\gamma_B \right) /180 \tag{20}$$

where $\left({}^{L}\gamma_B\right)_{max} \in G$, $\left({}^{L}\gamma_B\right)_{min} \in G$, $\left({}^{L}\beta_B\right)_{max} \in G$, $\left({}^{L}\beta_B\right)_{min} \in G$.

It can be seen from **Figure 16** that the angle measurement accuracy is better than the distance measurement accuracy. Therefore, based on these results, this chapter proposes an optimal trilateral localization algorithm based on angle measurement. The idea is to use the azimuth

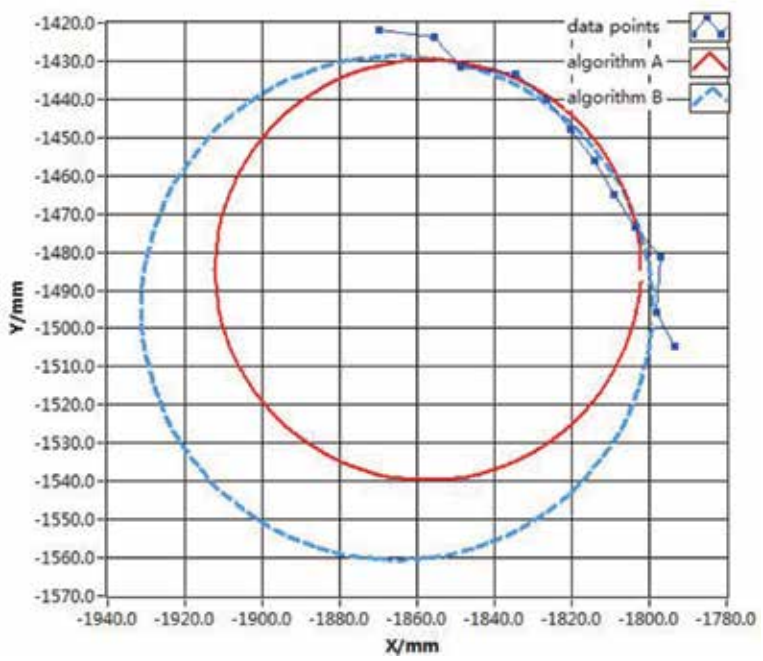


Figure 15. Effect diagram using different feature extraction algorithms.

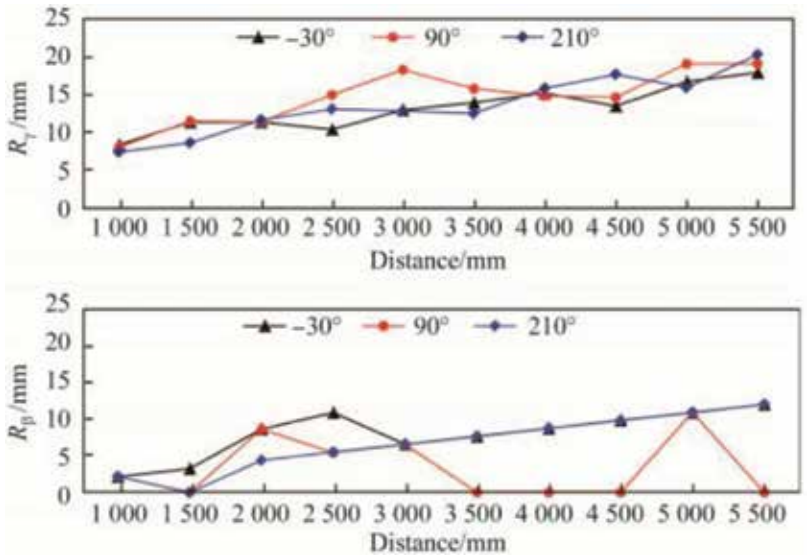


Figure 16. The position of single reflector in the laser range finder frame.

http://dx.doi.org/10.5772/intechopen.79598

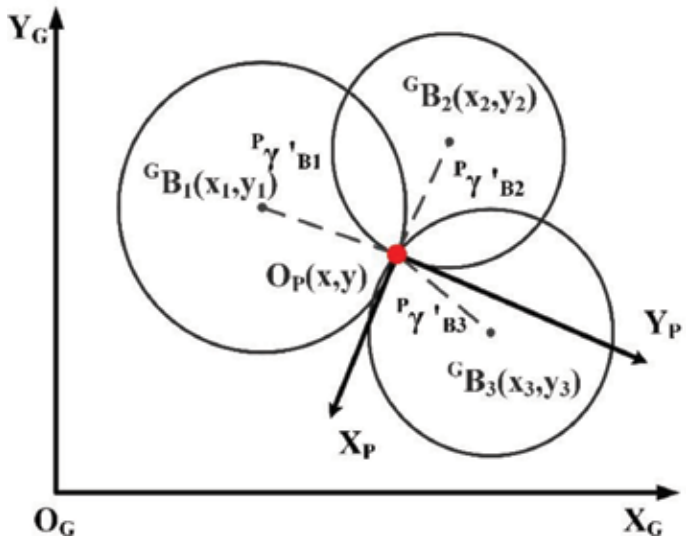


Figure 17. Schematic diagram of the localization algorithm.

angle β_b of the reflector in the platform frame to find the optima γ_b by the cosine theorem. The global pose of the mobile manipulator is then calculated based on the triangular method. The algorithm details are given as follows.

First, it is assumed that the finder can cover at least three reflectors at each position. After feature extraction, the positions of three reflectors B_1, B_2, and B_3 are calculated as $\left({}^{P}\gamma_{B_1}, {}^{P}\beta_{B_1}\right)$; $\left({}^{P}\gamma_{R,B_2}, {}^{P}\beta_{R,B_2}\right)$; and $\left({}^{P}\gamma_{R,B_3}, {}^{P}\beta_{R,B_3}\right)$, respectively. In the global frame, these positions can also be obtained as ${}^{G}B_1\,(x_1, y_1)$; ${}^{G}B_2(x_2, y_2)$; and ${}^{G}B_3(x_3, y_3)$. Since they are measured from the same finder, three circles must intersect at the same point ${}^{G}O_{MP}$, and the value of ${}^{G}O_{MP}$ $({}^{G}x_{MP}\ {}^{G}y_{MP})$, represents the position of the mobile platform, as shown in **Figure 17**.

According to the cosine theorem, the relations between the above variables can be expressed by the following equations:

$$\begin{cases} {}^{P}\gamma_{B_{1a}}^2 + {}^{P}\gamma_{B_{2a}}^2 - 2{}^{P}\gamma_{B_{1a}}\gamma_{B_{2a}}\cos\left({}^{P}\beta_{B1} - {}^{P}\beta_{B2}\right) = (x_1 - x_2)^2 + (y_1 - y_2)^2 \\ {}^{P}\gamma_{B_{2a}}^2 + {}^{P}\gamma_{B_{3a}}^2 - 2{}^{P}\gamma_{B_{2a}}\gamma_{B_{3a}}\cos\left({}^{P}\beta_{R} - {}^{P}\beta_{B3}\right) = (x_2 - x_3)^2 + (y_2 - y_3)^2 \\ {}^{P}\gamma_{B_{1a}}^2 + {}^{P}\gamma_{B_{3a}}^2 - 2{}^{P}\gamma_{B_{1a}}{}^{P}\gamma_{B_{3a}}\cos\left({}^{P}\beta_{B1} - {}^{P}\beta_{B3}\right) = (x_1 - x_3)^2 + (y_1 - y_3)^2 \end{cases} \tag{21}$$

In Eq. (21), the known parameters ${}^{P}\beta_{B1}$, ${}^{P}\beta_{B2}$, and ${}^{P}\beta_{B3}$ are used to solve ${}^{P}\gamma_{B_{1a}}$, ${}^{P}\gamma_{B_{2a}}$, and ${}^{P}\gamma_{B_{3a}}$. Since the above equations are nonlinear, the steepest descent method is adopted. The three circle equations can be expressed as

$$\begin{cases} \left(x_1 - {}^{G}x_{MP}\right)^2 + \left(y_1 - {}^{G}y_{MP}\right)^2 = {}^{P}\gamma_{B_{1a}}^2 \\ \left(x_2 - {}^{G}x_{MP}\right)^2 + \left(y_2 - {}^{G}y_{MP}\right)^2 = {}^{P}\gamma_{B_{2a}}^2 \\ \left(x_3 - {}^{G}x_{MP}\right)^2 + \left(y_3 - {}^{G}y_{MP}\right)^2 = {}^{P}\gamma_{B_{3a}}^2 \end{cases} \tag{22}$$

The position ${}^{G}O_{MP}$ $\left({}^{G}x_{MP}\ {}^{G}y_{MP}\right)$ of the system can be obtained in the global frame by solving the above equations. Since the actual position of the reflector in the global environment deviates from the theoretical position, these circles may not intersect at the same point. In order to minimize the difference between the calculated position of the system and the actual position, the least squares estimation principle is applied. Assuming that the coordinate of the reflector is ${}^{G}B_i$ $\left(x_i, y_i\right)(i = 1, 2, \ldots, n)$, n is the number of reflectors detected by laser range finder. The position value ${}^{G}O_{MP}$ $\left({}^{G}x_{MP}\ {}^{G}y_{MP}\right)$ of the system is calculated as

$$\left({}^{G}x_{MP}\ {}^{G}y_{MP}\right)^{T} = \left(\mathbf{A}^{T}\mathbf{A}\right)^{-1}\mathbf{A}^{T}\mathbf{b} \tag{23}$$

where

$$\mathbf{A} = \begin{bmatrix} 2(x_1 - x_n) & \cdots & 2\left(y_1 - y_n\right) \\ \vdots & \cdots & \vdots \\ 2(x_{n-1} - x_n) & \cdots & 2\left(y_{n-1} - y_n\right) \end{bmatrix} \tag{24}$$

$$\mathbf{b} = \begin{bmatrix} x_1^2 - x_n^2 + y_1^2 - y_n^2 + {}^{P}\gamma_{B_{2a}}^2 - {}^{P}\gamma_{B_{1a}}^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 + {}^{P}\gamma_{B_{2a}}^2 - {}^{P}\gamma_{B_{1a}}^2 \end{bmatrix} \tag{25}$$

The posture of the system also includes an azimuth angle ${}^{G}\theta$ in the global frame. First, ${}^{G}\theta_i$ is obtained from the ith reflector as

$${}^{G}\theta_i = \arctan\left(\left(y_i - {}^{G}y_{MP}\right)/\left(x_i - {}^{G}x_{MP}\right)\right) - {}^{P}\beta_{B_i} \tag{26}$$

The azimuth angle ${}^{G}\theta$ of the system is the averaged value from all the reflectors, as in

$${}^{G}\theta = \frac{1}{n}\sum_{i=1}^{n} {}^{G}\theta_i \tag{27}$$

The dynamic tracking algorithm is then carried out. Localization of the system is based on landmarks. The system needs to meet the following two conditions to complete real-time localization:

i. The system requires real-time localization of the reflector in the environment.

ii. The localization system correctly matches the real-time observed reflectors.

During the matching process, the reflectors observed in real time are made to correspond to all reflectors in the last moment in the environment one by one to extract the effective reflectors [20]. The localization can be achieved.

During localization, owing to the fact that some of the reflectors are obscured by obstacles or confused with a highly reflective metal surface object, the loss of position of the system is observed. To address the above problems, a dynamic tracking algorithm is proposed as shown in **Figure 18**.

After placing n reflectors in the global environment, the system actually observes q reflectors at the tth moment, and the coordinate value of the ith reflector in the platform frame is $\mathbf{M}_{t,i}\left(\gamma_{t,B_i},\beta_{t,B_i}\right)(0\ll i\ll q)$. The sampling time of the laser range finder is 0.1 s. Therefore, the theoretical position value $\mathbf{M}_{t-1,j-1}\left(\gamma_{t-1,B_j},\beta_{t-1,B_j}\right)$ of the jth reflector in the platform frame can be deduced by the position $O_{t-1,R}\left(x_{t-1},y_{t-1}\right)$ of the system at the $(t-1)$th moment and the position ${}^{G}O_j\left(x_j,y_j\right)$ of the jth reflector in the global environment.

If the difference between the theoretical value $\mathbf{M}_{t-1,j}\left(\gamma_{t-1,B_j},\beta_{t-1,B_j}\right)$ and the observed value $\mathbf{M}_{t,i}\left(\gamma_{t,B_i},\beta_{t,B_i}\right)$ is less than η, then the ith observed reflector is an effective reflector and is considered matched with the reference reflector B_j as follows:

$$\left|\sqrt{\left(r_{t-1,B_j}\cos\beta_{t-1,B_j}-\gamma_{t,B_j}\cos\beta_{t,B_i}\right)^2+\left(\gamma_{t-1,B_j}\sin\beta_{t-1,B_j}-\gamma_{t,B_i}\sin\beta_{t-1,B_j}\right)^2}\right|\le\eta \qquad (28)$$

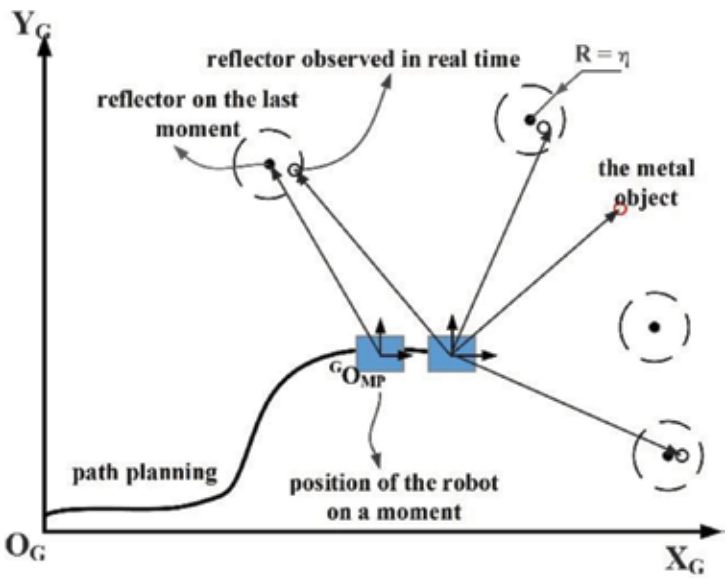


Figure 18. Schematic diagram of the dynamic tracking algorithm.

Therefore, a set $\mathbf{A}$ of effective reflectors at the tth moment can be obtained, and $\mathbf{A} = \left\{\mathbf{M}_{t,o}\left(\gamma_{t,B_o}\beta_{t,B_o}\right), \ldots, \mathbf{M}_{t,i}\left(\gamma_{t,B_o}, \beta_{t,B_i}\right) | i = 0, 1, \ldots, q\right\}$. Taking the mobile manipulator's moving speed into account, the value of η depends on the actual situation. Through the use of the optimal triangulation localization algorithm based on angle measurement, the pose $O_{t,R}\left(x_t, y_t\right)$ of the mobile manipulator can be calculated at the tth moment.

4.3. Experimental verification

The experimental data are obtained by the LMS 100 laser range finder with a scanning range of 270° and an angular resolution of 0.25°. The experimental platform is shown in **Figure 19**. The

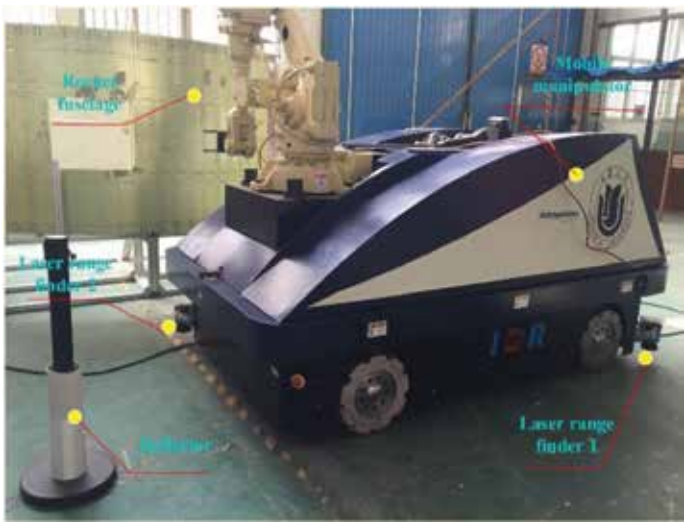


Figure 19. Experimental platform.

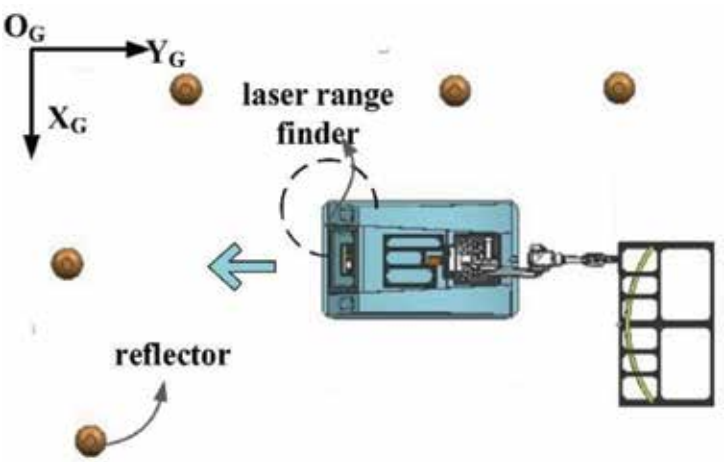


Figure 20. Experimental environment.

outside of the reflector is wrapped by reflective tape. In the experimental environment, five reflectors are placed around the system. Their global coordinate values are (0,995); (0, 0); (0, 1774); (2905, −2449); and (3956, −2032), and the unit is mm, as shown in **Figure 20**.

The optimal triangulation method based on angle measurement is used for validation by the repeatability of the system. In the stationary state of the system, the environment is scanned by

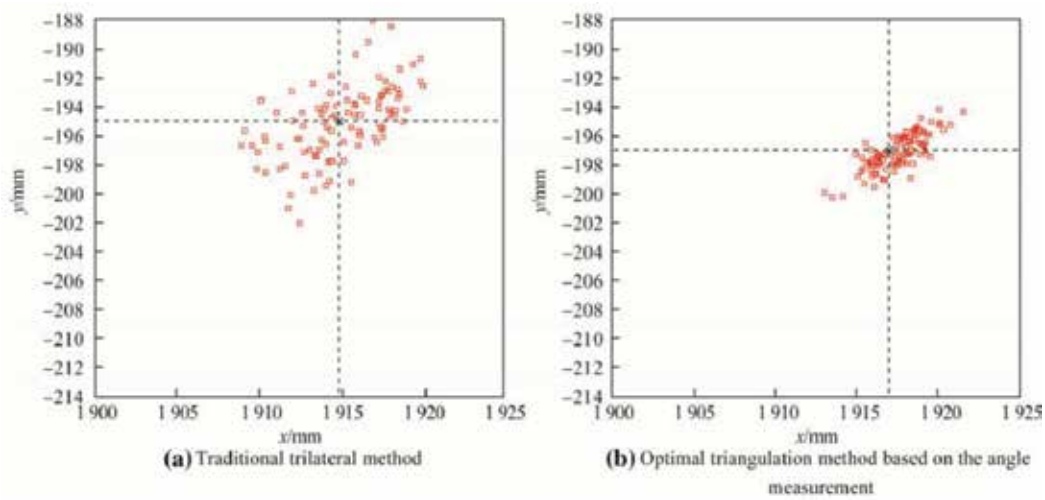


Figure 21. Repeatability localization of the system at the same location.

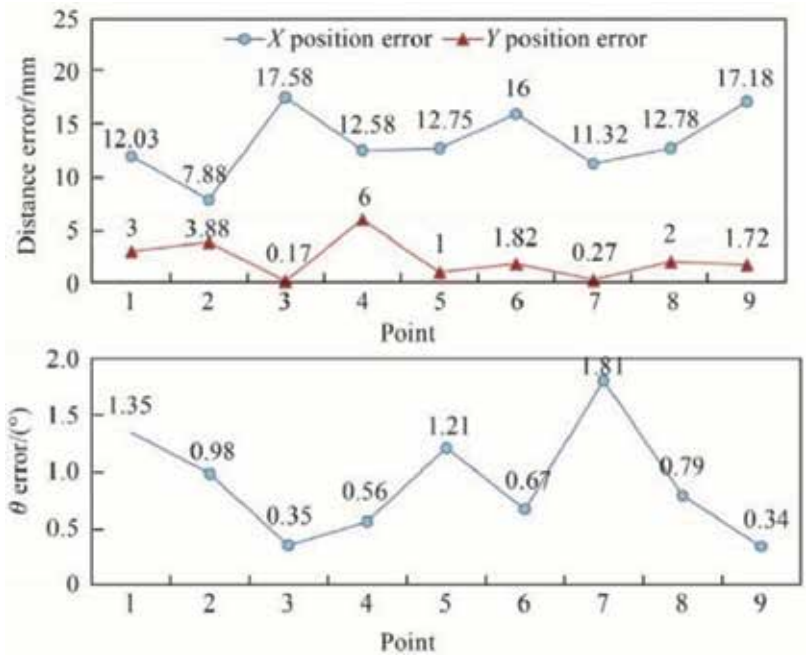


Figure 22. System localization error.

finders. Each result is indicated by a red dot in **Figure 21**. The repeatability obtained by the trilateral method is nearly 18 mm, while the repeatability of the optimal method is only 9 mm. It can be shown that the optimal method is better than the traditional method.

The mobile manipulator moves in the direction of the arrow in **Figure 19**, and each time the system moves a certain distance, the localization system will perform an experiment, i.e., it will use the left rear finder to calculate the current position. An average of 30 samples is taken for each experiment.

Figure 22 shows the results of static localization accuracy. The maximum distance error is 18 mm and the maximum angle error is 2°, which satisfies the localization requirement of the system.

The mobile manipulator moves in the designated route, and it needs to constantly calculate and record its own position in the moving process. As shown in **Figure 23**, the trajectory of the moving system based on the localization method is smoother.

This chapter demonstrates the feasibility of a tracking and localization algorithm for mobile manipulators. The following conclusions can be drawn from this study: (i) In the detection of a reflector in the laser range finder frame, the angle repeatability of the reflector is better than that of the distance repeatability based on the feature extraction algorithm; (ii) The repeatability localization accuracy using the optimal triangulation method based on the angle measurement is nearly 9 mm, which is better than that of the trilateral method; (iii) The localization error of the system is 18 mm, which satisfies the localization requirement of system. Improvements in the location method based on reflectors, such as optimizing the layout of reflectors and the map of reflectors selection strategy for localization, are still needed.

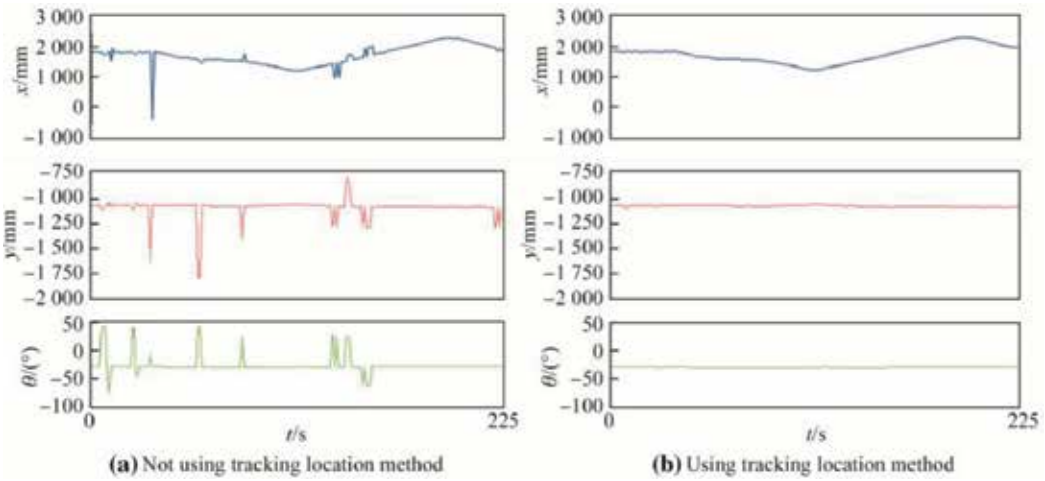


Figure 23. Tracking results.

5. Summary

In this chapter, through analyzing the roller deformation of the Mecanum wheel, the changed parameters of the motion equation of mobile system are found. The relative variation of the

parameters in the motion equation of the Mecanum motion platform is solved by Monte Carlo analysis and interval analysis. Using the relative variation of the parameters to revise the motion equation, the displacement errors in different spaces in theory are solved for and compared with the measured displacement errors. From the comparison, both the methods are found to satisfy the system's requirement. Then, the feasibility of a tracking and locating algorithm for mobile manipulator is demonstrated. The following conclusions can be drawn from this study: (i) In the detection of a reflector in the laser range finder frame, the angle repeatability of the reflector is better than that of the distance repeatability based on the feature extraction algorithm; (ii) The repeatability localization accuracy using the optimal triangulation method based on the angle measurement is nearly 9 mm, which is better than that of the trilateral method; (iii) The localization error of the system is 18 mm, which satisfies the localization requirement of system. Improvements in the localization method based on reflectors, such as optimizing the layout of reflectors and the map of reflectors selection strategy for localization, are still needed.

The method in this chapter is also used in the research of MoMaCo (Mobile manipulator in Construction), which can draw baseline for architectural decoration engineering as shown in **Figure 24**. The application result also verified the effectiveness of the method.

Figure 24. MoMaCo (mobile manipulator in construction).

Author details

Tao Song[1,2]*, Fengfeng (Jeff) Xi[3] and Shuai Guo[1,2]

*Address all correspondence to: songtao43467226@shu.edu.cn

1 Key Laboratory of Intelligent Manufacturing and Robotics, School of Mechatronics Engineering and Automation of Shanghai University, Shanghai, PR China

2 Shanghai Robot Industrial Technology Research Institute, Shanghai, PR China

3 Department of Aerospace Engineering, Ryerson University, Toronto, Ontario, Canada

References

[1] Guo S, Tao S, Xi F(J), Mohamed RP. Tip-over stability analysis for a wheeled mobile manipulator. Journal of Dynamic Systems, Measurement, and Control. 2017;**139**:054501

[2] Tao S, Xi F(J), Guo S, Lin Y. Optimization of a mobile platform for a wheeled manipulator. Journal of Mechanisms and Robotics. 2016;**8**:061007

[3] Tao S, Xi F(J), Guo S, Tu X, Li X. Slip analysis for a wheeled mobile manipulator. Journal of Dynamic Systems, Measurement, and Control. 2017;**140**(2):021005-021005-12

[4] Guo S, Jin Y, Bao S, et al. Accuracy analysis of omnidirectional mobile manipulator with Mecanum wheels. Advances in Manufacturing. 2016;**4**:363

[5] Guo S, Fang T-T, Tao S, Xi F-F, Wei B-G. Tracking and localization for omnidirectional mobile industrial robot using reflectors. Advances in Manufacturing. 2018;**6**(1):118-125

[6] Larsson U, Forsberg J, Wernersson A. Mobile robot localization: Integrating measurements from a time-of-flight laser. IEEE Transactions on Industrial Electronics. 1996;**43**(3):422-431

[7] Buschka P, Saffiotti A, Wasik Z. Fuzzy landmark-based localization for a legged robot. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2; 2000. pp. 1205-1210

[8] Jang G, Lee S, Kweon I. Color landmark based self-localization for indoor mobile robots. In: IEEE International Conference on Robotics & Automation; 2002. pp. 1037-1042

[9] Madsen CB, Andersen CS. Optimal reflector selection for triangulation of robot position. Robotics and Autonomous Systems. 1998;**23**(4):277-292

[10] Betke M, Gurvits L. Mobile localization using landmarks. IEEE Transactions on Robotics and Automation. 1997;**13**(2):251-263

[11] Kai OA, Tomatis N, Jensen BT, et al. Multi-laser range finder on-the-fly localization: Precision and reliability for applications. Robotics and Autonomous Systems. 2001;**34**(2–3): 131-143

[12] Muir PF, Neuman CP. Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. In: IEEE International Conference on Robotics and Automation; 1987. pp. 1772-1778

[13] Wang Y, Chang D. Preferably of characteristics and layout structure of Mecanum comprehensive system. Journal of Mechanical Engineering. 2009;**45**(5):307-310

[14] Shimada A, Yajima S, Viboonchaicheep P, et al. Mecanum-wheel vehicle systems based on position corrective control. In: IEEE Annual Conference on Industrial Electronics Society; 2005

[15] Qian J, Wang X, Xia G. Velocity rectification for mecanum wheeled omni-direction robot. Machine Tool and Manufacturing Technology. 2010;**11**:42-45. http://en.cnki.com.cn/Article_en/CJFDTotal-ZJYC201011014.htm

[16] Huang L et al. Design and analysis of a four-wheel omnidirectional mobile robot. In: 2nd International Conference of Autonomous Robots and Agents; 2004

[17] Song JB, Byun KS. Design and control of a four-wheeled omnidirectional mobile robot with steerable omnidirectional wheels. Journal of Robotic Systems. 2004;**21**(4):193-208

[18] Asama H et al. Development of an omni-directional mobile robot with 3 DOF decoupling drive mechanism. In: IEEE International Conference on Robotics and Automation

[19] Liu YM. Three-dimensional optical positioning method research [dissertation]. Shenyang University of Technology; 2003

[20] Jia W, Zhou W, Kaiser J. Efficient algorithm for mobile multicast using any cast group. IEE Proceedings: Communications. 2001;**148**(1):14-18

Chapter 9

4WD Robot Posture Estimation by Radial Multi-View Visual Odometry

Edgar Alonso Martínez-García and
Luz Abril Torres-Méndez

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79130

Abstract

This chapter presents a four-wheel robot's trajectory tracking model by an extended Kalman filter (EKF) estimator for visual odometry using a divergent trinocular visual sensor. The trinocular sensor is homemade and a specific observer model was developed to measure 3D key-points by combining multi-view cameras. The observer approaches a geometric model and the key-points are used as references for estimating the robot's displacement. The robot's displacement is estimated by triangulation of multiple pairs of environmental 3D key-points. The four-wheel drive (4WD) robot's inverse/direct kinematic control law is combined with the visual observer, the visual odometry model, and the EKF. The robot's control law is used to produce experimental locomotion statistical variances and is used as a prediction model in the EKF. The proposed dead-reckoning approach models the four asynchronous drives and the four damping suspensions. This chapter presents the deductions of models, formulations and their validation, as well as the experimental results on posture state estimation comparing the four-wheel dead-reckoning model, the visual observer, and the EKF with an external global positioning reference.

Keywords: 4WD, visual odometry, trinocular sensor, EKF, visual observer, trajectory estimation

1. Introduction

Autonomous robots obtain precise information about their surroundings by deploying their sensing devices and developing perceptual tasks to accomplish useful missions. Intelligent robots require to concurrently execute multiple functions such as path planning, collision avoidance,

self-localization, tasks scheduling, trajectory control, map building, environment recognition, kinematic/dynamic control, and so forth. Autonomous robots depend on multisensor fusion, which is the process of combining data from the physical sensors into a homogeneous data space.

This chapter presents robot's visual odometry using sensor data obtained from a homemade radial multi-view device (**Figure 1a**). For this case, trinocular sensing is divergent; hence, an inherent problem refers to different perspectives in each camera. Besides, the partial overlap between adjacent cameras allows sharing approximately 25% of the total sensing angles, which is too reduced and limits extracting numerous relevant features for data fusion affecting to infer consistent information. Besides perspective, radial cameras yield differences of scale, skew, rotation, and lighting intensities. To cope with this problem, this chapter deduces a geometric trinocular sensor model to directly measure 3D data by combining divergent pairs, the central camera with one of the two lateral cameras (**Figure 1b**). The robot's state vector (posture) is recursively estimated by a visual odometry model that triangulates multiple pairs of key-points. Thus, an EKF uses the 3D odometry model and estimates the robot's position. The mobile robot is a four-wheel drive (4WD) modeled by a differential control law involving the four passive damping suspensions to infer accurate positions.

Parallel trinocular stereo systems had been deployed either to detect the ground [1], or to estimate motion [2]. There are reported works on motion estimation with binocular divergent systems [3], trinocular divergence for visual odometry [4], and divergent visual simultaneous localization and mapping (SLAM) [5]. As a difference from the active sensing modalities for localization [6], and concurrent localization and mapping with parallel multi-view [7], this chapter intends to estimate the posture of a rolling vehicle by exploiting feedback of the rich data fusion that a divergent trinocular sensor provides. Numerous visual odometry algorithms had been reported, using stereo cameras [8], matching multi-frame features [9] and 3D point cloud [10]. Some outdoor visual odometry approaches for urban [11] environments estimate motion tracking by extraction of visual feature points. There are numerous works combining the benefits of visual SLAM algorithms [12–14] with visual odometry [15], detecting geometrical features [16].

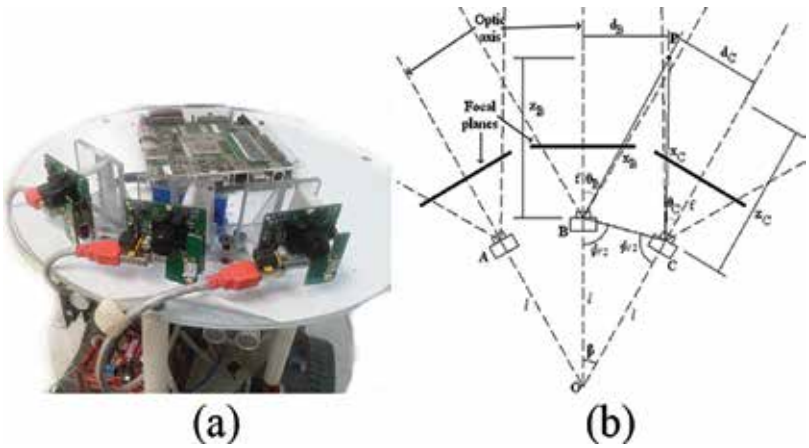


Figure 1. Robot's trinocular sensor. (a) Trinocular sensor onboard. (b) Camera's geometric divergence (top view).

http://dx.doi.org/10.5772/intechopen.79130

This chapter is organized into to the following sections. Section 2 deduces the sensor fusion observer modeling the trinocular system geometry. Section 3 models the 4WD direct/inverse kinematic solutions. Section 4 deduces the visual odometry formulation and EKF-based control state and estimation. Finally, conclusions are provided in Section 5.

2. Trinocular sensing model

This section describes the divergent multi-view geometric model, which basically combines the data of a pair of cameras radially arranged. In addition, this section presents an algebraic analysis of the lateral cameras' alignment and correction w.r.t. the central camera. The fundamental geometrical relationship of the system divergence was experimentally studied by deploying a homemade prototype onboard a mobile robot, see **Figure 2a**. Cameras with homogeneous intrinsic parameters are assumed, and cameras are mechanically fixed epipolar. The sensor model's purpose is to determine the depth information of a point in the scene $\mathbf{p} = (x, y, z)^\top$, which is projected onto the overlapping area of a divergent pair. The proposed multi-view geometric model combines data using the central camera as the reference (**Figure 1b**). The focal plane in cameras A, C are perspective transformed, in order to align them epipolar and coplanar w.r.t. the central reference B. As seen in **Figure 1b**, a point $\mathbf{p}$ is projected over two focal planes, for instance, at column x_C of lateral camera C, and at x_B of camera B. Thus, an isosceles triangle PBC is formed. For the triangle OBC, let β be the angle between the cameras' centers B and C, as deployed by expression (1). Let $\phi/2$ be the remaining angle of OBC, where the inner angles' total sum is π. By expressing $\beta + \phi = \pi$ and dropping off $\phi = \pi - \beta$, we easily deduce that $\frac{\phi}{2} = \frac{\pi}{2} - \frac{\beta}{2}$. The geometrical distance $\overline{\mathrm{BC}}$ is calculated by triangulation using the law of sines, with known distance l that converges to O. The linear distance between adjacent sensors BC commonly oriented w.r.t. the center O is

$$\overline{\mathrm{BC}} = \frac{l \sin \beta}{\sin\left(\frac{\pi}{2} - \frac{\beta}{2}\right)}. \tag{1}$$

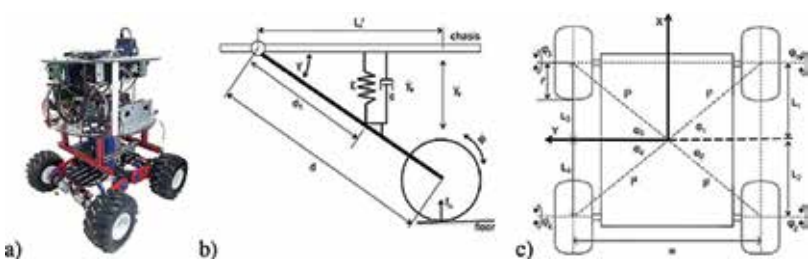


Figure 2. 4WD kinematics. (a) Deployed robot. (b) Damping system. (c) Robot's view from below.

To calculate the Cartesian coordinates of $\mathbf{p}$, let us state that the point $\mathbf{p}$ is projected through the horizontal coordinate x_B, and on camera B angle's θ_B, and focal distance f_B as expressed by

$$\theta_B = \tan^{-1}\left(\frac{x_B}{\mathrm{f}_B}\right) \quad \text{and} \quad \theta_C = \tan^{-1}\left(\frac{x_C}{\mathrm{f}_C}\right). \tag{2}$$

The complementary angles B and C are modeled by

$$\angle B = \frac{\pi}{2} - \theta_B + \frac{\beta}{2} \quad \text{and} \quad \angle C = \frac{\pi}{2} - \theta_C + \frac{\beta}{2}. \tag{3}$$

In the triangle BCO, the angle at point $\mathbf{p}$ is obtained by equivalence of similar triangles $\angle P = \theta_B + \theta_C - \beta$. Thus, to estimate the range of the radial system B and C w.r.t. $\mathbf{p}$, the linear distance is calculated by the law of sines:

$$\frac{\overline{\mathrm{BC}}}{\sin\angle P} = \frac{\overline{\mathrm{CP}}}{\sin\angle B} \quad \text{and} \quad \overline{\mathrm{CP}} = \frac{\overline{\mathrm{BC}}\sin\angle B}{\sin\angle P}. \tag{4}$$

Thus, for the other cameras' pair, similar expressions are stated

$$\frac{\overline{\mathrm{BC}}}{\sin\angle P} = \frac{\overline{\mathrm{BP}}}{\sin\angle C} \quad \text{and } \overline{\mathrm{BP}} = \frac{\overline{\mathrm{BC}}\sin\angle C}{\sin\angle P}. \tag{5}$$

Hence, the model to express depth information is given by $z_B = \overline{\mathrm{BP}}\cos\theta_B$. By substituting $\overline{\mathrm{BP}}$ and θ_B, the model is further specified by

$$\Lambda_1 = \frac{l\sin\beta}{\sin\left(\frac{\pi-\beta}{2}\right)\sin\left(\theta_B - \theta_C - \beta\right)},$$

where,

$$z_B = \Lambda_1 \sin\left(\frac{\pi+\beta}{2} - \theta_C\right)\cos\left(\tan^{-1}\left(\frac{x_B}{\mathrm{f}_B}\right)\right). \tag{6}$$

In addition, the range between camera C and $\mathbf{p}$ is defined by $z_C = \overline{\mathrm{C}\mathbf{p}}\cos\theta_C$. Thus, substituting $\overline{\mathrm{C}\mathbf{p}}$ and θ_C, we define

$$z_C = \Lambda_1 \sin\left(\frac{\pi+\beta}{2} - \theta_B\right)\cos\left(\tan^{-1}\left(\frac{x_C}{\mathrm{f}_C}\right)\right). \tag{7}$$

Using the depth models z_B and z_C, the distances d_B and d_C w.r.t. $\mathbf{p}$ are estimated, such that $d_B = z_B\tan(\theta_B)$. Hence,

$$d_B = z_B\tan\left(\tan^{-1}\left(\frac{x_B}{\mathrm{f}_B}\right)\right) \quad \text{or} \quad d_B = \frac{z_B x_B}{\mathrm{f}_B} \tag{8}$$

and being $d_C = z_C \tan(\theta_C)$, by substituting θ_C from expression (2) we have

$$d_C = z_C \tan\left(\tan^{-1}\left(\frac{x_C}{\mathrm{f}_C}\right)\right) \quad \text{or} \quad d_C = \frac{z_C x_C}{\mathrm{f}_C}. \tag{9}$$

Furthermore, the algebraic deduction along the Y component for the equalities $h_B \mathrm{f}_B = z_B y_B$ and $h_C \mathrm{f}_C = z_C y_C$, w.r.t $\mathbf{p}$ using distances h_B and h_C, is obtained by

$$h_B = \frac{z_B y_B}{\mathrm{f}_B} \quad \text{and} \quad h_C = \frac{z_C y_C}{\mathrm{f}_C},$$

thus the following term is stated as

$$\Psi = \frac{1}{\mathrm{f}_B}\left(\frac{l \sin\beta}{\sin\left(\frac{\pi}{2} - \frac{\beta}{2}\right)}\right) \cos\left(\tan^{-1}\left(\frac{x_B}{\mathrm{f}_B}\right)\right).$$

Therefore, the geometry vector model for camera B w.r.t. camera C, with substitution of the terms Ψ, z_B, and z_C in robot's inertial frame ^{R}, produce the next expression:

$$\mathbf{p}^R_{BC} = \frac{\Psi \sin\left(\frac{\pi+\beta}{2} - \theta_C\right)}{\sin\left(\theta_B + \theta_C - \beta\right)} \begin{pmatrix} x_B \\ y_B \\ \mathrm{f}_B \end{pmatrix} \tag{10}$$

and the same model is enhanced for camera B, using the geometry of cameras A and B by

$$\mathbf{p}^R_{AB} = \frac{\Psi \sin\left(\frac{\pi+\beta}{2} - \theta_A\right)}{\sin\left(\theta_A + \theta_B - \beta\right)} \begin{pmatrix} x_B \\ y_B \\ \mathrm{f}_B \end{pmatrix}. \tag{11}$$

Hence, the arbitrary point $\mathbf{p}_{ABC} \in \mathbb{R}^3$ is projected onto cameras AB, or onto cameras BC. In order to express a general formula, let us define the following theorem.

Theorem 1 (Trinocular depth model). *Let camera B be the reference for either divergent camera A or C. A point coordinates w.r.t. camera A is* $\mathbf{p}_{AB} = \left(x_A, y_A, z\right)^T$*, and w.r.t. camera C is* $\mathbf{p}_{AC} = \left(x_C, y_C, z\right)^T$*. Hence, the general depth coordinate model for x and y for any divergent pair is*

$$x, y_{A,C} = \frac{\Psi x_B \sin\left(\frac{\pi+\beta}{2} - \theta_{A,C}\right)}{\sin\left(\theta_{A,C} + \theta_B - \beta\right)}, \tag{12}$$

for coordinate z,

$$z_{A,C} = \frac{\Psi f_B^2 \sin\left(\frac{\pi+\beta}{2} - \theta_{A,C}\right)}{\sin\left(\theta_{A,C} + \theta_B - \beta\right)}. \tag{13}$$

The four points shown by the three cameras may illustrate their transformation, experimentally developed at 1-m distance between the robot and the marks.

3. 4WD dead-reckoning controller

Since the visual trinocular approach uses an exteroceptive sensor, we decided to challenge its detection and tracking capabilities with a robot having high holonomic properties. A 4WD robot's locomotion is prone to experience frequent swift turns resulting in numerous slippages. Thus, a 4WD has to depend more on exteroceptive rather than inner measurements. Comparatively, inner 4WD odometry differs greatly from external visual measurement to infer posture. The proposed dead-reckoning system obtains speed measurements by deploying odometer readings of the four asynchronous drives (**Figure 2**). A 4WD system is considerably different from a conventional differential dual approach. Moreover, four passive mass-spring-damper suspensions are included in this system (**Figure 2b**), which varies the inter-wheel distances over time. Particularly, the robot's 4WD and passive suspensions make the posture observations challenging.

The robot's dead-reckoning model is fundamental to sense and control position used as feedback, providing motion description as a kinematic reference to match the visual observations when estimating the robot's motion. The positioning and trajectory control [17], as well as the type of kinematic analysis [18] and the dynamic suspension [19] in this type of robot have been previously reported. The robot's instantaneous speed v_t (m/s) and yaw rate ω_t (rad/s) depend on the four wheels' asynchronous rolling motion, $\dot{\varphi}_1, \dot{\varphi}_2, \dot{\varphi}_3, \dot{\varphi}_4$. For a wheel's encoder, the velocity model approaches measurements by high-precision numerical derivatives (central divided differences) of the rotary angle φ_t (rad) w.r.t. time t, such that

$$\dot{\varphi}_t = \frac{\pi}{6Rt}\left[\eta_{t+2} + 7\eta_{t+1} + 7\varphi_{t-1} - \eta_{t-2}\right], \tag{14}$$

where the wheel's angular speed $\dot{\varphi}$ is measured through pulse detection η (dimensionless) of encoder's resolution R (pulses/rev); thus, the robot's instantaneous velocity is modeled by the averaged wheel speed, with wheels of nominal radius r (m)

$$v_t = \frac{r}{4}\sum_{i=1}^{4}\dot{\varphi}_i. \tag{15}$$

Further, the differential velocity $\hat{v}_t$ expresses the lateral speeds' difference that yields ω_t. Thus, $\hat{v}_t$ is formulated by the expression

$$\hat{v}_t = r\left(\dot{\varphi}_1 + \dot{\varphi}_2 - \dot{\varphi}_3 - \dot{\varphi}_4\right). \tag{16}$$

This model describes that the rotary motion of $\dot{\varphi}_1$ and $\dot{\varphi}_2$ contributes to robot's $+\omega_t$ (counterclockwise sense). Likewise, $\dot{\varphi}_3$ and $\dot{\varphi}_4$ contribute to robot's $-\omega_t$ (clockwise sense). Therefore, ω_t is yielded by the lateral speed component $\hat{v}_t \cos(\alpha_i)$ (see **Figure 2b**) modeled by

$$\omega_t = \frac{\hat{v}\cos\left(\cos\left(\alpha_i\right)\right)}{l_i}. \tag{17}$$

The previous equation expresses the conservation of angular motion, and the wheel's contact point turns w.r.t. the robot's center,

$$\cos\left(\alpha_i\right) = \frac{W}{2l_i}, \tag{18}$$

where for each length l_i there is an asynchronous model,

$$l_1 = \frac{\sqrt[2]{W^2 + \left(L_1 + L_4\right)^2}}{2}, \quad l_2 = \frac{\sqrt[2]{W^2 + \left(L_2 + L_3\right)^2}}{2},$$

as well as

$$l_3 = \frac{\sqrt[2]{W^2 + \left(L_3 + L_2\right)^2}}{2}, \quad l_4 = \frac{\sqrt[2]{W^2 + \left(L_4 + L_1\right)^2}}{2}.$$

Thus, substituting $\cos\left(\alpha_i\right)$ and l_i into ω_t and by considering both clockwise and counterclockwise motions, the robot's angular speed is

$$\omega_t = \sum_{i=1}^{2} \frac{rW\dot{\varphi}_i}{l_i^2} - \sum_{i=3}^{4} \frac{rW\dot{\varphi}_i}{l_i^2}. \tag{19}$$

The longitudinal contact point's distance l_i (m) takes as reference the robot's geometric center. When l_i varies, the contact point's position L_i changes.

$$L_i = d\cos\left(\arcsin\left(\gamma_i\right)\right), \tag{20}$$

where γ_i represents the vertical motion along the suspension,

$$\gamma_i = \arcsin\left(\frac{\Delta_y}{d_1}\right). \tag{21}$$

From **Figure 2a**, the vertical motion y_d is modeled assuming critical damping motion for a general spring-mass-damper system. The suspension is modeled by the following second-order homogeneous differential equation:

$$m\ddot{y}_d + \kappa_2\dot{y}_d + \kappa_1 y_d = 0, \tag{22}$$

where the elastic spring restitution coefficient is κ_1 (kg/s^2). The damping coefficient is κ_2 (kg/s). The restitution force $m\ddot{y}_d$ counteracts the vertical oscillatory damping effects. The oscillatory velocity and acceleration are denoted by $\dot{y}_d$ and $\ddot{y}_d$, respectively. Thus, solving the second-order differential equation as a first-order equation such that $\kappa_2\dot{y}_d = -\kappa_1 y_d$,

$$\int_{y_d} \frac{\mathrm{d}y_d}{y_d} = -\frac{\kappa_1}{\kappa_2} \int_t \mathrm{d}t, \tag{23}$$

hence

$$\ln\left(y_d\right) = -\frac{\kappa_1}{\kappa_2} t + c, \quad \zeta = -\frac{\kappa_1}{\kappa_2}, \tag{24}$$

with integration constant $c = 0$ for analysis purpose. The suspension's elongation derivatives as functions of time are

$$y_d = \mathrm{e}^{\zeta t}, \quad \dot{y}_d = \zeta \mathrm{e}^{\zeta t}, \quad \ddot{y}_d = \zeta^2 \mathrm{e}^{\zeta t}. \tag{25}$$

Substituting the previous expression in (22),

$$m\zeta^2 \mathrm{e}^{\zeta t} + \kappa_2 \zeta \mathrm{e}^{\zeta t} + \kappa_1 \mathrm{e}^{\zeta t} = 0, \tag{26}$$

and by algebraically simplifying, the characteristic equation is

$$\zeta^2 + \frac{\kappa_2}{m}\zeta + \frac{\kappa_1}{m} = 0, \tag{27}$$

and its analytic solution is

$$\zeta_{1,2} = \frac{-\kappa_2}{2m} \pm \frac{\sqrt[2]{\left(\frac{\kappa_1}{m}\right)^2 - 4\frac{\kappa_1}{m}}}{2}. \tag{28}$$

As we assume a critically damped system, $\left(\frac{\kappa_2}{m}\right)^2 = 4\left(\frac{\kappa_1}{m}\right)$ and there is only one real root solution, such that

$$\zeta = -\frac{\kappa_2}{2m}. \tag{29}$$

Therefore, the damping motion is analytically solved by

$$y_d(t) = A\mathrm{e}^{\zeta t}, \tag{30}$$

where A (m) is the elongation amplitude (m) parameter for the suspension system. Moreover, in this type of robotic platform, the four asynchronous drives simultaneously produce slip/skid motions that are advantageously used to maneuver the robot. This approach proposes inferring the instantaneous Z-turn axis location $\left(x_R, y_R\right)^\top$. The Z-turn axis is movable in the square region bounded by the wheels' contact point. The Z-turn axis is expressed by the first-order derivatives

$$\dot{x}_R = \frac{rW}{4v_{max}}\left(\ddot{\varphi}_1 - \ddot{\varphi}_2 - \ddot{\varphi}_3 + \ddot{\varphi}_4\right) \tag{31}$$

and

$$\dot{y}_R = \frac{rL}{4v_{max}}\left(-\ddot{\varphi}_1 - \ddot{\varphi}_2 + \ddot{\varphi}_3 + \ddot{\varphi}_4\right). \tag{32}$$

There is a maximal allowable Z-turn displacement speed v_{max}. Hereafter, with four independent equations, the control positioning system is instantaneously computed. The robot control vector is $\dot{\mathbf{u}}_R = \left(\dot{v}_t, \dot{\omega}_t, \dot{x}_R, \dot{y}_R\right)^\top$, and the control transition matrix $\mathbf{\Lambda}_t$ has the elements $\lambda_1 = r/4$, $\lambda_{2i} = rW/l_i^2$, $\lambda_3 = rW/(4v_{max})$, and $\lambda_4 = rL/(4v_{max})$. Thus, the forward kinematics solution is $\dot{\mathbf{u}}_R = \mathbf{\Lambda}_t \cdot \dot{\mathbf{\Omega}}$ or

$$\begin{pmatrix} \dot{v}_t \\ \dot{\omega}_t \\ \dot{x}_R \\ \dot{y}_R \end{pmatrix} = \begin{pmatrix} \lambda_1 & \lambda_1 & \lambda_1 & \lambda_1 \\ \lambda_{21} & \lambda_{22} & -\lambda_{23} & -\lambda_{24} \\ \lambda_3 & -\lambda_3 & -\lambda_3 & \lambda_3 \\ -\lambda_4 & -\lambda_4 & \lambda_4 & \lambda_4 \end{pmatrix} \cdot \begin{pmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \\ \ddot{\varphi}_3 \\ \ddot{\varphi}_4 \end{pmatrix}. \tag{33}$$

In addition, to inversely solve this matrix system, the analytical solution represents the vector of independent control rotary variables $\dot{\Omega}_t = \left(\ddot{\varphi}_1, \ddot{\varphi}_2, \ddot{\varphi}_3, \ddot{\varphi}_4\right)^\top$. Thus, let us define $\lambda_w = \lambda_1\lambda_3\lambda_4$, $\lambda_A = \lambda_{23} - \lambda_{24}$, $\lambda_B = \lambda_{22} - \lambda_{21}$, $\lambda_C = \lambda_{21} - \lambda_{23}$, $\lambda_D = \lambda_{24} - \lambda_{22}$, $\lambda_E = \lambda_{21} + \lambda_{24}$, and $\lambda_F = \lambda_{22} - \lambda_{23}$.

$$\ddot{\varphi}_1 = \frac{\lambda_3\lambda_4\lambda_D\dot{v} + 2\lambda_w\dot{\omega} - \lambda_1\lambda_4\lambda_A\dot{x}_R + \lambda_1\lambda_3\lambda_F\dot{y}_R}{2\lambda_w(\lambda_{21} - \lambda_{22} - \lambda_{23} + \lambda_{24})}, \tag{34}$$

$$\ddot{\varphi}_2 = \frac{\lambda_3\lambda_4\lambda_C\dot{v} - 2\lambda_w\dot{\omega} + \lambda_1\lambda_4\lambda_A\dot{x}_R - \lambda_1\lambda_3\lambda_E\dot{y}_R}{2\lambda_w(\lambda_{21} - \lambda_{22} - \lambda_{23} + \lambda_{24})}, \tag{35}$$

$$\ddot{\varphi}_3 = \frac{\lambda_3\lambda_4\lambda_D\dot{v} + 2\lambda_w\dot{\omega} + \lambda_1\lambda_4\lambda_B\dot{x}_R + \lambda_1\lambda_3\lambda_E\dot{y}_R}{2\lambda_w(\lambda_{21} - \lambda_{22} - \lambda_{23} + \lambda_{24})} \tag{36}$$

and

$$\ddot{\varphi}_4 = \frac{\lambda_3\lambda_4\lambda_C\dot{v} - 2\lambda_w\dot{\omega} - \lambda_1\lambda_4\lambda_B\dot{x}_R - \lambda_1\lambda_3\lambda_F\dot{y}_R}{2\lambda_w(\lambda_{21} - \lambda_{22} - \lambda_{23} + \lambda_{24})}. \tag{37}$$

The matrix form of the inverse analytical solution for all wheels' speed under damping variations is stated as $\dot{\mathbf{\Omega}} = \mathbf{\Lambda}_t^{-1} \cdot \dot{\mathbf{u}}_t$ or

$$\dot{\Omega}_t = \begin{pmatrix} \frac{\lambda_3\lambda_4\lambda_D}{2\lambda_w\lambda_G} & \frac{2\lambda_w}{2\lambda_w\lambda_G} & \frac{-\lambda_1\lambda_4\lambda_A}{2\lambda_w\lambda_G} & \frac{\lambda_1\lambda_3\lambda_F}{2\lambda_w\lambda_G} \\ \frac{\lambda_3\lambda_4\lambda_C}{2\lambda_w\lambda_G} & \frac{-2\lambda_w}{2\lambda_w\lambda_G} & \frac{\lambda_1\lambda_4\lambda_A}{2\lambda_w\lambda_G} & \frac{-\lambda_1\lambda_3\lambda_E}{2\lambda_w\lambda_G} \\ \frac{\lambda_3\lambda_4\lambda_D}{2\lambda_w\lambda_G} & \frac{2\lambda_w}{2\lambda_w\lambda_G} & \frac{\lambda_1\lambda_4\lambda_B}{2\lambda_w\lambda_G} & \frac{\lambda_1\lambda_3\lambda_E}{2\lambda_w\lambda_G} \\ \frac{\lambda_3\lambda_4\lambda_C}{2\lambda_w\lambda_G} & \frac{-2\lambda_w}{2\lambda_w\lambda_G} & \frac{-\lambda_1\lambda_4\lambda_B}{2\lambda_w\lambda_G} & \frac{-\lambda_1\lambda_3\lambda_F}{2\lambda_w\lambda_G} \end{pmatrix} \cdot \begin{pmatrix} \dot{v}_t \\ \dot{\omega}_t \\ \dot{x}_R \\ \dot{y}_R \end{pmatrix}, \tag{38}$$

where $\lambda_G = \lambda_{21} - \lambda_{22} - \lambda_{23} + \lambda_{24}$.

4. State estimation and feedback position control

This section formulates a deterministic geometric model for visual odometry and the state estimation by an EKF. The proposed model combines pairs of key-points at times t and $t-1$. The robot's displacements are deduced by inverse geometric triangulations to feed forward an EKF and estimate the robot's posture.

In **Figure 3a**, the instantaneous angle α_{t-1} is formed by a pair of key-points defined by

$$\alpha_{t-1} = |\theta^a_{t-1}| + |\theta^b_{t-1}| \tag{39}$$

and such key-points' distance c_{t-1} is defined by

$$c_{t-1} = \sqrt{\left(\delta^a_{t-1}\right)^2 + \left(\delta^b_{t-1}\right)^2 - 2\delta^a_{t-1}\delta^b_{t-1}\cos\alpha_{t-1}}. \tag{40}$$

The angle $\beta^{a,b}_{t-1}$ of either key-point a or b is calculated by the law of sines,

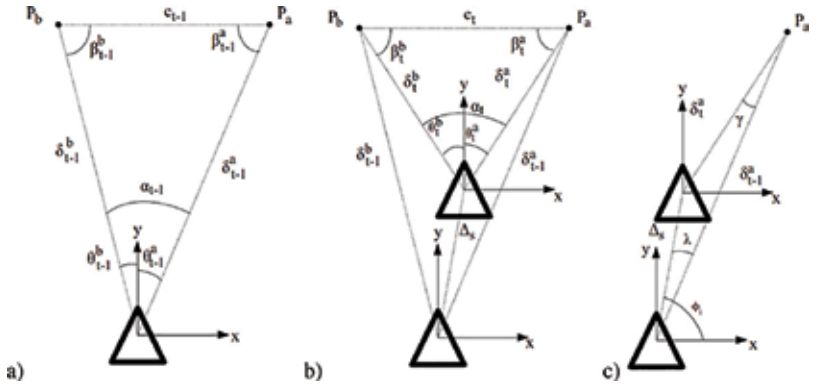


Figure 3. Robot's visual odometry. (a) Robot's key-point pair observed at $t-1$. (b) Same pair observed at t. (c) Robot's displacement Δ_s by triangulation of key-points $\mathbf{p}_a$.

$$\beta_{t-1}^{a,b} = \arcsin\left(\frac{\delta_{t-1}^{b,a}\sin\alpha_{t-1}}{c_{t-1}}\right). \tag{41}$$

However, at time t in **Figure 3b**, the instantaneous angle α_t is obtained by

$$\alpha_t = |\theta_t^a| + |\theta_t^b|, \tag{42}$$

with the key-point's distance c_t as

$$c_t = \sqrt{\left(\delta_t^a\right)^2 + \left(\delta_t^b\right)^2 - 2\delta_t^a\delta_t^b\cos\left(\alpha_t\right)}, \tag{43}$$

which is used to obtain the angle $\beta_t^{a,b}$ of the key-point a or b at actual time

$$\beta_t^{a,b} = \arcsin\left(\frac{\delta_t^{b,a}\sin\alpha_t}{c_t}\right). \tag{44}$$

Further, the differential angle $\hat{\beta}$ is defined by triangulation of previous and actual poses and an arbitrary 3D point $\mathbf{p}_{a,b}$ (**Figure 3c**),

$$\hat{\beta} = \beta_{t-1}^{a,b} - \beta_t^{a,b}. \tag{45}$$

Proposition 1 (Triangulation odometric displacement). The robot's displacement Δ_s (**Figure 3c**) that is inferred by triangulation of visual key-points over time is

$$\Delta_s = \sqrt{\delta_{t-1}^a + \delta_t^a - 2\delta_{t-1}^a\delta_t^a\cos\left(\hat{\beta}\right)}. \tag{46}$$

The triangulation angle λ is calculated by the law of sines,

$$\lambda^{a,b} = \arcsin\left(\frac{\delta_t^{a,b}\sin\left(\hat{\beta}\right)}{\Delta_s^{a,b}}\right) \tag{47}$$

and the orientation angle for each reference a is

$$\phi^{a,b} = \lambda^{a,b} + \left(\frac{\pi}{2} - \theta_{t-1}^{a,b}\right), \tag{48}$$

which is required to know the X displacement

$$\Delta_x = \Delta_s^{a,b}\cos\left(\phi^{a,b}\right), \tag{49}$$

as well as the Y displacement

$$\Delta_y = \Delta_s^a\sin\phi^a. \tag{50}$$

When obtaining numerous key-point pairs simultaneously, the total robot's displacement is an averaged value of the displacements yielded by all key-point pairs,

$$\Delta_{x_t} = \frac{\sum_{i=1}^{n} \Delta_{x_i}}{n} \quad \text{and} \quad \Delta_{y_t} = \frac{\sum_{i=1}^{n} \Delta_{y_i}}{n}. \tag{51}$$

Therefore, without loss of generality, for state estimation, let us assume a nonlinear robot's model state vector

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k), \tag{52}$$

where the state vector is $\mathbf{x} = (x, y, \theta, v, \omega)$, and combined with a nonstationary state transition matrix $\mathbf{A}_t$, such that $\mathbf{x}_k = \mathbf{A}_t \cdot \mathbf{x}_{k-1}$ or

$$\mathbf{x}_k = \begin{pmatrix} 1 & 0 & 0 & \cos(\theta)\Delta t & 0 \\ 0 & 1 & 0 & \sin(\theta)\Delta t & 0 \\ 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ \theta \\ v \\ \omega \end{pmatrix}, \tag{53}$$

by developing the dot product from previous expression, we obtain

$$\mathbf{x}_k = \begin{pmatrix} x_{k-1} + v\cos(\theta)\Delta t \\ x_{k-1} + v\sin(\theta)\Delta t \\ \theta_{k-1} + \omega\Delta t \\ v \\ \omega \end{pmatrix}. \tag{54}$$

The measurement model requires the displacements that were inferred through key-point triangulation

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k), \tag{55}$$

where $\mathbf{w}_k$ and $\mathbf{v}_k$ are the process and measurement noise models, respectively. These are statistically independent and supposed to have a Gauss distribution with zero average value and known variance. To approximate the nonlinear robot's measurement model, a linearized first-order approximation by the expansion of the Taylor series is used, and a linear approximation of a function is built, with slope obtained through partial derivatives by

$$f'(u_t, x_{t-1}) = \frac{\partial f(u_t, x_{t-1})}{\partial x_{t-1}}. \tag{56}$$

Thus, the linearized models of the process and measurement are defined next in (54) and (55), such that

$$f(\mathbf{x}) = f(\hat{\mathbf{x}}) + \underbrace{f'(\hat{\mathbf{x}})}_{=J} (\mathbf{x} - \hat{\mathbf{x}}) \tag{57}$$

and

$$h(\mathbf{x}) = h(\hat{\mathbf{x}}) + \underbrace{h'(\hat{\mathbf{x}})}_{=H} (\mathbf{x} - \hat{\mathbf{x}}). \tag{58}$$

In addition, the EKF's prediction models (56) and the correction models (57) are formulated and linearized as

$$\hat{\mathbf{x}}_k^- = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, 0) \tag{59}$$

and

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^\top + \mathbf{W}_k \mathbf{Q}_{k-1} \mathbf{W}_k^\top. \tag{60}$$

Moreover, the recursive Kalman gain for system convergence is

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\top \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^\top \right)^{-1} \tag{61}$$

and the state vector of the system is described by

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}\bar{\mathbf{x}}_k), \tag{62}$$

with covariance matrix of the system

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-. \tag{63}$$

Thus, hereafter, the vector and matrix models describing the proposed robot's system are formulated and incorporated into the conventional EKF. Let us define the robot's pose vector $\mathbf{x} = \left(x_t, y_t, \theta_t\right)^\top$. The control vector is comprised of the robot's absolute and angular speeds, $\mathbf{u}_k = (\upsilon, \omega)^\top$. Furthermore, the observation vector with sensor measurement $\mathbf{z}_k = \left(X_p, Y_p, Z_p\right)^\top$. Eventually, the process noise vector $\mathbf{w}_k = \left(w_x, w_y, w_\theta\right)^\top$. The measurement noise vector $\mathbf{v}_k = \left(v_{X_p}, v_{Y_p}, v_{Z_p}\right)^\top$.

Therefore, from the displacement equation (46), which arises from exteroceptive observations, the robot's Cartesian displacements are

$$\Delta x = (\Delta s) \cos\left(\theta_{k-1} + \lambda + \left(\frac{\pi}{2} - \theta_{t-1}^a\right)\right) \tag{64}$$

and

$$\Delta y = (\Delta s)\sin\left(\theta_{k-1} + \lambda + \left(\frac{\pi}{2} - \theta^{a}_{t-1}\right)\right), \tag{65}$$

as well as $\Delta\theta$ is given by

$$\Delta\theta = \lambda + \left(\frac{\pi}{2} - \theta^{a}_{t-1}\right). \tag{66}$$

By substituting an averaged Cartesian displacement, one considers n key-point observations to calculate the recursive noisy state vector $\mathbf{x}_k$.The Jacobian matrix $\mathbf{J}_k$ of the robot's temporal matrix state transition $\mathbf{A}_k \cdot \mathbf{x}_k$, where $x_k = x_{k-1} + v\cos(\theta)\Delta t$, $y_k = y_{k-1} + v\cos(\theta)\Delta t$, and $\theta_k = \theta_{k-1} + \omega\Delta t$ is stated by

$$\mathbf{J} = \frac{\partial \mathbf{A}_k \cdot \mathbf{x}_k}{\partial \mathbf{x}_{k-1}} = \begin{pmatrix} \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial \omega} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial \omega} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \theta} & \frac{\partial \theta}{\partial v} & \frac{\partial \theta}{\partial \omega} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial \theta} & \frac{\partial v}{\partial v} & \frac{\partial v}{\partial \omega} \\ \frac{\partial \omega}{\partial x} & \frac{\partial \omega}{\partial y} & \frac{\partial \omega}{\partial \theta} & \frac{\partial \omega}{\partial v} & \frac{\partial \omega}{\partial \omega} \end{pmatrix} = \begin{pmatrix} 1 & 0 & -v\sin(\theta)\Delta t & \cos(\theta)\Delta t & 0 \\ 0 & 1 & v\cos(\theta)\Delta t & \sin(\theta)\Delta t & 0 \\ 0 & 0 & 1 & 0 & t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{67}$$

Thus, a measurement is a 3D point arising from either divergent pair Eq. (10) or (11) and deployed by Proposition 1.1. Thus, the robot's measurement vector model $\mathbf{z}$ includes noise measurements. The Jacobian matrix $\mathbf{H}$ of the expected state model w.r.t. measurements is defined by

$$\mathbf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} == \begin{pmatrix} \frac{\partial \Delta x}{\partial x_{k-1}} & \frac{\partial \Delta x}{\partial y_{k-1}} & \frac{\partial \Delta x}{\partial \theta_{k-1}} \\ \frac{\partial \Delta y}{\partial x_{k-1}} & \frac{\partial \Delta y}{\partial y_{k-1}} & \frac{\partial \Delta y}{\partial \theta_{k-1}} \\ \frac{\partial \Delta \theta}{\partial x_{k-1}} & \frac{\partial \Delta \theta}{\partial y_{k-1}} & \frac{\partial \Delta \theta}{\partial \theta_{k-1}} \end{pmatrix}. \tag{68}$$

The process noise covariance matrix $\mathbf{Q}_k$ is defined by

$$\mathbf{Q}_k |2\pi\mathbf{\Sigma}|^{-1/2} \exp\left\{0.5(\mathbf{z} - \boldsymbol{\mu})^{\top}\mathbf{\Sigma}^{-1}(\mathbf{z} - \boldsymbol{\mu})\right\}. \tag{69}$$

Let us define the nonstationary covariance matrix $\mathbf{P}$,

$$\mathbf{P} = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{pmatrix}, \tag{70}$$

the matrix diagonal variances are experimental measurements that describe the trend of the robot motion's error.

The robot's motion covariance matrix was obtained experimentally through 500 tests—straight motion, right turns, left turns, clockwise and counterclockwise rotations, with $N = 100$ tests for each type of motion. Exteroceptive sensing devices onboard are tied to the robot's geometry of motion, and with their observations the robot's posture can be estimated and therefore matched with the robot's deterministic kinematic model. From Section 3, the inverse (38) and direct (33) models were used experimentally to obtain the following statistical covariance about the measurement model

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\int_t (\lambda_1,\lambda_1,\lambda_1,\lambda_1)^\top \cdot \dot{\Omega}\mathrm{d}t \cdot \cos\left(\int_t \left(\frac{rW}{l_1^2}\ \frac{rW}{l_2^2}\ \frac{-rW}{l_3^2}\ \frac{-rW}{l_4^2}\right)^\top \cdot \dot{\boldsymbol{\Omega}}\mathrm{d}t\right) - \overline{x}\right)^2 \tag{71}$$

and

$$\sigma_y^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\int_t (\lambda_1,\lambda_1,\lambda_1,\lambda_1)^\top \cdot \dot{\Omega}\mathrm{d}t \cdot \sin\left(\int_t \left(\frac{rW}{l_1^2}\ \frac{rW}{l_2^2}\ \frac{-rW}{l_3^2}\ \frac{-rW}{l_4^2}\right)^\top \cdot \dot{\boldsymbol{\Omega}}\mathrm{d}t\right) - \overline{y}\right)^2, \tag{72}$$

as well as the robot's yaw statistical measurement model

$$\sigma_\theta^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\int_t \left(\frac{rW}{l_1^2}\ \frac{rW}{l_2^2}\ \frac{-rW}{l_3^2}\ \frac{-rW}{l_4^2}\right)^\top \cdot \dot{\boldsymbol{\Omega}}\mathrm{d}t - \overline{\theta}\right)^2. \tag{73}$$

Furthermore, the measurement noise covariance matrix is

$$\mathbf{R}_k = \begin{pmatrix} \sigma_{x_p}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{y_p}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{y_p}^2 & 0 & 0 & 0 \end{pmatrix}, \tag{74}$$

and the matrix $\mathbf{W}_k$ which is the partial derivative of the process model w.r.t. the process noise vector is

$$\mathbf{W}_k = \frac{\partial f}{\partial \mathbf{w}_k} = \begin{pmatrix} \frac{\partial x_k}{\partial w_x} & \frac{\partial x_k}{\partial w_y} & \frac{\partial x_k}{\partial w_\theta} \\ \frac{\partial y_k}{\partial w_x} & \frac{\partial y_k}{\partial w_y} & \frac{\partial y_k}{\partial w_\theta} \\ \frac{\partial \theta_k}{\partial w_x} & \frac{\partial \theta_k}{\partial w_y} & \frac{\partial \theta_k}{\partial w_\theta}. \end{pmatrix}. \tag{75}$$

The matrix $\mathbf{V}_k = \partial h/\partial \mathbf{v}_k$ is the partial derivative w.r.t. the measurement noise vector,

$$\mathbf{V}_k = \begin{pmatrix} \frac{\partial x_k}{\partial v_{X_p}} & \frac{\partial x_k}{\partial v_{Y_p}} & \frac{\partial x_k}{\partial v_{Z_p}} \\ \frac{\partial y_k}{\partial v_{X_p}} & \frac{\partial y_k}{\partial v_{Y_p}} & \frac{\partial y_k}{\partial v_{Z_p}} \end{pmatrix}. \tag{76}$$

Let us summarize the 3D points $\mathbf{p}_{AB}$ and $\mathbf{p}_{BC}$ obtained by Theorem 1.

4.1. State feedback position control

This section describes in six general steps the combined use of the visual observers and the EKF geometric odometer as a recursive feedback for the robot's positioning control. The robot's deterministic kinematic model conveys predictions about the robot's geometry of motion and its observations. Therefore, the deterministic model is used to infer the robot's motion observations implicitly by the trinocular sensor. The following formulation illustrates how the EKF and the visual odometry model are fed back for the 4WD kinematics.

4.1.1. Kalman gain

The initial estimate of Kalman gain is

$$\mathbf{k}_k = \mathbf{P}_k \mathbf{H}_k^{\top} \left(\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_x \mathbf{V}_k^T \right)^{-1} \tag{77}$$

4.1.2. Observation

From Proposition 1.1, the visual observers provide m 3D key-points from Theorem 1, $\mathbf{p}_{AB,\,BC}$

$$\Delta s(\mathbf{p}) = \sqrt[2]{\|\mathbf{p}_{t-1}\| + \|\mathbf{p}_t\| - 2\|\mathbf{p}_{t-1}\|\|\mathbf{p}_t\| \cos\left(\beta_{t-1} - \beta_t\right)}$$

The angle of each key-point $\mathbf{p}$ or $\mathbf{q}$ w.r.t. to the robot in actual time is

$$\lambda_t = \arcsin\left(\frac{\|\mathbf{p}_t \sin\left(\hat{\beta}\right)\|}{\Delta s} \right),$$

and the local angle of the robot w.r.t. the robot's previous position is

$$\phi = \lambda_t + \left(\frac{\pi}{2} - \theta_{t-1} \right),$$

thus the inferred displacement is

$$\mathbf{x}_k = \begin{pmatrix} \frac{1}{m+n} \sum_i \Delta s_i^a(\mathbf{p}) \cos\left(\phi_i^a\right) \\ \frac{1}{m+n} \sum_i \Delta s_i^a(\mathbf{p}) \sin\left(\phi_i^a\right) \\ \lambda + \left(\frac{\pi}{2} - \theta_{t-1}^a \right) \end{pmatrix}. \tag{78}$$

Therefore, the observation vector with Gauss noise **w** is

$$\mathbf{z}_k = \mathbf{H} \cdot \mathbf{x}_k + \begin{pmatrix} w_x \\ w_y \\ w_\theta \end{pmatrix}. \tag{79}$$

4.1.3. Update estimate

The update estimate is obtained by

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1}^{-} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{x}_{k-1}). \tag{80}$$

4.1.4. Update error covariance

The covariance matrix error dispersion of the system is updated

$$\hat{\mathbf{P}}_k = \mathbf{P}_k - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_k. \tag{81}$$

4.1.5. Deterministic control model

Therefore, the prediction is firstly obtained through the robot's inverse position control model, from the inverse kinematics equation, Eq. (38)

$$\mathbf{\Omega}_{t+1} = \mathbf{\Omega}_t + \mathbf{\Lambda}_t^{-1} \cdot \left(\mathbf{u}_R^{ref} - \hat{\mathbf{x}}_k\right),$$

where, in the previous expression, $\mathbf{u}_R = \left(s, \theta, x_R, y_R\right)^\top$. Thus, $\hat{\mathbf{\Omega}}_t = \left(\frac{2\pi}{R}\Delta\eta_1, \frac{2\pi}{R}\Delta\eta_2, \frac{2\pi}{R}\Delta\eta_3, \frac{2\pi}{R}\Delta\eta_4\right)^\top$ is the vector of the wheels' instantaneous measurements

$$\mathbf{u}_{t+1} = \mathbf{u}_t - \mathbf{\Lambda}_t^{-1} \cdot \left(\mathbf{\Omega}_{t+1} - \hat{\mathbf{\Omega}}_t\right).$$

This step converges until $\left(\mathbf{u}_R^{ref} - \hat{\mathbf{u}}_t\right) < \varepsilon_u$, where ε_u is the convergence error. Then, the robot's prediction model is

$$\overline{\mathbf{x}}_k = \overline{\mathbf{x}}_{k-1} + \mathbf{B} \cdot \mathbf{u}_{t+1}$$

B being a control transition matrix.

4.1.6. State prediction

It follows that state prediction is

$$\overline{\mathbf{x}}_{k+1} = \mathbf{\Phi}_k + \overline{\mathbf{x}}_k + \mathbf{q}_k \tag{82}$$

and the error dispersion covariance matrix is also predicted at $t + 1$

$$\mathbf{P}_{k+1} = \mathbf{P}_k + \left(\mathbf{A} + \mathbf{A}^{\top}\right)\mathbf{P}_k\Delta t + \left(\mathbf{A}\mathbf{P}_k\mathbf{A}^{\top} - \mathbf{\Sigma}_W\right)\Delta t^2 \tag{83}$$

From the previous step, the estimation process repeats again, going to step one. The previous Kalman process is performed until the robot reaches the goal and the estimation error converges by numerical approximation according to $(\mathbf{x}_k - \hat{\mathbf{x}}_k) \leq \varepsilon_{\mathbf{x}}$.

Therefore, **Figure 4a** shows the robot's trajectory obtained by the different comparative approaches conducted in this study. The postures measured by an external visual global reference system are the main references to be compared with. The EKF estimation was obtained by the use of Theorem 1, Proposition 1.1, and Eqs. (71)–(77). In addition, the trinocular key-points used as inputs of the visual odometry model inferred the robot's displacements, which are shown in same **Figure 4a**. Furthermore, the dead-reckoning robot system was deployed to infer the robot's postures and is also shown in **Figure 4a**. Raw odometry refers to the robot's dead-reckoning kinematic model used as a mean for direct posture observation through direct kinematics (33) and inverse kinematics (38), but using direct encoder readings by (14).

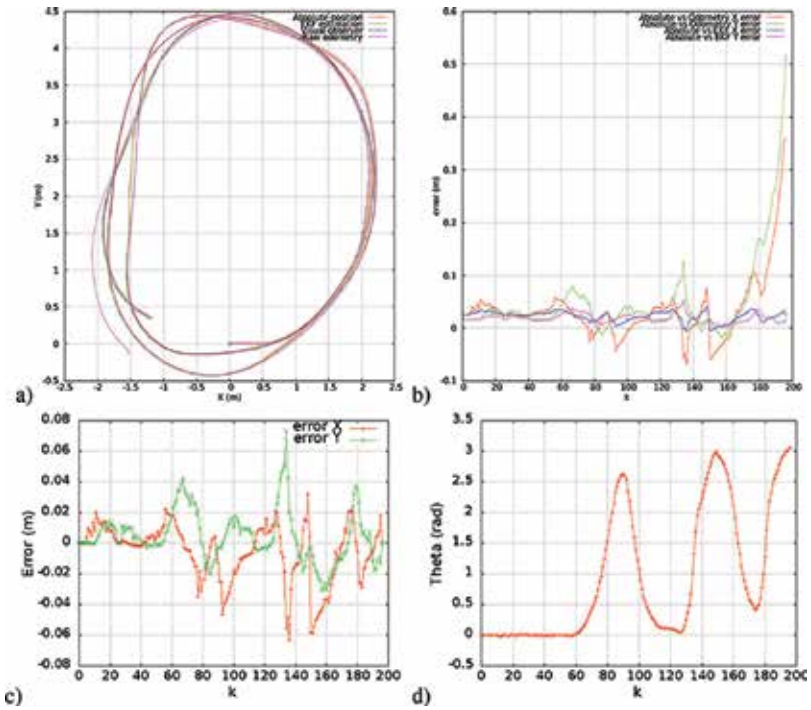


Figure 4. Positions and errors. (a) Cartesian positions. (b) Errors vs. global reference. (c) EKF's Cartesian errors' convergence. (d) EKF's angle error convergence.

http://dx.doi.org/10.5772/intechopen.79130

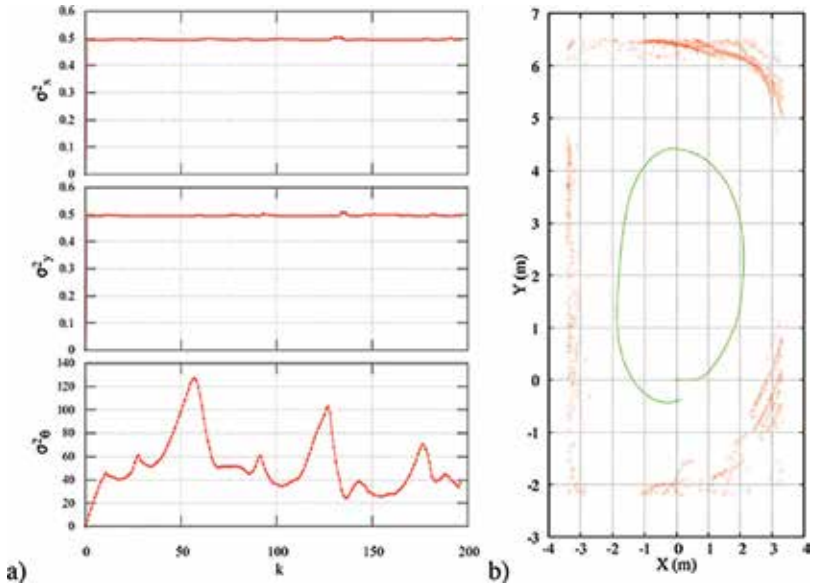


Figure 5. Errors' convergence behavior. (a) EKF variances over time. (b) Key-point map's divergence using state vectors.

Figure 4b shows the dead-reckoning and the EKF Cartesian absolute errors, taken as main reference for the visual global reference system. As for the direct dead-reckoning measurements, the absolute error grows exponentially, where the position observation starts diverging before the robot reaches the third turn. As for the EKF model, the Cartesian error w.r.t. the global reference does not diverge but preserves bounded error magnitudes.

As for **Figure 4c** and **d**, the EKF's Cartesian and angular absolute errors w.r.t. the global visual tracker are shown. In **Figure 4d**, the local minimums and maximums determine the Cartesian regions where the robot performed its turns.

Finally, **Figure 5a** shows the covariance error behavior obtained at each control loop during the EKF recursive calculations. **Figure 5b** is a mapping of the measured key-points registered using the state vector (posture) of a robot's turn to illustrate the map's divergence.

5. Conclusion

This chapter presented a visual odometry scheme for a trinocular divergent visual system that was combined with an EKF for visual odometry estimation. The proposed trinocular geometric model observer geometrically combined adjacent radial views. About 20% of adjacent multi-view overlapping data allowed inference of small volumes of depth information. In measuring 3D key-points, the X-axis metrical error was reported to be lower than 7 cm, with error less

than 10 cm in $+Y$ component and less than 3 cm in $-Y$ (vertical). Likewise, we found an averaged Z-axis error less than 5 cm (depth). Such errors were mostly produced by the angular divergence of the object w.r.t. the central camera, rather than linear distances. Indoor experiments, measuring distances up to 10 m were developed. In addition, a set of experimental results in convergent robot's course gave closed loops, and as the robot moved, the trinocular sensor incrementally stored environmental 3D key-points.

The robot's trajectory was obtained by different comparative approaches conducted in this study. The postures were measured by an external visual global reference system, which was the main reference system to be compared with. The robotic platform's kinematics was modeled in terms of a dead-reckoning approach. The direct and the inverse solutions were combined to produce a recursive linearized control model and this was used as the prediction model for EKF estimator. The dead-reckoning robot system was deployed to infer the robot's postures using directly the four encoders' readings, with good results obtained only for very short paths. As a comparative perspective, using only the 4WD dead-reckoning system the posture exponentially diverged.

We found bounded Cartesian error for this 4WD robot by deploying the EKF. The trinocular 3D key-points were used as inputs of the visual odometry model that inferred the robot's displacements by geometrical triangulations.

Author details

Edgar Alonso Martínez-García[1]* and Luz Abril Torres-Méndez[2]

*Address all correspondence to: edmartin@uacj.mx

1 Laboratorio de Robótica, Universidad Autónoma de Ciudad Juárez, Mexico

2 Robotics Active Vision Group, CINVESTAV, Campus Saltillo, Mexico

References

[1] Milella A, Reina G. Towards autonomous agriculture: Automatic ground detection using trinocular stereovision. Sensors. 2012;**12**(9):12405-12423

[2] Gwo-Long L, Chi-Cheng C. Acquisition of translational motion by the parallel trinocular. Information Sciences. 2008;**178**:137-151

[3] Tae Choi B, Kim J-H, Jin Chung M. Recursive estimation of motion and a scene model with a two-camera system of divergent view. Pattern Recognition. 2010;**43**(6):2265-2280

[4] Jaeheon J, Correll N, Mulligan J. Trinocular visual odometry for divergent views with minimal overlap. In: IEEE Workshop on Robot Vision; 2013. pp. 229-236

[5] Dellaert F, Kaess M. Probabilistic structure matching for visual slam with a multi-camera rig. Computer Vision and Image Understanding. 2010;**114**(2):286-296

[6] Lee C-H, Kwon S, Lee J-h, Lim Y-C, Lee M. Improvement of stereo vision-based position and velocity estimation and tracking using a stripe-based disparity estimation and inverse perspective map-based extended kalman filter. Optics and Lasers in Engineering. 2010; **48**(9):859-886

[7] Harmat A, Wang DWL, Sharf I, Waslander SL, Tribou MJ. Multi-camera parallel tracking and mapping with non-overlapping fields of view. The International Journal of Robotics Research. 2015;**34**(12):1480-1500

[8] Seet G, Wei M, Han W. Efficient visual odometry estimation using stereo camera. In: 11th IEEE International Conference on Control and Automation; 2014. pp. 1399-1403

[9] Kanade T, Badino H, Yamamoto A. Visual odometry by multi-frame feature integration. In: IEEE International Conference on Computer Vision Workshops; December 2013. pp. 222-229

[10] Liang Z, Xu X. Improved visual odometry method for matching 3D point cloud data. In: 33rd Chinese Control Conference; July 2014; pp. 8474-8478

[11] Llorca DF, Parra I, Sotelo MA, Ocaña M. Robust visual odometry for vehicle localization in urban environments. Robotica. 2010;**28**:441-452

[12] Rendon-Mancha JM, Fuentes-Pacheco J, Ruiz-Ascencio J. Visual simultaneous localization and mapping: A survey. Artificial Intelligence Review. 2015;**43**(1):5581

[13] Neira J, Cisneros R, Lavion JE, Hernandez E, Ibarra JM. Visual slam with oriented landmarks and partial odometry. In: 21st International Conference on Electrical Communications and Computers; February–March 2011. pp. 39-45

[14] Hong S, Ye C. A pose graph based visual slam algorithm for robot pose estimation. In: World Automation Congress; August 2014. pp. 917-922

[15] Williams B, Reid I. On combining visual slam and visual odometry. In: IEEE International Conference on Robotics and Automation; May 2010. pp. 3494-3500

[16] Huitl R, Schroth G, Kranz M, Steinbach E, Hilsenbeck S, Moller A. Scale-preserving long-term visual odometry for indoor navigation. In: International Conference on Indoor Positioning and Indoor Navigation; November 2012. p. 110

[17] Martínez-García EA, Lerin E, Torres-Cordoba R. A multi-configuration kinematic model for active drive/steer four-wheel robot structures. Robotica. 2015;**33**:2309-2329. Cambridge University Press

[18] Martinez-Garcia E, Mar A, Torres-Cordoba T. Dead-reckoning inverse and direct kinematic solution of a 4W independent driven rover. In: IEEE ANDESCON; 15–17 September 2010; Bogota, Colombia

[19] Martinez-Garcia EA, Torres-Cordoba T. 4WD skid-steer trajectory control of a rover with spring-based suspension analysis. In: International Conference in Intelligent Robotics and Applications (ICIRA2010); 10–12 November 2010; Shanghai, China

Chapter 10

IntelliSoC: A System Level Design and Conception of a System-on-a-Chip (SoC) to Cognitive Agents Architecture

Diego Ferreira, Augusto Loureiro da Costa and
Wagner Luiz Alves De Oliveira

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79265

Abstract

This chapter presents a system level design and conception of a System-on-a-Chip (SoC) for the execution of cognitive agents. The computational architecture of this SoC will be presented using the cognitive model of the concurrent autonomous agent (CAA) as a reference. This cognitive model comprises three levels that run concurrently, namely the reactive level, the instinctive level and the cognitive level. The reactive level executes a fast perception-action cycle. The instinctive level receives perceptions from and sends the active behavior to the reactive level, and using a Knowledge Based System (KBS) executes plans by selecting reactive behaviors. The cognitive level receives symbolic information from the instinctive level to update its logical world model, used for planning and sends new local goals to instinctive level. Thus, this work proposes a novel SoC whose architecture fits the computational demands of the aforementioned cognitive model, allowing for fast, energy-efficient, embedded intelligent applications.

Keywords: cognitive agents, intelligent robots, mobile robots

1. Introduction

Every entity that can perceive its environment and perform actions upon it can be called an *agent* [1]. When this process is achieved using knowledge about the environment, then the agent is a *cognitive agent* [2]. Cognition, according to [3], is a process that allows to a system to robustly behave adaptively and autonomously, with anticipatory capabilities. The authors proceed by classifying cognitive systems in two broad classes, namely the cognitivist and the

emergent systems. Inside the cognitivist class goes systems that relies on symbolic representation and information processing. In the second class, the emergent systems, are connectionist, dynamical and enactive systems.

There are aspects of cognitive agents that remain invariant in time and over different tasks. These aspects generally include the short-term and long-term memories where knowledge is stored, the knowledge representation structure and the processes that performs over the previous elements (possibly changing its contents, like learning). The aspects cited above are comprised by a cognitive architecture [4].

An example of architecture for cognitive agents is the concurrent autonomous agent (CAA), an autonomous agent architecture for mobile robots that has already proven to be very powerful [5–7]. Three parallel levels compose this architecture: the reactive, the instinctive and the cognitive levels. The first is responsible for executing the perception-action cycle, the second uses a knowledge based system (KBS) to select behaviors in the reactive level, and the third also uses a KBS, but for planning.

In [8] the CAA was embedded in a microcontrollers network specially designed to fit its cognitive architecture. The intention of the authors was to optimize the performance of the agent for an embedded environment, allowing it to be physically embedded into a mobile robot. In this work a step forward is given in that direction, since its main objective is to design a System-on-a-Chip (SoC) dedicated to the execution of the CAA.

Hardware design for cognitivist systems (using [3] aforementioned classification) is not a recent concern. The first paper about the Rete matching algorithm [9] already presents a low-level (assembly) implementation of the matching algorithm for production systems. Years later, [10] designed a reduced instruction set computer (RISC) machine for OPS production systems, focusing on optimizing the branch prediction unit for the task. A more recent approach by [11] proposes a parallelization strategy to use the parallel processing power of graphics processing units (GPUs) for Rete pattern matching.

Still searching for dedicated hardware for the *Rete* algorithm, [12] present a special purpose machine that is connected to a host computer as a support for agents execution in software. The authors also shown that increasing the number of inputs of the beta nodes in the network generated by the algorithm enhances the performance of the proposed hardware: using four or five inputs in a join node of the beta network allowed for a 15–20% increase in performance. For the operation of the processor the network is compiled into an inner representation and loaded into an external RAM. A Token Queue Unit stores partial matches, controlled by a microprogram of the Control Unit and by a Match Unit.

The authors in [13] propose a processor for the execution of production systems aiming applications of real-time embedded general artificial intelligence. The production system executed by the processor is the *Street* system. The processor, named *Street Engine*, interprets RISC instructions from its ISA, which is composed by only one type of instructions: a form of parallel production rules. Hence, instead of being executed sequentially, the instructions are executed in parallel by hardware units named producers. The producers are interconnected in a network-on-a-chip (NoC), and each one has only one associated production. The knowledge

representation language employed by the Street Engine is the *Street Language*, designed to map a variant of the *Rete* algorithm to the hardware. The street engine is controlled by events, where each producer stores a subset of the working memory (corresponding to the alpha memories in the *Rete* algorithm) and changes in the working memory of one producer causes changes in other memories of producers affected by the change.

Deviating from production systems, a processor oriented to the real-time decision making of mobile robots is proposed by [14]. The processor comprises four search processors with 8 threads each for trajectory planning and a reinforcement learning acceleration module for obstacle avoidance. The search processors consist of a 6-stage pipeline that uses a three-level cache to implement a transposition and avoid redundant computation between threads. The reinforcement learning accelerator, in turn, uses a 2D array of processing elements to implement a SIMD architecture. A penalty is considered when a search processor tries to plan a trajectory that collides with an obstacle.

The common theme among these works is that they are concerned with expert (production) systems. According to [4], while cognitive architectures are intended to perform successfully in a broad range of domains, expert systems have a narrower range of applications. The authors then continues by saying that cognitive architectures "offers accounts of intelligent behaviors at the system level, rather than at the level of component methods designed for specialized tasks". Therefore, the design of a SoC for cognitive systems can accomplish the optimized performance of a dedicated low-level machine while maintaining its powerful intelligent behavior, which justifies the importance of this work.

This chapter is divided as follows. In Section 2, the CAA is presented, with its levels explained. Section 3 then exposes how knowledge is represented and inference is performed by the CAA KBS (in both instinctive and cognitive levels). Sections 4 and 5 explains the *Rete* and *Graphplan* algorithms, laying the basis for the following sections, which describes the hardware architectures proposed to implement these algorithms. First, in Section 6, the overall architecture is described. Then, Section 7 describes the *Rete* RISC machine. And in Section 8 the cognitive module is presented. Section 9 shows the results of simulations and some final considerations are presented in Section 10.

2. The concurrent autonomous agent (CAA)

The architecture of the concurrent autonomous agent (CAA) was inspired by the generic model for cognitive agents. This model comprises three levels: the reactive level, the instinctive level and the cognitive level [15]. The CAA levels are shown in **Figure 1** [5, 6]. As can be seen in this figure, the reactive level is responsible for interacting with the environment. It contains reactive behaviors that enables the agent to perform a fast perception-action cycle. The perceptions are sent to the instinctive level which, in turn, uses it to update the state of the world that it maintains in a KBS. It also uses this information to update the logical model of the world in the cognitive level and select the reactive behavior in the reactive level. Finally, the cognitive level does the planning, sending local goals to the instinctive level, that will coordinate the execution of the plan [6].

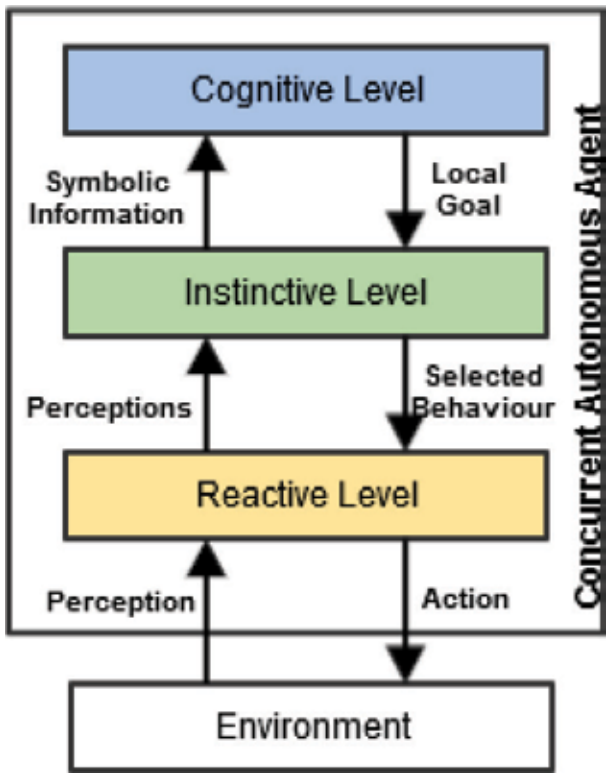


Figure 1. Concurrent autonomous agent architecture.

The level that interacts with the environment executing a fast-perception-action cycle is the reactive level. It consists of a collection of reactive behaviors that determines the interaction of the agent with the environment. Only one behavior can be active at a time, and the instinctive level makes the selection. The architecture used in [8, 16] consists of a kinematic position controller for the omnidirectional robot *AxéBot*. The reactive behaviors were implemented based on the embedded kinematic controller. The behaviors implemented were simple: there is one behavior for each cardinal direction, i.e., selecting the behavior corresponds to selecting the direction (relative to the orientation of the robot) in which one wishes the robot to move onto.

The instinctive level, as the reactive level, is identical to the one purposed in [8]: its reasoning mechanism consists of a KBS that executes a plan generated by the cognitive level, sending symbolic information about the environment to the latter. The plans are executed by coordinating behavior selection in the reactive level, which sends the perceptions to this level.

The cognitive level also uses a KBS as automatic reasoning method. Its facts base consists of a logical model of the world. The inference engine is multi-cycle, meaning that it keeps running independent of the update of the facts base by the instinctive level. This level does the planning, coordinating the instinctive level for the execution of the plans. It is not used in this work.

http://dx.doi.org/10.5772/intechopen.79265

3. Knowledge-based systems (KBS)

The CAA uses a KBS in its two higher levels: the instinctive and the reactive. Its inner structure is shown in **Figure 2** [6].

All knowledge of the agent is stored in the *facts base*. The elements of the facts base use the format *logic*(*object attribute value*) to represent elements in the world. The facts base contains the states of the agent and of the environment in the KBS of the instinctive level and the logical model of the world in the cognitive level one.

The format above is used also to form the premises of the rules in the *rules base*. But in this case they are restrictions or specifications that the facts in the facts base must met in order to *fire* the rule and execute its consequence, which may contain instructions on how to modify its own facts base, or the facts base of another level (adding, removing or updating facts). Additionally, in the rules syntax the fields (*object*, *attribute* and *value*) of the logical elements may contain variables, which are denoted by a symbol preceded by a interrogation (?) token.

The rules may also contain filters to further specify restrictions on the values of the fields in the premise elements. Filters have the format *filter*(*operator parameter*1 *parameter*2), where *parameter*1 and *parameter*2 are variables or symbols present in some premise elements and *operator* tells how they must compare for the rule to be fired.

The inference engine uses the *Rete* matching algorithm to search the rules base for rules that are enable to fire by the facts in the facts base without having to iterate through both bases. The enabled rules then form the *conflict set*, and the inference engine must decide which rules in this sets should be fired. Finally, the inference engine executes the consequence of the fired rules and restart the process. This is the *forward chaining* algorithm.

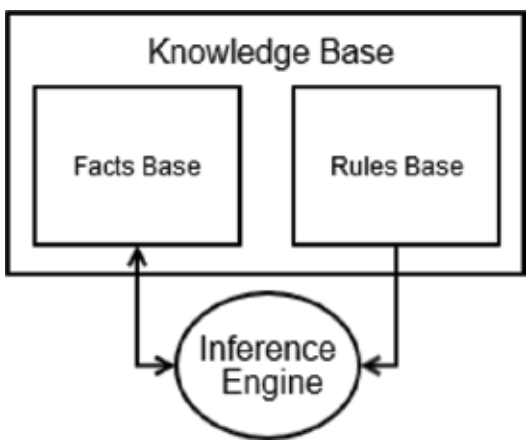


Figure 2. Block diagram of a KBS.

4. The Rete algorithm

The *Rete* algorithm is used in the inference engine of a KBS to efficiently match rules premises with the facts without the need of looping through both rules and facts bases at each inference cycle, greatly improving performance. It was proposed by Charles Forgy in 1979, in its doctoral thesis [17]. Its name, *rete*, is latin for *network*, because of how it organizes the information in the knowledge base.

The *Rete* algorithm starts by constructing a network of nodes and memories based on the premises of the rules and eventual filters they may contain. This network is divided in two parts: the alpha and the beta networks.

The alpha network is composed by the following nodes:

- a *Root Node*, which is the entry point for new facts;
- *Constant Test Nodes* (CTN), which checks whether the non-variable (constant) fields of premises matches the corresponding ones in the current fact; and
- *Alpha Memories* (AM), that stores facts that successfully passed through *constant test nodes*.

The beta network, in turn, have:

- *Join Nodes* (JN), where a set of test are performed to check variable binding consistency;
- *Beta Memories* (BM), which conjunctively "accumulates" facts that passed the corresponding JN tests in *tokens*, which are partial matches to specific premises; and
- *Production Nodes*, which are terminal nodes for full matches.

5. The Graphplan algorithm

The cognitive level of the CAA contained a classical planner that used a KBS to perform a state-space search in order to generate the plan. This approach may produce an explosion in the number of states that, in turn, may overwhelm the memory capacity of the computational system that executes it. In the case of the present work, where a SoC is supposed to run the algorithm, this issue becomes specially expressive. The alternative considered here is the utilization of a planning graph algorithm search instead, which is known as *Graphplan*.

The Graphplan algorithm uses a propositional representation for states and actions, and its basic idea to reduce the search graph is to structure it as a sequence of layers comprising sets of states, or *propositions*. Following a state (proposition) layer there is an inclusive disjoint of actions, which is then followed by another propositional layer, and so on, alternately.

For a mathematical formulation of the *Graphplan* algorithm, one should first define mathematically a general planning problem. Let $\mathcal{P} = (\Sigma, s_j, g)$ be a planning problem, where:

http://dx.doi.org/10.5772/intechopen.79265

- $\Sigma = (S, A, \gamma)$ is the problem domain, with S being the set of states, A the set of actions and $\gamma = S \times A \rightarrow S$ is a state transformation application;
- $s_j \in S$ is the initial state; and
- g is the goal state.

An action $a \in A$ is composed by two sets of propositions: $prencond(a)$, its preconditions, and $effects^{(a)=effects^{+}(a) \cup effects^{-}(a)}$, its effects, where $effects^{+}(a)$ is the set of positive propositions and $effects^{-}(a)$ is the set of negative propositions in the effects of the action. For an action a to be applicable in a given state s, one should have $precond(a) \subseteq s$, and the new state would be given by $\gamma(s, a) = (s - effects^{-}(a)) \cup effects^{+}(a)$.

Now back to *Graphplan*, given the action layer A_j and the propositional layer P_{j-1} preceding it, the former contains all actions a such that $precond(a) \subseteq P_{j-1}$ and the later is composed by all propositions p such that $p \in P_{j-1}$. Also, there are three types of edges:

- connecting a proposition $p \in P_{j-1}$ to an action $a \in A_j$;
- connecting an action $a \in A_j$ to a proposition $p \in P_{j-1}$, such that $p \in effects^{+}(a)$ (positive arc); and
- connecting an action $a \in A_j$ to a proposition $p \in P_{j-1}$, such that $p \in effects^{-}(a)$ (negative arc).

Two actions $a_1, a_2 \in A_j$ are called *independent* if $effects^{-}(a_1) \cap \left(precond(a_2) \cup effects^{+}(a_2)\right) = \emptyset$ and $effects^{-}(a_2) \cap \left(precond(a_1) \cup effects^{+}(a_1)\right) = \emptyset$; otherwise they are said to be *dependent*.

If two actions are dependent, they are said to be *mutually exclusive*, or *mutex* for short. Another situation that makes two actions in the same layer to be mutex is when a precondition of one of them is mutex with a precondition of the other. And two propositions p and q are mutex if for all $a_1 \in A_j$ such that $p \in effects^{+}(a_1)$ is mutex with every $a_2 \in A_j$ such that $q \in effects^{+}(a_2)$, and $\nexists a \in A_j$ such that $p, q \in effects^{+}(a)$.

The algorithm works as follows. It first expands the planning graph, generating the direct layered graph described before, with the successive proposition and action layers. Then when a propositional layer P_j contains g, the expansion stops and the graph is searched backwards for a sequence of sets of non-mutex actions. The plan is then extracted from this sequence.

An important difference between the algorithm used here and the one from the standard approach is that here we use a KBS to represent knowledge and execute the inference. So the expansion step The pseudo-code for the expansion step is given in Algorithm 1.

Algorithm 1 Planning graph expansion

1: **procedure** EXPAND(s_i) ▷s_i: i-th state layer

2: $A_{i+1} \leftarrow KBS.InferenceCycle(s_i, A)$ ▷A: action profiles

3: $s_{i+1} \leftarrow \cup A_{i+1}.effects^{+}$

4: $\mu A_{i+1} \leftarrow \left\{(a,b) \in A^2_{i+1}, a \neq b \,|\, Dependent(a,b) \vee \exists (p,q) \in \mu s_i : p \in preconds(a), q \in preconds(b)\right\}$

5: $\mu s_{i+1} \leftarrow \left\{(p,q) \in s^2_{i+1}, p \neq q \,|\, \forall (a,b) \in A^2_{i+1} : p \in effects^+(a) \wedge q \in effects^+(b) \rightarrow (a,b) \in \mu A_{i+1}\right\}$

6: **end procedure**

Whenever the set of WME in g - the goal state - is contained in a given state layer s_i, a recursive procedure must be executed to search for sets of non-mutex actions in each layer that could have produced all the WMEs in the goal state. This procedure is composed by the functions Search and Extract.

Algorithm 2 Search for non-mutex actions.

1: **procedure** SEARCH(g, π_i, i)

2: **if** $g = \emptyset$ **then**

3: $\Pi \leftarrow Extract(\cup\{preconds(a) \,|\, \forall a \in \pi_i\}, i-1)$

4: **if** $\Pi = Failure$ **then**

5: **return** *Failure*

6: **end if**

7: **return** $\Pi.\pi_i$

8: **else**

9: select any $p \in g$

10: $resolvers \leftarrow \left\{a \in A_i \,|\, p \in effects^+(a) \wedge \forall b \in \pi_i : (a,b) \notin \mu A_i\right\}$

11: **if** $resolvers = \emptyset$ **then**

12: **return** *Failure*

13: **end if**

14: non-deterministically choose $a \in resolvers$

15: **return** Search($g - effects^+(a)$, $\pi_i \cup \{a\}$, i)

16: **end if**

17: **end procedure**

Algorithm 3 Extract a plan.

1: **procedure** EXTRACT(g, i)

2: **if** $i = 0$ **then**

http://dx.doi.org/10.5772/intechopen.79265

3: **return** $\emptyset$

4: **end if**

5: $\pi_i \leftarrow Search(g, \emptyset, i)$

6: **if** $\pi_i \neq Failure$ **then**

7: **return** π_i

8: **end if**

9: **return** *Failure*

10: **end procedure**

6. The proposed architecture

Figure 3 shows an overview of the SoC architecture for cognitive agents proposed in this work. As mentioned before, the cognitive model of the CAA (**Figure 1**) was used as a base model for

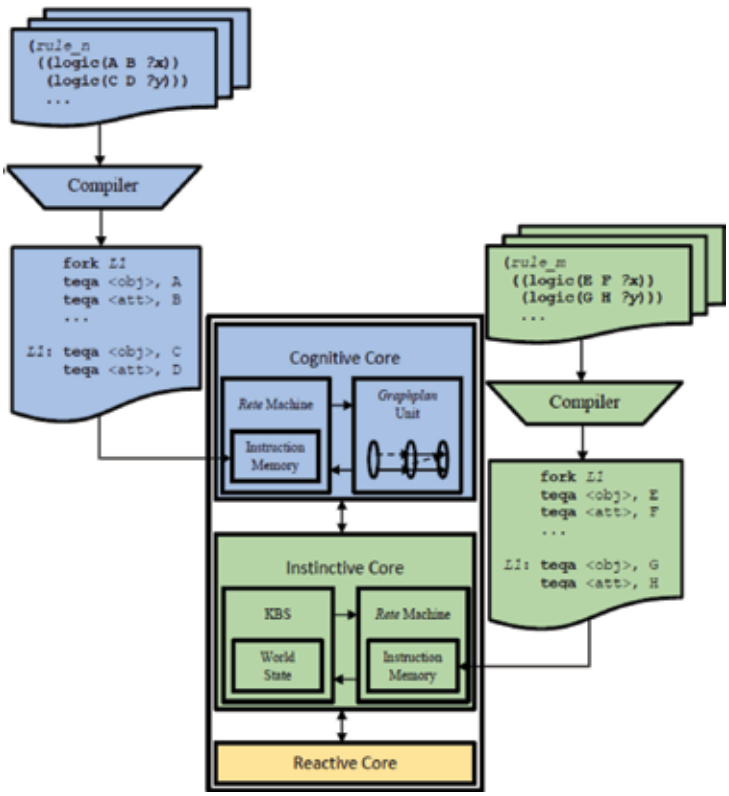


Figure 3. Block diagram of the proposed SoC.

the development of this architecture, and hence it can be seen in the image that each CAA level has a corresponding computational module in the SoC.

The modules were designed according to the task executed by the corresponding CAA level. Still referring to **Figure 3**:

- the cognitive level is implemented by a *Graphplan* module, responsible for the execution of graph planning algorithms and a module dedicated to the *Rete* algorithm. The later is used by the former for the SBC-based state space expansion;
- the instinctive level comprises only a *Rete* module that is applied to coordinate with the reactive level the execution of a plan; and
- the reactive level, which in fact interacts with the environment, must be as general as possible, and is implemented by a general purpose CPU.

7. The RISC Rete Machine

In this section, the proposed processor system level architecture is described. As it was stated in the introduction, this processor is part of the design of a SoC for cognitive agents. The idea is to have this processor working with another unit for planning in the cognitive level and with a plan execution unit in the instinctive level. The knowledge bases should first be compiled into the application specific ISA of the *Rete* processor and then downloaded into its instruction memory.

The system level architecture here presented is a RISC application specific processor (ASP) whose special purpose ISA was inspired on the description of the *Rete* algorithm given in [9], the first paper written about the algorithm. The author uses an assembly-like set of instructions to describe the operation of the algorithm. But they serve only as guidelines for a high-level implementation described afterwards in that paper; no hardware implementations are presented.

Inspired by the aforementioned (pseudo-)instructions presented in [9], this work purposes an actual machine for the execution of the *Rete* matching algorithm whose ISA implements a modified version of the pseudo-instructions presented in Forgy's seminal paper.

The overall processor architecture is shown in **Figure 4**. The alpha and beta memories are pre-allocated at compiling time.

The rules are compiled in a sequence of these instructions instead of being used for the creation of the *Rete* tree in memory. The alpha and beta memories will still exist but the nodes (constant test and join nodes) will be implemented by instructions (**Figure 4**).

The new fact is stored in a register together with a bit indicating whether it is being added or deleted. The instructions arguments and operation are detailed below:

- FORK <label>: Represents a branch in the tree, with one of the nodes represented by a node instruction immediately after it and the other at the instruction address given by <label>. This address is stacked and the next instruction is fetched. The instruction address corresponding to <label> is popped from stack when a mismatch occurs and the program jumps to it. If the stack is empty, the match failed.

http://dx.doi.org/10.5772/intechopen.79265

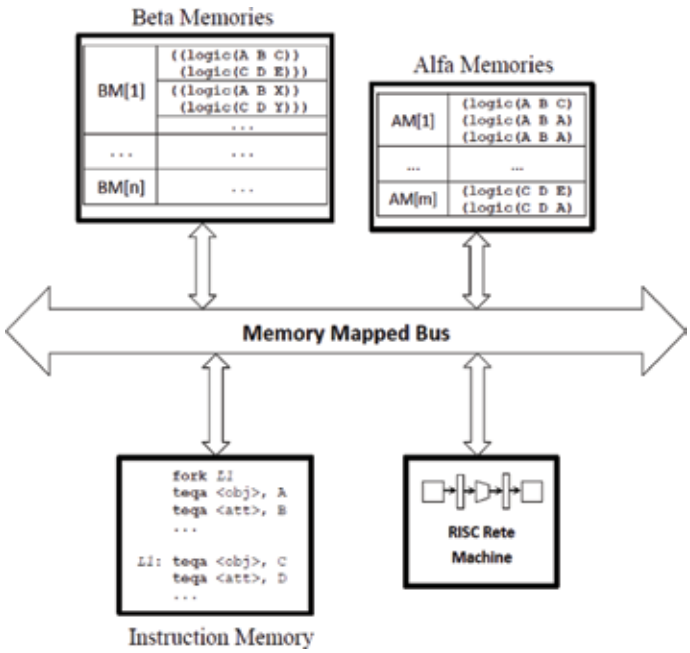


Figure 4. Architecture of the *Rete* processor.

- TEQA <field>, <constant>: Implements a CTN where <field> is the field to be tested (object, attribute or value) of the fact register and <constant> is the value this field must be equal to.
- FILTER <field1>, <field2>, <comparison>: Compares two fields inside a fact using a given comparison operation. If the comparison fails, so does the match.
- MERGE <parent-bm>, <bm>, <am>, <next-join>: Saves in registers the addresses of its parent memories and of the next MERGE instruction. Also, it updates an alpha memory in a right activation.
- TEST <field1>, <premise>, <field2>, <comparison>: Deals with left and right activations of the JN. It triggers an interruption, jumping to a pre-programmed routine that runs through the memories testing whether or not the <field1> compares to <field2> of <premise>-th premise. The <comparison> operation is given in the field comparison.
- JOIN <lbl>: jumps to a JN (MERGE instruction) defined previously in the code, on a right activation.
- TERM <rule>, <nsubs>: Represents production nodes. It saves the address of the consequence of the matched rule in a register, for further use. <nsubs> is the number of substitutions this rule has, so that it can jump to the last one (for popping the test stack) in the case where the current fact is to be excluded instead of added.
- SUBST <p1>, <f1>, <p2>, <f2>, <lst>: Uses the matched token to create substitutions for the variable in the consequence. The pairs (<p1>, <f1>) and (<p2>, <f2>) are "coordinates" of occurrences of the same variable in the premises and the consequence, respectively.

<lst> indicates whether it is the last substitution for that match or not. If it is, the test stack must be popped to proceed with interrupted tests.

8. The cognitive module

Figure 3 shows that the computational module associated with the cognitive level of the CAA (and from now on referred to as cognitive module) contains a *Graphplan* module that communicates with a *Rete* module, whose architecture was presented in the previous section. The objective of the cognitive module is to solve planning problems using graph planning algorithms. The *Graphplan* algorithm was used as an inspiration for the conception of the architecture of this module.

This module needs to be generic enough to execute not only the *Graphplan* algorithm, but also algorithms based on planning graphs or that uses *Graphplan* as heuristic function. Also, it should have two execution modes: the search mode, where a solution for the planning problem is searched, and the monitor mode, the monitors the execution of found plans.

The system level model of this module consists of two general purpose processors with individual instruction memory (i.e. executing independent programs) and a *Rete* module communicating with each other through a bus. **Figure 5** illustrates this architecture.

The taxonomy of the task this module should execute justifies this architecture. During planning, two main processes should be executed: the expansion of the states and actions graph, and the recursive search for a solution whenever the goal state is found in the graph. With the architecture described it is possible to execute these processes in parallel.

With this approach, when the goal state is found, the solution search can be started while the graph expansion goes on. Hence, even if the found goal set do not produce a solution, the expansion made progress in parallel, until another valid goal set is found, when the search is triggered again. The dynamics of *Graphplan* was modified for this module: a KBS is responsible for calculating the applicable actions; this KBS is implemented by the *Rete* module. The sequence diagram in **Figure 6** illustrates this operation.

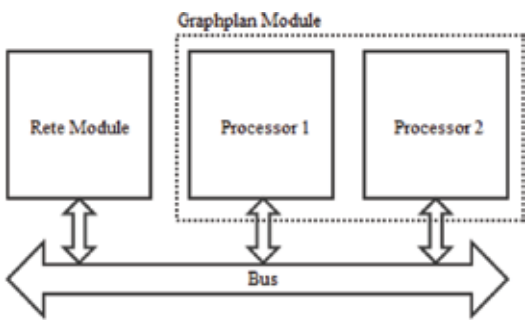


Figure 5. Block diagram of the cognitive module.

http://dx.doi.org/10.5772/intechopen.79265

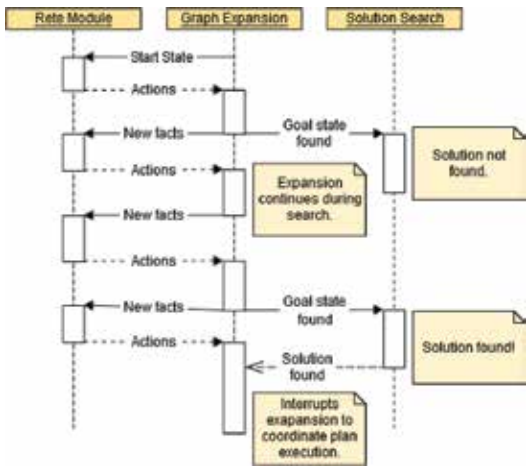


Figure 6. Sequence diagram of the cognitive module operation.

9. Case study

The architecture were simulated using a program written in the Scala programming language, using array structures for the instruction, alpha and beta memories, lists for fork and JN test stacks and variables for registers (program counter, direction flags, stacks counters, alpha and beta indices etc.). The dynamics of the program was dictated by the way the instructions changed the program counter.

9.1. Domain definition

The architecture will be validated using the block world domain example, from [1]. **Figure 7** shows the knowledge base that is going to be used in the simulation. The same filter shown in the *Move* rule could be present in the *MoveToTable* rule, but it was omitted for simplicity. In spite of the fact that the consequences will not be used here (only the matching procedure is important), one should notice the *add* and the *rem* symbols. Those are simply instructions on how to modify the facts base in case of a match.

9.2. Simulated tests: *Rete* module

The knowledge base shown in **Figure 7** is then compiled into the program presented in **Figure 8**. This program is the input for the simulator, in the form of an array of instruction objects (objects from an instruction class defined inside the simulator code). It then waits for a new fact to come in and then executes the program once for that fact, changing the memories accordingly.

The tests performed consisted of feeding some facts that are known to cause a match, one by one, to the simulator and check whether the system detects that match. The facts $(logic(C, on, A))$, $(logic(C, type, block))$ and $(logic(C, state, clear))$ must match the premise of the rule *MoveToTable*.

Rule Base

```
(Move(?b,?x,?y)
   (if     (logic (?b on ?x))
           (logic (?b state clear))
           (logic (?y state clear))
           (logic (?b type block)))
           (logic (?y type block)))
   (filter (!= ?b ?x))
   (then   (add (logic (?b on ?y))
                (logic (?x state clear)))
           (rem (logic (?b on ?x))
                (logic (?y state clear)))))


(MoveToTable(?b,?x)
   (if     (logic (?b on ?x))
           (logic (?b state clear))
           (logic (?b type block)))
   (then   (add (logic (?b on table))
                (logic (?x state clear)))
           (rem (logic (?b on ?x)))))
```

Figure 7. Knowledge base for the block world domain example.

Figure 9 shows the output of the simulator after feeding it with the fact $(logic(C, on, A))$. This output contains all the instruction executed by the processor for the given fact. As the entry point of the network is the root node and it has three CTNs as children, a FORK is always processed first. For the current fact, the forked address is only taken after the fact is stored in $AM1$ (and consequently in $BM1$ too, since the parent BM is a dummy one), when the JN test fails ($pc = 5$) due to the absence of facts in $AM2$. It is noteworthy that the instruction at forked address is another fork, because the root node has three children. In the last CTN ($pc = 33$) the fork stack is empty, and as the test failed, the program finishes.

For $(logic(C, type, block))$ the processing mechanism is the same, but the execution path is different, since the fact is going to a different alpha memory.

When $(logic(C, state, clear))$ is added, a match occurs ($pc = 22$), as can be seen in **Figure 10**. In this output it is possible to see the JN tests being stacked once a partial match is found ($pc = 5$). After that ($pc = 8$), a test fails, but the previously stacked test is not popped yet: there is a FORK at $pc = 6$, and forks have priority. Also, in this execution four substitutions take place ($pc = 23$ to 26).

Finally, **Figure 11** shows the output of the simulator for the exclusion of the fact $(logic(C, on, A))$. The procedure for exclusion starts as the same as the one for addition: the instructions guide the traverse of the tree looking for a match. The difference is that no changes are made to the memories. Instead, for every activation or match caused by the input fact, the index of the corresponding memory and its position inside that memory are stacked. At the end of the execution, when there are no more forks or tests stacked, the exclusion stack is pop and the elements of the memories given by the indices stored in it are deleted.

http://dx.doi.org/10.5772/intechopen.79265

```
        fork    ctn_state_clear
        teqa    attribute, "on"
        filter  object, value, "!="
        merge   dummy_bm, am1, join2
join2:  merge   bm1, am2, jfrk
        test    obj, 0, obj, "=="
jfrk:   fork    join6
join3:  merge   bm2, am2, join4
        test    obj, 1, obj, "!="
join4:  merge   bm3, am_type_block, join5
        test    obj, 0, obj, "=="
join5:  merge   bm4, am_type_block, join6
        jtest   obj, 2, obj, "=="
        term    Move, 6
        subst   (0, object), (0, object), last=false
        subst   (0, object), (2, object), last=false
        subst   (0, value), (1, object), last=false
        subst   (0, value), (2, value), last=false
        subst   (2, object), (0, value), last=false
        subst   (2, object), (3, object), last=true
join6:  merge   bm2, am_type_block, NULL
        jtest   obj, 0, obj, "=="
        term    MoveToTable, 4
        subst   (0, object), (0, object), last=false
        subst   (0, object), (2, object), last=false
        subst   (0, value), (1, object), last=false
        subst   (0, value), (2, value), last=true
ctn_state_clear:
        fork    ctn_type_block
        teqa    attribute, "state"
        teqa    value, "clear"
        fork    join23
        join    join2
join23: join    join3
ctn_type_block:
        teqa    attribute, "type"
        teqa    value, "block"
        fork    join456
        join    join4
join456:
        fork    join56
        join    join5
join56: join    join6
```

Figure 8. Code for the *Rete* network of the block world example.

```
================================
(+) (C,on,A):
================================
pc=0>   fork    27
pc=1>   teqa    attr, on
pc=2>   filter  obj, val, "!="
pc=3>   merge   dummy, bm1, am0, 4
pc=4>   merge   bm1, bm2, am1, 6
pc=5>   jtest   obj, 0, obj, "==" // TEST FAILED: FORKING
pc=27>  fork    33
pc=28>  teqa    attr, state // TEST FAILED: FORKING
pc=33>  teqa    attr, type // TEST FAILED
```

Figure 9. Output of the simulator for $(logic(C, on, A))$.

```
================================
(+)(C,state,clear):
================================
pc=0>   fork    27
pc=1>   teqa    attr, on // TEST FAILED: FORKING
pc=27>  fork    33
pc=28>  teqa    attr, state
pc=29>  teqa    val, clear
pc=30>  fork    32
pc=31>  join    4
pc=4>   merge   bm1, bm2, am1, 6
pc=5>   jtest   obj, 0, obj, "==" // R.A. OK: PUSH TEST
pc=6>   fork    20
pc=7>   merge   bm2, bm3, am1, 9
pc=8>   jtest   obj, 1, obj, "!=" // TEST FAILED: FORKING
pc=20>  merge   bm2, bm6, am2, -1
pc=21>  jtest   obj, 0, obj, "==" // L.A. OK: PUSH TEST
pc=22>  term    MoveToTable, 4 // ** (+) MATCH **
pc=23>  subst   (0,0), (0,0), last=false
pc=24>  subst   (0,0), (2,0), last=false
pc=25>  subst   (0,2), (1,0), last=false
pc=26>  subst   (0,2), (2,2), last=true // POP TEST
pc=21>  jtest   obj, 0, obj, "==" // TEST FAILED: FORKING
pc=32>  join    7
pc=7>   merge   bm2, bm3, am1, 9
pc=8>   jtest   obj, 1, obj, "!=" // TEST FAILED: FORKING
pc=33>  teqa    attr, type // TEST FAILED: UNSTACKING TEST
pc=5>   jtest   obj, 0, obj, "==" // TEST FAILED
```

Figure 10. Output of the simulator for $(logic(C, state, clear))$.

```
================================
(-)(C,on,A):
================================
pc=0>   fork    27
pc=1>   teqa    attr, on
pc=2>   filter  obj, val, "!= "
pc=3>   merge   dummy, bm1, am0, 4
pc=4>   merge   bm1, bm2, am1, 6
pc=5>   jtest   obj, 0, obj, "==" // L.A. OK: PUSH TEST
pc=6>   fork    20
pc=7>   merge   bm2, bm3, am1, 9
pc=8>   jtest   obj, 1, obj, "!=" // TEST FAILED: FORKING
pc=20>  merge   bm2, bm6, am2, -1
pc=21>  jtest   obj, 0, obj, "==" // L.A. OK: PUSH TEST
pc=22>  term    MoveToTable, 4 // ** (-) MATCH **
pc=26>  subst   (0,2), (2,2), last=true// UNSTACKING TEST
pc=21>  jtest   obj, 0, obj, "==" // TEST FAILED: FORKING
pc=27>  fork    33
pc=28>  teqa    attr, state // TEST FAILED: FORKING
pc=33>  teqa    attr, type // TEST FAILED: UNSTACKING TEST
pc=5>   jtest   obj, 0, obj, "==" // TEST FAILED
```

Figure 11. Output of the simulator for deleting $(logic(C, on, A))$.

9.3. Simulated tests: cognitive module

For the cognitive module simulation the Akka library was used. The reason is that Akka implements the actor model of parallelism, which allows the program to contain routines running in parallel but with isolated execution contexts, communicating only through message passing; this suits the module specification given in Section 8.

http://dx.doi.org/10.5772/intechopen.79265

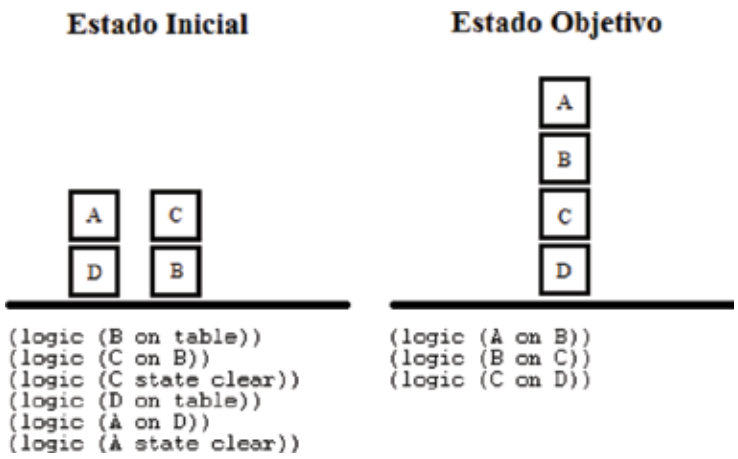


Figure 12. Start and goal states for the planning simulation.

As mentioned early in this chapter, one of the tasks of the cognitive module is to monitor plan execution. But this simulated experiment deals only with the execution of the *Graphplan* algorithm for planning problem solution, since this is the computationally heavier task that inspired the architecture.

The problem domain chosen for this test was, as in the *Rete* module, the block world domain. But this time, for the establishment of a planning problem, a start state and a goal state must be provided. Those are shown in **Figure 12**.

To show the execution of the experiment, the sequence diagram in the **Figure 13** was generated by the simulation[1]. In the image it is possible to see that the expansion sub-module (that represents the processor implementing the graph expansion) communicates with the *Rete* module to generate the state and action layers of the planning graph. This sub-module is also responsible for the calculation of mutex relations in each layer and for detecting the presence of the goal set in the most recent state layer. It can be seen that after three expansion cycles the goal is found and the search sub-module is triggered, starting the recursive search for a solution.

The solution is not found, and this is the point where the main advantage of the architecture became apparent: while the search sub-module tries to find a plan in the graph, the expansion and the *Rete* modules work together to generate a new layer in the graph. And, for this experiment, the new layer also has the objective state. The search module finds a solution this time, namely: $\langle\{moveToTable(A, D),\ moveToTable(C, B)\},\ \{move(C, table, D)\},\ \{move(B, table, C)\},\ \{move(A, table, B)\}\ \rangle$. The planning is then halted.

[1]The simulation generated a sequence of commands corresponding to the task being executed by each sub-module and the messages being exchanged between them, and the commands were transformed in the sequence diagram by the tool WebSequenceDiagrams (https://www.websequencediagrams.com/).

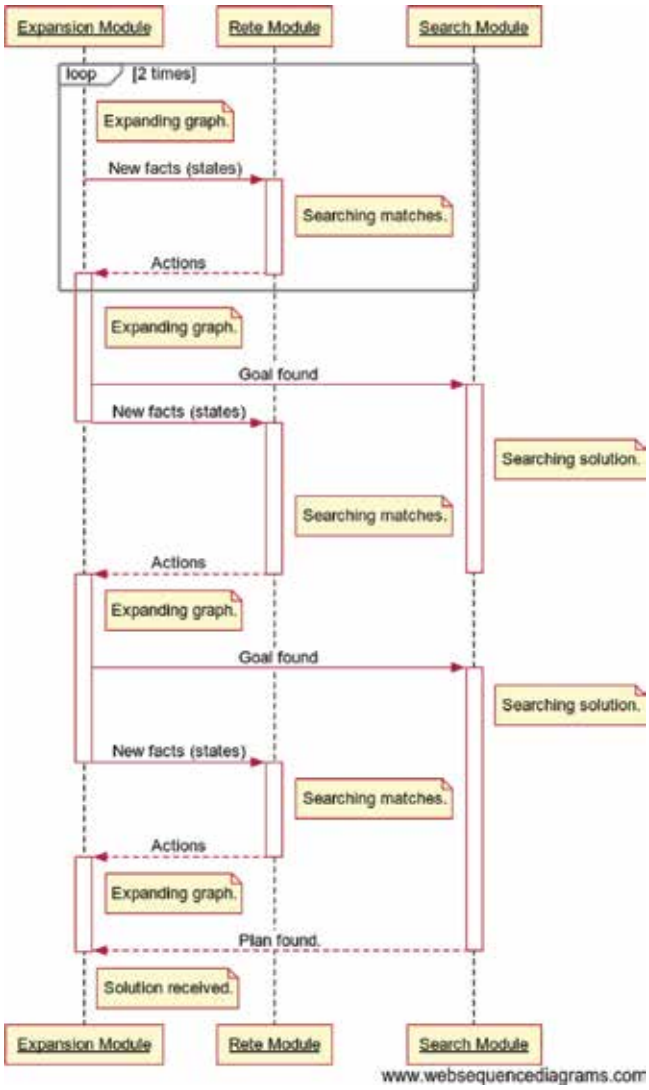


Figure 13. Sequence diagram for the simulation of the cognitive module.

10. Final considerations

This chapter presented the system level design of a SoC for the execution of cognitive agents. It uses the cognitive model of a agent architecture named concurrent autonomous agent (CAA). The SoC architecture was described in terms of the levels of the CAA and the tasks these levels execute. Two main computational modules of the SoC are the RISC *Rete* machine that employs an application specific ISA to detect matches in a KBS (in both the SoC correspondents of the instinctive and cognitive levels of the CAA) and the multiprocessed cognitive level dedicated to (graph) planning.

Simulated experiments shown that the architecture proposed for the aforementioned modules are valid in the sense of performing correctly their tasks. As future works, a formal verification method should be applied in order to validate the all modules separately and working together, as well as their computational and energetic performance, to show that the proposed architecture allows for low-level symbolic processing, which can give embedded and on-chip systems fast automatic reasoning capabilities, with low energy consumption.

Author details

Diego Ferreira*, Augusto Loureiro da Costa and Wagner Luiz Alves De Oliveira

*Address all correspondence to: diego.stefano@gmail.com

Federal University of Bahia, Salvador, BA, Brazil

References

[1] Russel S, Norvig P. Inteligencia Artificial. Rio de Janeiro: Elsevier; 2004

[2] Huhns MN, Singh MP. Cognitive agents. IEEE Internet Computing. 1998;**2**(6):87-89

[3] Vernon D, Metta G, Sandini G. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. IEEE Transactions on Evolutionary Computation. 2007;**11**(2):151

[4] Langley P, Laird JE, Rogers S. Cognitive architectures: Research issues and challenges. Cognitive Systems Research. 2009;**10**(2):141-160

[5] Costa AL, Bittencourt G. From a concurrent architecture to a concurrent autonomous agents architecture. Lecture Notes in Artificial Intelligence. 1999;**1856**:85-90

[6] Bittencourt G, Costa ALd. Hybrid cognitive model. In: Third International Conference on Cognitive Science ICCS'2001 Workshop on Cognitive Agents and Agent Interaction; 2001

[7] Cerqueira RG, Costa ALd, McGill SG, Lee D, Pappas G. From reactive to cognitive agents: Extending reinforcement learning to generate symbolic knowledge bases. In: Simpsio Brasileiro de Automao Inteligente; 2013

[8] Ferreira DSF, Silva RRd, Costa ALd. Embedding a concurrent autonomous agent in a microcontrollers network. In: 5th ISSNIP-IEEE Biosignals and Biorobotics Conference (2014): Biosignals and Robotics for Better and Safer Living (BRC). IEEE; 2014. pp. 1-6

[9] Forgy CL. Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence. 1982;**19**(1):17-37

[10] Lehr TF. The implementation of a production system machine; 1985. Technical Report CMU-CS-85-126

[11] Peters M, Brink C, Sachweh S, Zündorf A. Scaling parallel rule-based reasoning. In: The Semantic Web: Trends and Challenges. Cham: Springer; 2014. pp. 270-285

[12] Panteleyev MG, Puzankov DV, Kolosov GG, Govorukhin IB. Design and implementation of hardware for real-time intelligent agents. In: 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002). IEEE; 2002. pp. 6-11

[13] Frost J, Numan M, Liebelt M, Phillips B. A new computer for cognitive computing. In: 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC). IEEE; 2015. pp. 33-38

[14] Kim Y, Shin D, Lee J, Lee Y, Yoo HJ. A 0.55 V 1.1m Wartificial-intelligence processor with PVT compensation for micro robots. In: 2016 IEEE International Solid-State Circuits Conference (ISSCC). IEEE; 2016. pp. 258-259

[15] Bittencourt G. The quest of the missing link. In: International Joint Conference of Artificial Intelligence; 1997

[16] Ferreira DSF, Paim CC, Costa ALd. Using a real-time operational system to embed a kinematic controller in an omnidirectional mobile robot. In: XI Simpósio Brasileiro de Automação Inteligente (SBAI) e XI Conferência Brasileira de Dinâmica, Controle e Aplicações (DINCON); 2013. pp. 1-6

[17] Forgy CL. On the Efficient Implementation of Production Systems. Dissertation. Department of Computer Science, Carnegie-Mellon University; 1979

Edited by Efren Gorrostieta Hurtado

This book includes a selection of research work in the mobile robotics area, where several interesting topics are presented. In this way we find a review of multi-agents, different techniques applied to the navigation systems, artificial intelligence algorithms, which include deep learning applications, systems where a Kalman filter estimator is extended for visual odometry, and finally the design of an on-chip system for the execution of cognitive agents. Additionally, the development of different ideas in mobile robot applications are included and hopefully will be useful and enriching for readers.

ISSN 2632-5195
ISBN 978-1-83962-086-7

Published in London, UK

IntechOpen