Humanoid Robot Navigation Based on A Multimodal Cognitive Interface

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

eingereicht beim Department Informatik von

Mohammed Elmogy

aus Dakahlia, Ägypten

August 2010

Genehmigt von der
Fakultät für Mathematik, Informatik und Naturwissenschaften
Department Informatik der Universität Hamburg

auf Antrag von
Prof. Dr. Wolfgang Menzel (Vorsitzender)
Prof. Dr. Jianwei Zhang (Erstgutachter/Doktorvater)
Prof. Dr. Christopher Habel (Zweitgutachter/Doktorvater)

Hamburg, den 16.09.2010 (Tag der Disputation)

Abstract

As the capabilities of the mobile robots as well as their abilities to perform more tasks in an autonomous manner are increased, we need to think about the interactions that humans will have with these robots. Human-robot interaction (HRI) has recently received considerable attention in the academic community, government labs, technology companies, and through the media. The interdisciplinary nature of HRI requires researchers in the field to understand their research within a broader context. Since natural language is the easiest and most natural mode of communication for humans, it is desirable to use it to instruct the robot and to generate easy-to-understand messages for the user. Using natural language to teach a navigation task to a robot is an application of a more general instruction-based learning methodology. It can be used to instruct the robot with higher-level goals or to handle certain behaviors and modify their execution. One effective way is to describe the route to the robot in a multimodal way.

On the other hand, significant progress has been made towards stable robotic bipedal walking in the last few years. This is creating an increased research interest in developing autonomous navigation strategies which are tailored specifically to humanoid robots. Efficient approaches to perception and motion planning, which are suited to the unique characteristics of bipedal humanoid robots and their typical operating environments, are receiving special interest. One important area of research involves the design of algorithms to compute robust navigation strategies for humanoid robots in human environments. Therefore, autonomous robot navigation based on route instruction is becoming an increasingly important research topic with regard to both humanoid and other mobile robots.

In this dissertation, the problem of humanoid robot navigation in indoor environments is addressed. A complete framework is presented for humanoid robot navigation based on a multimodal cognitive interface. First, a spatial language to describe route-based navigation tasks for a mobile robot is proposed. This language is implemented to present an intuitive interface that enables novice users to easily and naturally describe a route to a mobile robot in indoor environments. An instruction interpreter is implemented to analyze the user's route to generate its equivalent symbolic and topological map representations which are used as an initial path estimation for the humanoid robot.

Second, a robust lightweight object processing system with a high detection rate is developed. It can actually be used by mobile robots and meet their hard constraints to recognize landmarks during navigation. A landmark processing system is developed to detect, identify, and localize different types of landmarks during robot navigation in indoor or miniature city environments. The system is based on a two-step classification stage which is robust and invariant towards scaling and translations. By combining the strengths of appearance-based and model-based object classification techniques, it provides a good balance between fast processing time and high detection accuracy.

Finally, a time-efficient hybrid motion planning system for a humanoid robot in indoor environments is implemented. The proposed technique is a combination of sampling-based planner and D* Lite search to generate dynamic footstep placements in unknown environments. A modified cylinder model is used to approximate the trajectory for the robot's body-center during navigation. It calculates the actual distances required to execute different actions of the robot and compares them to the distances from the nearest obstacles. D* Lite search is then used to find dynamic and low-cost footstep placements within the resulting configuration space.

Zusammenfassung

Da die Fähigkeiten von mobilen Robotern einschließlich ihrer Möglichkeiten, Aufgaben autonom durchzuführen, erweitert wurden, muss die Interaktion zwischen Mensch und Roboter neu betrachtet werden. Human-Robot-Interaction (HRI) ist ein aktuelles Thema in der Forschung, in Technologie-Unternehmen und in den Medien. Der interdisziplinäre Charakter des HRI-Bereiches erfordert Forschung innerhalb eines breiten Themenkomplexes. Da natürliche Sprache das einfachste und natürlichste Mittel der Kommunikation für Menschen ist, ist es wünschenswert, diese Form der Kommunikation auch bei der HRI zu nutzen, um einem Roboter Anweisungen zu geben und leicht verständliche Botschaften für den Benutzer zu generieren. Die Verwendung natürlicher Sprache zur Instruierung bei Navigations-Aufgaben ist eine Anwendung einer allgemeineren instruktions-basierten Lernmethodologie. Dem Roboter können so übergeordnete Ziele mitgeteilt werden, bestimmte Verhaltensweisen geändert oder auch die Ausführung einzelner Aktionen modifiziert werden. Eine effiziente Methode zur Beschreibung der Route ist die Verwendung multimodaler Anweisungen.

Weil die vergangenen Jahre einen bedeutenden Fortschritt auf dem Gebiet der humanoiden Roboter und des stabilen zweibeinigen Gehens gebracht haben, besteht ein verstärktes Forschungsinteresse an der Entwicklung autonomer Navigationsstrategien, die speziell auf humanoide Roboter zugeschnitten sind. Von besonderem Interesse sind effiziente Ansätze zur kombinierten Perzeptions- und Aktionsplanung, die an die speziellen Eigenschaften von zweibeinigen humanoiden Robotern und ihre typischen Betriebsumgebungen angepasst sind. Ein wichtiges Gebiet der Forschung ist der Entwurf von Algorithmen zur Berechnung von robusten Navigations-Strategien für humanoide Roboter in menschlicher Umgebung. Aus diesem Grunde ist die auf Routen-Instruktion beruhende autonome Roboter-Navigation ein zunehmend interessantes Thema im Hinblick auf humanoide und andere mobile Roboter.

Diese Dissertation befasst sich mit dem Problem der humanoiden Roboter-Navigation in Innenräumen. Es wird ein komplettes Framework für humanoide Roboter-Navigation basierend auf einer multimodalen Schnittstelle vorgestellt. Zunächst wird eine formale Sprache eingeführt, mit der die routen-basierten Navigationsaufgaben beschrieben werden können. Diese Sprache stellt eine intuitive Schnittstelle bereit, mit der auch unerfahrene Anwender leicht einen mobilen Roboter in einer Route in Innenräumen instruieren können. Ein Befehls-Interpreter analysiert die Benutzer-Eingabe und generiert entsprechende symbolische und topologische Darstellungen, die als erste Pfad-Schätzung für den humanoiden Roboter verwendet werden.

Des Weiteren wird in dieser Arbeit ein robustes und effizientes Objekterkennungssystem mit einer hohen Erkennungsrate entwickelt. Es kann bei mobilen Robotern eingesetzt werden und erfüllt die Anforderung, Landmarken während der Navigation zu erkennen. Das Landmarken-Detektions-System ist in der Lage, während der Roboter-Navigation in einer Miniatur-Stadt verschiedene Typen von Landmarken zu detektieren, identifizieren und zu lokalisieren. Das System basiert auf einem zweistufigen Klassifikations-Prozess, der robust und invariant gegenüber Skalierung und Translation ist. Durch die Kombination der Stärken der erscheinungs-basierten und modell-basierten Objekt-Klassifikation bietet es einen guten Kompromiss zwischen schnellen Bearbeitungszeiten und hoher Erkennungsgenauigkeit.

Ebenfalls Bestandteil dieser Arbeit ist die Implementierung eines zeiteffizienten hybriden Bewegungs-Planungs-Systems für humanoide Roboter in einer Innenraum-Umgebung. Die vorgeschlagene Technik ist eine Kombination aus Sampling-basierter Planung und D* Lite-Suche, die ermöglicht, dynamisch Tritt-Platzierungen in unbekannten Umgebungen zu erzeugen. Ein modifiziertes Zylinder-Modell wird verwendet, um die Trajektorie des Roboters während der Navigation näherungsweise zu bestimmen. Die Planungskomponente berechnet die benötigten Freiräume, um verschiedene Aktionen des Roboters auszuführen und vergleicht sie mit der aktuellen Entfernung zu den nächstgelegenen Hindernissen. D* Lite-Suche wird dann verwendet, um eine dynamische und effiziente Platzierung der Schritte innerhalb des resultierenden Konfigurations-Raumes zu finden.

Acknowledgments

First and foremost, I would like to thank ALLAH for supporting me to continue my research and compile this thesis.

Many people contributed to this thesis in one way or another, and I offer my regards and blessings to all of those who supported me in any respect during the completion of this work.

I am heartily thankful to my supervisors, Prof. Dr. Jianwei Zhang and Prof. Dr. Christopher Habel for their inexhaustible wisdom. Their brilliant discussions and comments have made me enjoy working with them. Without their help, it would have been impossible to continue my stay in Germany. Their bright observations always helped me to see the big picture of the problem. Their suggestions and improvements have helped shape this thesis.

I would like to express my deepest appreciation to Dr. Houxiang Zhang and Dr. Carola Eschenbach for their valuable advices and friendly help. Their discussions around my work have been very helpful for this study.

I warmly thank my colleagues from the TAMS & WSV groups for providing professional support and personal advice during my research and writing of this thesis. Specifically, I want to thank Tatjana Tetsis (Lu), Hildegard Westermann, Dr. Andreas Mäder, Manfred Grove, Hannes Bistry, Bernd Schütz, Denis Klimentjew, and Dr. Norman Hendrich.

I wish to thank my colleagues from the CINACS graduate research group: Tian Gan, Patrick McCrae, Cengiz Acartürk, Sascha Jockel, Martin Weser, Christian Graf, Dominik Off, and Kris Lohmann.

I send all my respect and gratitude to the Egyptian Cultural Bureau and Educational Mission for supporting me financially during my scholarship.

Last but not the least, I would like to dedicate this work to my family and all my friends. My parents who always inspired me to work hard and to be positive at all times deserve a special word of gratitude and dedication. I am grateful for my lovely wife, Nabila, and my children, Mariam and Yousuf, for their endless support. Without their encouragement and understanding it would have been impossible for me to finish this work.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER 1 Introduction

Humanoid robots are developed to have an overall appearance and behavior based on that of humans. They have many unique characteristics that distinguish them from the other types of mobile robots, such as a human-like shape, bipedal locomotion, and many degrees of freedom. These features help them to behave like and interact with humans, as well as integrate into human environments without any special considerations.

As the capabilities of the humanoid and mobile robots as well as their abilities to perform more tasks in an autonomous manner are increased, we need to think about the interactions that humans will have with these robots. We also need to design systems that can be used by inexpert users to communicate with the robot. Consequently, human–robot interaction (HRI) has recently received considerable attention in the academic community, government labs, technology companies, and through the media. HRI can be defined as understanding and shaping the interactions between one or more humans and one or more robots. Interactions between humans and robots are inherently present in all of robotics, even for so-called autonomous robots – after all, robots are still used by and are doing work for humans. The interdisciplinary nature of HRI requires researchers in the field to understand their research within a broader context. This field includes many challenging problems and has the potential to produce solutions with positive social impact.

Since natural language is the easiest and most natural mode of communication for humans, it is desirable to use it to instruct the robot and to generate easy-to-understand messages for the user. Using natural language to teach a navigation task to a robot is an application of a more general instruction-based learning methodology. It can be used to instruct the robot with higher-level goals or to handle certain behaviors and modify their execution. Spatial knowledge can be represented in various ways to increase the interaction between humans and mobile robots. Autonomous mobile robots need to use spatial information about the environment in order to effectively plan and execute navigation tasks. The information can be represented at different levels of abstraction. One effective way is to describe the route to the robot in a multimodal way. This method can permit computer language-naive users to instruct their mobile robots to perform complex tasks using simple instructions.

On the other hand, significant progress has been made towards stable robotic bipedal walking in the last few years. This is creating an increased research interest in developing autonomous navigation strategies which are tailored specifically to humanoid robots. Efficient approaches to perception and motion planning, which are suited to the unique characteristics of bipedal humanoid robots and their typical operating environments, are receiving special interest. One important area of research involves the design of algorithms to compute robust navigation strategies for humanoid robots in human environments. Therefore, autonomous robot navigation based on route instruction is becoming an increasingly important research topic for both humanoid and mobile robots.

1.1 Robot Navigation Challenges

The ultimate goal of robot navigation systems is that the robot should be able to easily navigate in dynamic or unknown environments. To accomplish this goal, the robot should have a dialog with the user; should detect, localize, and classify landmarks, avoid obstacles; and generate a feasible path to the goal position. It has been widely recognized that, for such systems, different processes have to work in synergy: high-level cognitive processes for abstract reasoning and planning, low-level sensory-motor processes for data extraction and action execution, and mid-level processes mediating these two levels.

The design and application of robot navigation systems face multiple challenges in the intersection of natural language processing, computer vision, robot control, and motion planning. Some of the key issues targeted in this thesis are summarized in the following subsections.

1.1.1 Cognitive Interface for Robot Navigation

The first challenge is how to give the robot the ability to use human-like spatial language and provide the human user with an intuitive interface that is consistent with his innate spatial cognition.

The second challenge is how the robots overcome ambiguities and misunderstanding in route descriptions to increase the interaction between them and the humans. The robots must not only have the ability to understand perfectly clear and complete instructions, but they must also resolve ambiguities and complement missing information that is inherent in information supplied by humans.

Finally, there is the problem of how to represent a good route description that contains adequate information about the navigation actions which are performed by the robot to reach its destination and the spatial environment in which the intended locomotion of the robot will take place. This information can be represented at different levels of abstraction. One effective way is to describe the route for the robot in a multimodal way. Multimodal route instructions combine natural language route descriptions and visualizations of the route. This method can permit inexpert users to instruct their mobile robots, which understand spatial descriptions, to naturally perform complex tasks using succinct commands.

1.1.2 Object Processing during Navigation

Vision provides the ideal sensor for robots due to its low cost, wide availability, high data content and information rate, and suitability for human environments. For autonomous navigation in unknown or dynamic environments, being able to extract significant information about the world is crucial to operate effectively, making vision an attractive sensor for many robot platforms.

For vision-based robot navigation, there are two basic challenges. The first one is how to choose robust and fast vision processing algorithms to process in real-time scenarios. The choice is based on past experience and intuition to learn which algorithm should be used in execution time. In a dynamic or unknown world, information about the objects in the scene can become obsolete even before it is ready to be used if the recognition algorithm is not fast enough to handle the current robot's view. Consequently, robust and light-weight techniques for object detection, image segmentation, object recognition, and pose estimation are essential for robots working in human environments.

The second challenge is how to detect and classify objects in real scenes during robot navigation. The visual appearance of objects can change enormously due to viewpoint variation, occlusions, and illumination changes. Therefore the vision processing techniques should be capable to deal with these difficulties.

1.1.3 Motion Planning for Humanoid Robots

Motion planning can be defined as the problem that requires the computation of a collision-free feasible path for a robot from a given initial position to a destination position through a workspace populated with obstacles which may be either stationary or moving objects.

For humanoid robots, the motion problem poses two main challenges to developing practical motion planning methods. The first one is how to re-plan a shortest path from the robot's current position to the goal position. While obstacle avoidance remains a key issue, the path planner should consider some important constraints during path generation such as robot dynamics, time-changing workspaces, real-time planning, and dealing with uncertainty in motion and sensors.

The second challenge is how the humanoid robot plans online the sequence of footstep locations to execute the resulting roadmap graph of the path and avoid obstacles while walking in an unknown environment.

1.2 Contribution

The contribution of this work is to present a complete framework for humanoid robot navigation based on a multimodal cognitive interface. The goal is to enable robots to automatically compute their motions from high-level descriptions of tasks and models acquired through sensing. The specific research contributions this thesis makes to the problem of humanoid robot navigation may be summarized as:

- We implemented a complete navigation system for a humanoid robot to execute navigation tasks in indoor environments without any prior knowledge of its environment. The system is used by novice users to describe routes for the robot via a multimodal interface.

- We proposed a spatial language to describe route-based navigation tasks for a mobile robot. This language is implemented to present an intuitive interface that will enable users to easily and naturally describe a route to a mobile robot in indoor and miniature city environments. An instruction interpreter is implemented to analyze the user's route to generate its equivalent symbolic and topological map representations which are used as an initial path estimation for the humanoid robot. A topological map is generated to describe relationships among features of the environment in a more abstract representation without any absolute reference system. It is used to handle uncertainty conditions when the robot cannot recognize the current landmark.

- A robust lightweight object processing system with a high detection rate is implemented. It can actually be used by mobile robots and meet their hard constraints to recognize landmarks during navigation. We developed a landmark processing system which is used by a humanoid robot to detect, identify, and localize different types of landmarks during its navigation in indoor or miniature city environments. The system is based on a two-step classification stage which is robust and invariant towards scaling and translations. By combining the strengths of appearance-based and model-based object classification techniques, it provides a good balance between fast processing time and high detection accuracy. On the other hand, stereo triangulation is calculated to determine the landmark's position in the environment by using the robot's cameras.

- A time-efficient hybrid motion planning system for a humanoid robot in indoor environments is proposed. The proposed technique is a combination of sampling-based planner and D* Lite search to generate dynamic footstep placements in unknown environments. It generates the search space depending on non-uniform sampling of the free configuration space. A modified cylinder model is used to approximate the trajectory for the robot's body-center during navigation. It calculates the actual distances required to execute different actions of the robot and compares them to the distances from the nearest obstacles. D* Lite search is then used to find dynamic and low-cost footstep placements within the resulting configuration space.

1.3 Thesis Outline

The thesis is organized in ten chapters:

In Chapter 2, we present a brief introduction to humanoid robots and their applications. The main features of bipedal humanoid robots, which distinguish them from other types of mobile robots, are discussed. Afterwards, the general issues of HRI and the relationship to HCI are presented. Finally, the basic effective attributes which affect the behavior of the HRI problem are elucidated.

Chapter 3 describes the route-based navigation for the mobile robots. First, the natural language interaction between humans and robots is represented. The spatial reasoning effect on the robot's ability to use human-like spatial language is described. Then, some current implementations of natural language interfaces for both mobile robots and simulated artificial agents are introduced.

The robot navigation systems based on vision sensors are discussed in Chapter 4. The general issues and classes of vision-based robot navigation are introduced. The basic stages of stereo vision which are used to calculate the landmarks' positions and range information from the environment are elucidated. Finally, the object detection and recognition techniques which are suitable for mobile robotics scenarios are presented.

In Chapter 5, a review of previous work carried out on the robot motion planning problem is presented. An overview of the current research efforts in motion planning for mobile and humanoid robots is given. Then, the obstacle avoidance problem and some issues related to real robot motion planning are presented. Finally, the sampling-based motion planning algorithms are elucidated.

The architecture of our Humanoid Robot Navigation System (HRNS) is introduced in Chapter 6. The main modules of HRNS are described briefly. The hardware and software characteristics of the experimental platform are discussed. Finally, the experimental environment of HRNS is presented.

Chapter 7 discusses the route processing module of the HRNS in detail. The multimodal route description interface of the system is presented. The structures of the proposed spatial language and the graphical route representation are introduced. The instruction interpreter and the lexicon structure are illustrated. Then the generation of the topological map of the processed route is described. Finally, the results of some conducted experiments are discussed.

In Chapter 8, our robot landmark processing system (RLPS) is described. RLPS is developed to detect, identify, and localize different types of landmarks during humanoid robot navigation. The architecture of RLPS is discussed in detail. Following this, the results of the vision experiments are presented.

Chapter 9 presents our proposed hybrid motion planning system for a humanoid robot. The components of the proposed technique are described in detail. Then the results of the conducted experiments are discussed.

Chapter 10 concludes with a brief summary of this dissertation. It summarizes the principal achievements of our work, points out the most serious limitations, and suggests a few directions for future work.

CHAPTER 2

Interaction and Communication with Humanoid Robots

An essential aspect distinguishing robotics from other areas of artificial intelligence is their interaction with humans and their surrounding environments. The field of robotics is changing at an unprecedented pace. For example, humanoid robotic hardware and control techniques have been developed rapidly during the last two decades. Lately, several companies have announced the commercial availability of various bipedal humanoid robot prototypes such as Sony QRIO, Fujitsu HOAP, Pal REEM, and Honda ASIMO. As humanoid robots continue to advance and gain new capabilities, software for high-level control and autonomous motion generation will be required to integrate them in human environments without any special considerations.

Therefore, as the capabilities of robots and their ability to perform more tasks in an autonomous manner are increasing, we need to think about the interactions that humans will have with robots. We also need to design systems that can be used by inexpert users to communicate with the robot. Consequently, human–robot interaction (HRI) is a growing field of research and application. Its interdisciplinary nature requires that researchers in the field understand their research within a broader context. This field includes many challenging problems and has the potential to produce solutions with positive social impact.

In this chapter, we first discuss the main characteristics, technologies, and the current humanoid robot prototypes. Next, the technical features which distinguish humanoid robots from other mobile robots are presented. The shape, bipedal walk, and the usage of many degrees of freedom are elucidated. The general issues of HRI and the accepted practices that are emerging in HRI are illustrated. Afterwards, the relationship between human–computer interaction (HCI) and HRI is presented. The unique challenges posed by HRI are discussed. Then, different roles of humans in HRI applications will be described. Finally, the basic ingredients which affect the nature of the HRI problem are elucidated.

2.1 The Need for Humanoid Robots

Robots are increasingly being used in assistive technology, rehabilitation, surgery, and therapy. Other areas, in which the use of robotics is growing, include service and entertainment domains. Methods that enable easy and effective communication between robots and humans are crucial in all of these areas [124]. Humanoid robotics labs worldwide are working on creating robots that are one step closer to science fiction's androids. Building a humanlike robot is a multidisciplinary engineering task requiring a combination of mechanical, electrical, and software engineering; computer architecture; and real-time control. However, over the past 20 years, humanoid robots have become the focus of many research groups, conferences, and special issues. The target of the researchers is to create robots which are capable of interacting with humans in humanlike ways. At the same time, the development of humanoid robots will also lead to learning more about the nature of human intelligence and how to simulate it to create intelligent practical applications.

Humanoid robots can be defined as robots with an overall appearance and behavior based on that of humans [159, 131]. They usually possess a large number of degrees of freedom (DOFs) to imitate some of the same physical and mental tasks that humans carry out daily. They are suitable for coexisting with humans in built-for-human environments because of their anthropomorphism, human-friendly design and applicability of locomotion. The goal is for humanoid robots to be able to both understand human intelligence and reason and act like humans one day. If humanoid robots were able to do so, they could eventually coexist and work alongside humans. They also could act as proxies for humans to do dangerous or dirty work that would not be done by humans if there were a choice, hence provide humans with more safety, freedom, and time. Figure 2.1 shows some current prototypes of humanoid robots which were developed by companies, research institutes, and universities. It also lists some technical details of these prototypes, such as number of degrees of freedom, length, and weight [2, 65, 109, 57, 138, 112, 41].

Therefore, humanoid robots can be considered as a challenging area both in terms of engineering human-like movements and expressions and in terms of the challenges that arise when a robot takes a human form [44]. With such a form, social and emotional aspects of interaction become paramount. Humanoid robots have been widely investigated, particularly in the field of dynamic walking control and mechanical design [111, 23]. However, the current status of these humanoid robots is to walk along a pre-programmed path. To realize humanoid robots in real and unknown environments, a

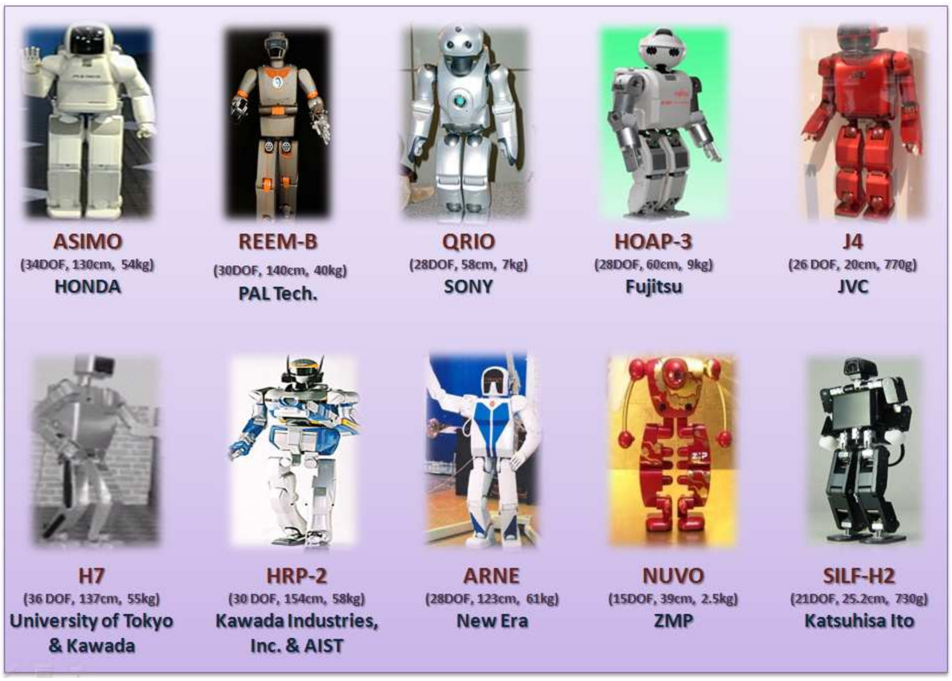


Figure 2.1: Some current humanoid robot prototypes.

sensor-based navigation system is required. A navigation system generates a path that a robot moves along, from a current position to a target position based on spatial information around a robot.

Rapid development of humanoid robots brings about new shifts of the boundaries of robotics as a scientific and technological discipline. On the one hand, new technologies of components, sensors, microcomputers, as well as new materials have recently put up the barriers to real-time integrated control of some very complex dynamic systems, like the fact that humanoid robots already possess about thirty DOFs and are updated in microseconds [136, 156]. On the other hand, significant progress has been made in the design and control of humanoid robots, particularly in the realization of dynamic walking in several full-body humanoid robots [77]. As the technology and algorithms for real-time 3D vision and tactile sensing improve, humanoid robots will be able to perform tasks that involve complex interactions with the environment (e.g. navigation and grasping/manipulating objects). The enabling software for such tasks includes motion planning for obstacle avoidance and integrating planning with visual and tactile sensing data.

To improve the performance of the current humanoid robots, the results of the other research worldwide could be integrated to the humanoid body to provide a powerful platform to develop. There are many examples of several technologies available which can be incorporated in the humanoid robots, such as [142]:

Language technologies: They include voice recognition, speech to text, and voice synthesis. They are sufficiently developed to allow simple interaction with the robot in spoken language.

Face and gesture recognitions: They sufficiently developed to allow robots to read "moods" of the instructor, follow cues, etc. [108].

Knowledge base, dialog and logical reasoning: They prove useful artificial intelligent techniques for the humanoid robots.

Sensing: They include improved vision, hearing, olfaction, tactile sensing, etc. They are developed to a certain extent and incorporated in various commercial devices (artificial retinas, e-nose, and e-tongue) [124].

Consequently, incorporating available technologies and developing new ones on the same integrated platform is an efficient way to bridge the gap between the current state of the art and the future humanoid robots. There is a tight connection between achieving human-friendly means for cognitive-motor skill transfer and interaction through dialog in natural language; similarly, cognition and self-awareness are also related to the development of perceptual maps and schemes; embodying and experimenting with the world are dependent on perceiving the world with multiple sensors, etc.

2.2 Humanoid Robot Features

Recent studies show additional advantages of humanoid robots over other types of robots. First, human acceptance and interaction with robots are easier if the robots have a human shape [64, 130, 5]. Second, the efficiency of teaching and programming a robot is highest with humanoid robots [142, 17]. In particular related to the last aspects, one should stress here that mobility, flexibility, and adaptation to human environments offered by the human shape are a convenient advantage. Then, the key reason for preferring humanoid robots is their optimal shape for being taught by humans and learning from humans. This can be considered as the only way to develop cognitive and perceptual-motor skills for truly intelligent, cognitive robots. A wealth of knowledge ready to be transmitted to the humanoid robots is waiting to be used: while in the first stage robots may learn directly from humans, in the future they could learn by watching humans on training videos and movies.

Consequently, humanoid robots have many features that distinguish them from other types of mobile robots. These features and characteristics let them navigate in complex environments and handle difficult situations. In the ensuing subsections, the main features of humanoid robots are discussed.

2.2.1 Humanoid Robot Shape

One of the most exciting aspects of humanoid robotics research is the potential for a wide range of applications, such as maintenance tasks of industrial plants, security services of home and office, human care, teleoperations of construction machines, and cooperative works in the open air. With their shape and movement inspired by that of humans, they are poised to successfully operate in a human environment, and possibly to perform any task that a human can perform [63]. The humanoid robot shape also produces feelings useful for friendly communication between the robot and the human [81]. Additionally, the humanoid robot shape can be considered as one of the best shapes for controlling robots remotely [145].

With human-like physical form, humanoid robots are potential tools to function in the real world, which is designed for humans. They can coexist and collaborate with humans, and perform tasks that humans cannot. These robots have the ability to move forward and backward, turn in any direction, sideways to the right or the left, and move diagonally [55]. In addition, legged humanoid robots have the unique ability to step onto and over obstacles or unsafe footholds, which can allow them to traverse terrains that would be impassable to a wheeled mobile robot [46, 24]. Their feet can be placed with a greater choice and the change of their body posture permits them to overcome different types of obstacles where the wheeled robots fail. Therefore, these features enable humanoid robots to be integrated in human environments without any special considerations and they can do some actions which cannot be done by any other type of robots, such as step onto objects, climb up and down stairs, step over obstacles, or crawl underneath objects.

Furthermore, due to the unique posture stability control of humanoid robots, they are able to maintain their balance despite unexpected complications such as uneven ground. As a part of their integrated functions, they are able to move on a planned path autonomously and to perform simple operations via wireless teleoperation. Consequently, humanoid robots can work in human environments without any special requirements and they can handle many types of obstacles, such as entrances, stairs, doors, and furniture [55].

Consequently, humanoid robots can be proxies for humans to do dangerous or dirty work that would not be done by people if there were a choice, hence providing humans with more safety, freedom, and time [131, 159]. Projects utilizing teleoperated humanoid robots have begun in searching for systems which can do the same work currently done by humans in critical environments, for example executing space missions, operations of power

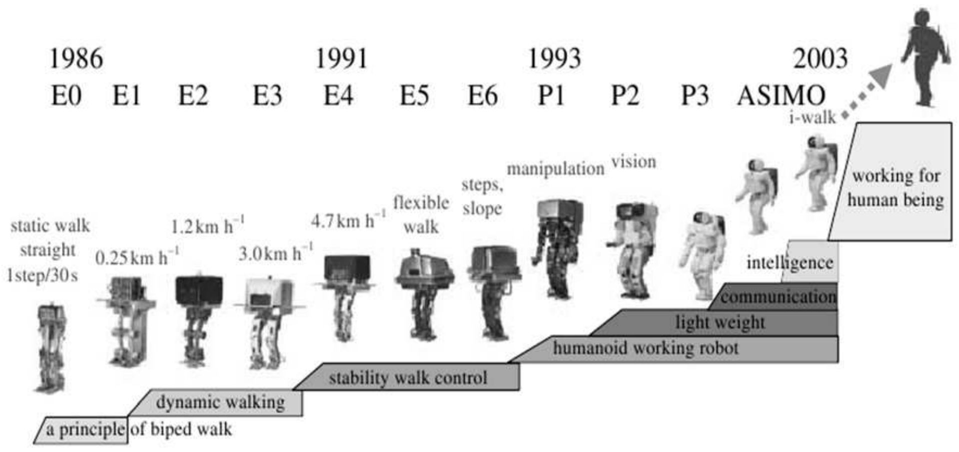


Figure 2.2: Evolution of Honda bipedal humanoid robots [56].

generation plants, tasks at construction sites, and disaster relief missions.

2.2.2 Biped Locomotion

Biped locomotion has become a key topic in robotic research over the last two decades. Both hardware improvements and new software architectures have allowed the implementation of impressive humanoid robots, such as ASIMO, QRIO, REEM, and HOAP. These prototypes have attracted many researchers for their performance of human-like behaviors (see Figure 2.1).

Theoretical studies from various aspects have been accompanied by a lot of simulation work and various practically realized systems, from the simplest cases of planar mechanisms to the Honda and Sony humanoid robots, the most advanced biped locomotion robots designed to date. Figure 2.2 shows the evolution of the biped walking of Honda humanoid robots [56, 57]. The research started with straight and static walking of the first two-legged prototype until the development of ASIMO, which is considered as the most advanced robot of Honda in the mechanism and the control system.

Irrespective of the structure of humanoid robots and the number of DOFs involved, all biped locomotion systems have some basic characteristics which can be summarized in the following three points [122]:

1. The possibility of rotation of the overall system about one of the foot edges caused by strong disturbances, which is equivalent to the appearance of an unpowered (passive) DOF.

2. Gait repeatability (symmetry), which is related to regular gait only.

3. Regular interchangeability of single-support (SSP) and double-support phases (DSP).

During humanoid robot walking, two different situations arise in sequence: the statically stable DSP in which the mechanism is supported on both feet simultaneously and statically unstable SSP when only one foot of the mechanism is in contact with the ground while the other is being transferred from the back to front positions. Thus, the locomotion mechanism changes its structure during a single walking cycle from an open to a closed kinematic chain. All these circumstances have to be taken into account in artificial gait synthesis.

From another perspective, most of the researchers in biped locomotion use one of the two main approaches to execute robot motion: static or dynamic [122, 136]. Static walkers rely on the static equilibrium condition: maintain the Center of Gravity (CoG) on the convex hull of the contact area with the ground. This approach denies inertial forces and therefore can be applied only if robot movements are very slow. Dynamic walkers achieve a fast and natural walking motion following the principle of dynamic equilibrium: they use Zero Moment Point (ZMP) [155, 156, 63] instead of CoG, so that inertia components and gravity are considered. The main objective, in any case, is to produce a gait as natural and stable as possible.

In ZMP, all of the biped mechanism joints are powered and directly controllable except for the contact of the foot with the ground, which is the only point at which the mechanism interacts with the environment. This contact is essential for the walk realization, because the mechanism's position with respect to the environment depends on the relative position of the foot with respect to the ground. The foot cannot be controlled directly, but in an indirect way – by ensuring appropriate dynamics of the mechanism above the foot. Thus, the overall indicator of the mechanism's behavior is the point where the influence of all the forces acting on the mechanism can be replaced by one single force. This point was termed ZMP. Recognition of the significance and role of ZMP in biped artificial walking was a turning point in gait planning and control. Figure 2.3 illustrates the position of ZMP in straightforward and turning movements in humanoid robots.

As mentioned above, all of the bipedal mechanism joints are powered and directly controllable except for the contact between the foot and the ground (which can be considered as an additional passive DOF), where only the interaction of the mechanism and the ground takes place. This contact is essential for the walk realization because the mechanism's position with respect to the environment depends on the relative position of the foot/feet with respect to the ground.

In addition to the ease of design and speed of ZMP calculation, this method of describing a walking trajectory has another advantage. One can easily change the points that are interpolated to define the trajectory. This means that the walking pattern is highly adjustable, because parameters such as stride length (how far the robot steps in one cycle), sway distance (how far the robot shifts its weight to one side when the other leg swings

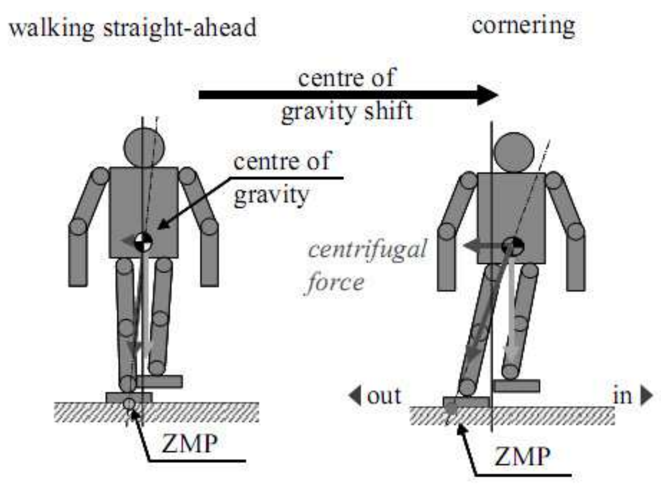


Figure 2.3: Continuous transition for the humanoid robot from walking in a straight line to making a turn [56].

forward), step clearance (how high the robot lifts its feet when it steps), as well as the home position of its feet (the location of the feet beneath the robot when it stands still, to which all other positions are relative) can all be changed easily [63].

2.2.3 Degrees of Freedom

The number of degrees of freedom (DOFs) in modern humanoid robots is large and ever-increasing. The aim of researchers is to produce smooth movements for humanoid robots such as in humans by providing robots with more DOFs. The number of DOFs of the current humanoid robots ranges from about 20 to about 40, while a human has more than 200 DOFs. Figure 2.1 lists the number of DOFs of some current humanoid robot prototypes. In addition to these prototypes, there have been seen some attempts to create a humanoid robot with many DOFs, such as Kotaro (91 DOFs) [104].

On the other hand, methods that attempt to search for or automatically construct complex movements in the full joint space often run afoul of the well-known curse of the dimensionality principle. As the number of DOFs increases, finding and representing an optimal control policy becomes difficult. Approaches, which work well in low-dimensional spaces such as discretization, succumb to unmanageable (exponential) space and processing time requirements [20]. In order to achieve a stable walk for humanoid robots with many DOFs, the previous methods need very accurate parameters and models of robot dynamics and its environment to execute smooth and flexible movements.

2.3 Interaction with Mobile Robots

Human–Robot Interaction (HRI) has recently received considerable attention in the academic community, government labs, technology companies, and through the media. It can be defined as understanding and shaping the interactions between one or more humans and one or more robots. Interactions between humans and robots are inherently present in all of robotics, even for so-called autonomous robots – after all, robots are still used by and are doing work for humans. As a result, evaluating the capabilities of humans and robots and designing the technologies and training that produce desirable interactions are essential components of HRI. Such work is inherently interdisciplinary in nature, requiring contributions from cognitive science, linguistics, and psychology; from engineering, mathematics, and computer science; and from human factors engineering and design.

Both the widely acknowledged shifts from industrial to service/mobile robotics and the resulting increase of robots that operate in close proximity to people raise a number of research and design challenges. The most important characteristic of these new target domains is that service/mobile robots share physical spaces with people. In some applications, people will be professionals that may be trained to operate robots. In others, they may be children, elderly, or handicapped people whose ability to adapt to robotic technology may be limited. The design of the interface, while dependent on the specific target application, will require substantial consideration of the end user of the robotic device. Herein lies one of the great challenges that the field of robotics faces today [149, 164].

Interaction between a human and a robot may take several forms, but these forms are largely influenced by whether the human and the robot are in close proximity to each other or not. Thus, the interaction can be separated into two general categories: remote and proximate [149]. In remote interaction, the human and the robot are not co-located and are separated spatially or even temporally. The operator controls the robot, whereas the robot gives feedback to the operator and provides him with information about its environment and its task. Remote interaction with robots is often referred to as teleoperation or supervisory control, and remote interaction with a physical manipulator is often referred to as telemanipulation. In proximate interaction, the humans and the robots are co-located (for example, service robots may be in the same room as humans). Therefore, proximate interaction with mobile robots may take the form of a robot assistant, and it may include a physical interaction. Social interaction includes social, emotive, and cognitive aspects of interaction. In social interaction, the information flow is bi-directional. The information is communicated between the robot and the user in both directions, and they are interacting as peers or companions.

On the other hand, there are a number of accepted practices that are emerging in HRI [44]. A key practice is to include experts from multiple disciplines on research efforts. These disciplines frequently include robotics, electrical and mechanical engineering, computer science, Human–Computer Interaction (HCI), cognitive science, and human factors engineering. Other relevant disciplines include design, organizational behavior, and the social sciences.

A second emerging practice is to create real systems (robot autonomy, interaction modes, and etc.) and then evaluate these systems using experiments with human subjects. Proof-of-concept technologies, although important, are less valuable than they would be if they were supported by careful experiments that identify key attributes of the design or principles that span applications. Identification of descriptive interaction phenomena is interesting, but elaboration on the psychological principles underlying these phenomena with an eye toward harnessing these principles in design is more useful. Thus, engineering, evaluation, and modeling are key aspects of HRI.

A third emerging practice is conducting experiments that include a careful blending of results from simulated and physical robots. On the one hand, because of the cost and reliability issues, it is often difficult to conduct carefully controlled experiments with physical robots. On the other hand, it is often difficult to replicate simulation-only results with physical robots because the physical world presents challenges and details that are not present in many simulations. It is common to “embody” at least one portion of the interaction, be it a physical robot, some physical sensor, or real-world speech. Some research includes work using carefully controlled simulation environments and replication of selected results with physical robots.

2.4 From Human–Computer Interaction to Human–Robot Interaction

As the field of HRI has grown, it has seen many contributions from researchers in HCI and it has been nurtured by HCI organizations. HRI research is attractive to many members of the HCI community because of the unique challenges posed by the field. Of particular interest is the fact that robots occupy physical space. This offers unique challenges not offered in desktop metaphors or even pervasive computing. Physical location in a 3D space imposes strong requirements on how information is displayed in remote operation, and even stronger requirements on how space is shared when robots and humans occupy the same space. HRI benefits from contributions presented by HCI researchers in the form of methodologies, design principles, and computing metaphors [44].

Therefore, HRI is fundamentally different from typical HCI in several dimensions. HRI concerns systems which have complex, dynamic control systems, exhibit autonomy and cognition, and which operate in changing, real-world environments. In addition, differences occur in the types of interactions (interaction roles); the physical nature of robots; the number of systems a user may be called to interact with simultaneously; and the environment in which the interactions occur. Accordingly, the differences between HRI and HCI can be summarized in the following points [126, 44, 149, 130]:

Human roles: In HRI, humans can interact with robot in different ways. Each of these interactions has different tasks and hence different situational awareness needs. The interactions of the human with the robot can be classified into seven different roles: supervisor, operator, mechanic, peer, bystander, pupil, and information consumer (for more details, see section 2.6).

The physical nature of mobile robots: Robots need some awareness of the physical world in which they move. As robots can physically move from one location to another, they present more interesting challenges. As robots move about in the real world, they build up a "world model". This model needs to be conveyed to the human in order to understand decisions made by the robot as the model may not correspond exactly to reality due to the limitations of the robot's sensors and processing algorithms.

The dynamic nature of the robot platform: Typical HCIs assume that computer behavior is for the most part deterministic and that the physical state of the computer does not change in such a way that the human must track it. However, robotic platforms have physical sensors that may fail or degrade. While some functionality may be affected, the platform may be able to carry out some limited tasks.

The environment : Platforms to monitor robots may have to function in harsh conditions such as dust, noisy, and low-light conditions. Environments may be dynamic as well. For example, search and rescue robots may encounter more building or tunnel collapses during the operation.

The number of independent systems: Typical HCI assumes one user interacting with one system. In the case of humans and robots, the ultimate HRI goal is to have a person interacting with a number of heterogeneous robots.

The ability to perform autonomously: While typical desktop computers perform autonomously in that they execute code based on user commands, robots use planning software to alleviate the user from dealing with low-level commands and decisions. Thus a robot can go from point A to point B without asking the operator how to deal with each obstacle encountered along the path.

2.5 Evolution of HRI Systems

In the last two decades, the HRI field has developed rapidly. It was first associated with teleoperation of factory robotic platforms. Telerobotics are defined as “direct and continuous human control of the teleoperator” or a “machine that extends persons sensing and/or manipulating capability to a location remote from that person” [149]. In this period, telerobotics was considered as a part of HCI.

HRI goes beyond teleoperation of a remote platform and allows for some set of autonomous behaviors to be carried out by the robot. This could range from a robot responding to extremely precise commands from a human about adjustment of a control arm to a more sophisticated robot system planning and executing a path from a start point to an end point supplied by a user. The concept of HRI has only become possible in the last two decades because of advances in the field of robotics (perception, reasoning, and programming) that make semi-autonomous systems feasible [126].

Kidd [68] noted that human skills are always required in robotic systems. He maintained that designers should use robot technology to support and enhance skills of the human as opposed to substituting skills of the robots for skills of the human. He argued for developing and using robotic technology such that human skills and abilities become more productive and effective, such as freeing humans from routine or dangerous tasks. He pointed out that robotic researchers tend to focus on issues that are governed by legislative requirements such as safety. Human-centered design issues have been mostly ignored. Kidd suggested that human-centered design of HRI needs to look beyond technology issues and to consider issues such as task allocations between people and robots, safety, and group structure. These issues need to be considered in the early stages of the technology designs. If they are only considered in the final stages, the issues become secondary and have little impact on design considerations.

Fong et al. [40] argued that if robots and humans work together as partners, the system will gain many benefits from this cooperation. But partners must engage in dialogue, ask questions of each other, and jointly solve problems. They proposed a system for collaborative control which provides the best aspects of supervisory control without requiring user intervention within a critical window of time. In collaborative control, the human gives advice but the robot can decide how to use human advice. This is not to say that the robot has the final authority but rather than the robot follows a higher-level strategy set by the human with some freedom in execution. If the user is able to provide relevant advice, the robot can act on that. However, if the user is not available within the time needed, the robot will use default behaviors to react to the situation. Collaborative control is only possible if the robot is self-aware, has self-reliance and can maintain its own safety, has a dialoge capacity, and is adaptive. Dialoge management and user models are needed to implement collaborative control systems.

At the Idaho National Laboratory (INL), Nielson et al. [107] focused primarily on the use of robots as tools that can effectively help first responders, soldiers, and other individuals involved in emergency response and other critical, hazardous endeavors. To develop the technologies correctly required numerous user studies of differing types and situations including simulation, the physical world, different environments, different tasks, and participants with varying levels of prior knowledge regarding the task and use of robots. All the different experiments were required because each led to new insights about how people view and use robots as team members.

Robotics researchers use the term human–robot intervention, often in place of human-robot interaction. For robotic systems that have plan-based capabilities, the term intervention is used when a human needs to modify a plan that has some deficiency or when the robot is currently unable to execute some aspect of a plan. While robots carrying out preplanned behaviors is certainly desired (e.g., clean the kitchen floor, watch the perimeter, check all the rooms on the floor for an object), more closely coupled human–robot teams need to interact spontaneously as well [126].

Kollar et al. [75] are taking steps towards building a robust natural language system by focusing on understanding a subset of robot tasks. They presented an HRI system that follows natural language directions by extracting a sequence of spatial description clauses from the linguistic input and then infers the most probable path through the environment given only information about the environmental geometry and detected visible objects. They used a probabilistic graphical model that factors into three key components. The first component grounds landmark phrases such as "the computers" in the perceptual frame of the robot by exploiting co-occurrence statistics from a database of tagged images. Second, a spatial reasoning component judges how well spatial relations such as "past the computers" describe a path. Finally, verb phrases such as "turn right" are modeled according to the amount of change in orientation in the path.

2.6 Human Roles in HRI Systems

The goal of HRI is to create teams of both humans and robots that are efficient and effective to take advantage of the skills of each member in the team. An important sub-goal is to increase the number of robotic platforms that can be handled by individuals. In order to accomplish this aim, the types of interactions that will be needed between humans and robots should be examined first. Then, the desirable information that will be interchanged between humans and robots should be defined. Finally, the software and interaction architectures are developed to accommodate these needs.

With respect to the interaction between humans and robots, a taxonomy of roles that robots can assume from humans can be divided into seven different classes as follows [126, 44, 60]:

Supervisor: A supervisor role could be characterized as monitoring and controlling the overall situation. This could mean that a number of robots would be monitored and the supervisor would be evaluating the given situation with respect to a goal that needs to be carried out. For robots that possess planning systems, the goals and intentions have been given to the planning system, and the robot software is generating the actions based on a perception of the real world. The supervisor can step in and specify an action or modify plans. In either case, a formal representation of the goal and intention is necessary so that the supervisor can formulate the effect an intervention will have on the longer-term plan. The main loop in this type of interaction is the perception/evaluation loop as most of the actions are automatically generated by the robot software. Supervisor interactions at the action and intention level must be supported as well (see Figure 2.4).

Operator: The operator is called upon to modify internal software or models when the robot behavior is not acceptable. The operator deals mainly with interacting at an action level which contains the allowed actions to the operator.

Mechanic: The mechanic deals with physical interventions, but it is still necessary for the mechanic to determine if the interaction has the desired effect on the behavior. So, the model looks similar to the model for the operator interaction. However, the difference is that while the modifications have been made to the hardware, the behavior testing needs to be initiated in software and observations of both software and hardware behavior need to be made to ensure that the behavior is now correct.

Peer: Teammates of the robots can give them commands within the larger goal/intentions. The supervisor has only the authority to change the larger goal/intentions. Therefore, the interaction needs to occur at a higher level of behavior than the operator interactions allow [40].

Bystander: The bystander might be able to cause the robot to stop by walking in front of it, for example. In this model, the bystander has only a subset (sub A) of the actions available. He is not able to interact at the goal or intention level. Feedback must be directly observable. The largest challenge here is how to advise the bystander of the capabilities of the robot that are under his control.

Pupil: The robot is in a teaching or leadership role for the human. The pupil has only a subset (sub A) of the actions available, such as the

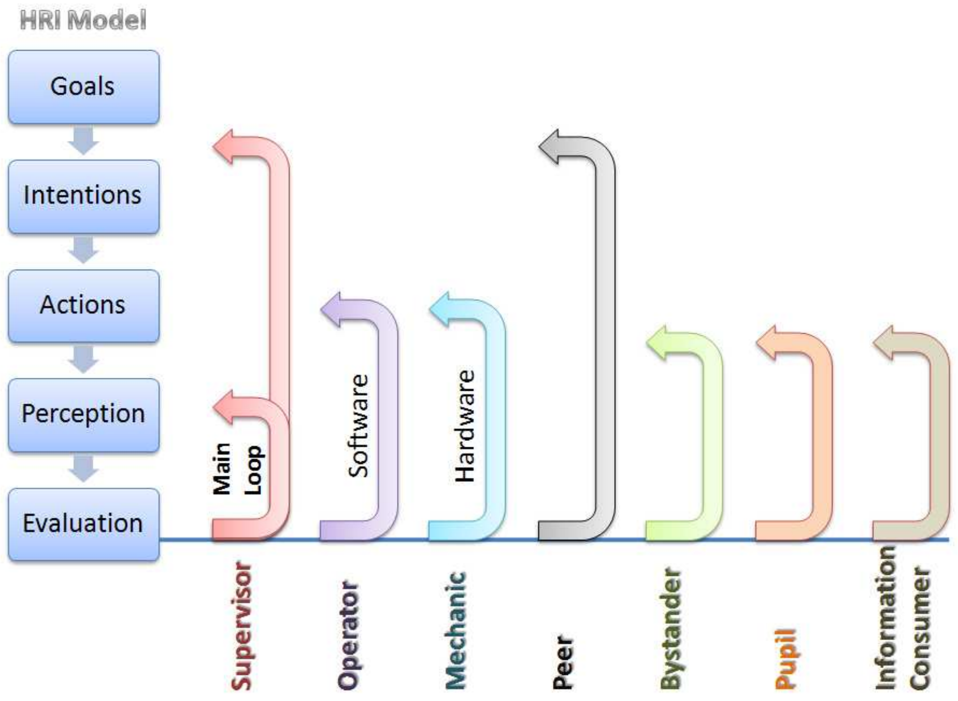


Figure 2.4: Different models of human–robot interaction.

bystander. He is not able to interact at the goal or intention level [60].

Information Consumer: The human does not control the robot, but the human uses information coming from the robot in, for example, a reconnaissance task or a game [157].

Similar taxonomies are certainly possible, but identifying how people perceive a robot's role has important ramifications for how they interact with the robot. The last classification provides insight into the current and future interactions in these applications. Table 2.1 classifies the most frequent types of interactions in different application areas of robotics.

2.7 Effective Attributes of HRI

From the design perspective, it is helpful to adopt the designer's perspective by breaking the HRI problem down into its constituent parts. In essence, a designer can affect five attributes that have influence on the interactions between humans and robots [44]: (1) the level and behavior of robot autonomy; (2) the nature of information exchange between the user and the robot; (3) the adaptation, learning, and training of people and the robot (common grounding); (4) the structure of the team; and (5) the shape of the task.

Application Area	**Type**	**Role**	**Example**
Search and rescue	Remote	Human is supervisor or operator	Remotely operated search robots
	Proximate	Human and robot are peers	Robot supports unstable structures
Assistive robotics	Proximate	Human and robot are peers, or robot is tool	Assistance for blind, and therapy for the elderly
	Proximate	Robot is mentor	Social interaction for autistic children
Military and police	Remote	Human is supervisor	Reconnaissance de-mining
	Remote or Proximate	Human and robot are peers	Patrol support
	Remote	Human is information consumer	Commander using reconnaissance information
Edutainment	Proximate	Robot is mentor	Robotic classroom assistant
	Proximate	Robot is mentor	Robotic museum tour guide
	Proximate	Robot is peer	Social companion
Space	Remote	Human is supervisor or operator	Remote science and exploration
	Proximate	Human and robot are peers	Robotic astronaut assistant
Home and industry	Proximate	Human and robot are peers	Robotic companion
	Proximate	Human is supervisor	Robotic vacuum
	Remote	Human is supervisor	Robot construction

Table 2.1: Examples of roles and proximity patterns that arise in several areas of HRI applications [44].

HRI emerges from the confluence of these factors to achieve the system's goal. The designer attempts to understand and shape the interaction itself, with the objective of making the exchange between humans and robots beneficial in some sense. In the following subsections, these attributes will be discussed in detail.

2.7.1 Levels of Autonomy

Autonomy refers to a robot's ability to accommodate variations in its environment. Hence, a system with a high level of autonomy is one that can be neglected for a long period of time without interaction. Autonomy is not an end in itself in the field of HRI, but rather a means to support productive interaction. Indeed, autonomy is only useful insofar as it supports beneficial interaction between a human and a robot. Consequently, the physical embodiment and type of autonomy varies dramatically across robot platforms.

Perhaps the most strongly human-centered application of the concept of autonomy is in the notion of levels of autonomy (LOA). Levels of autonomy describe to what degree the robot can act on its own accord. Different robots exhibit different degrees of autonomy. The degree of autonomy is often measured by relating the degree at which the environment can be varied to the mean time between failures, and other factors indicative of robot performance. HRI cannot be studied without consideration of a robot's degree of autonomy, since it is a determining factor with regards to the tasks a robot can perform, and the level at which the interaction takes place.

The kinds of robotics are characterized by different levels of autonomy, largely pertaining to the complexity of environments in which they operate [149]. It should come as no surprise that industrial robots operate at the lowest level of autonomy. In industrial settings, the environment is usually highly engineered to enable robots to perform their tasks in an almost mechanical way. As a result, careful environment engineering indeed minimizes the amount of autonomy required – a key ingredient of the commercial success of industrial robotics.

While environment modifications are still commonplace, the complexity of service/mobile robot environments mandates higher LOA than in industrial robotics. In general, robots which operate in close proximity to people require a high degree of autonomy, partially because of safety concerns and partially because people are less predictable than most objects. It is common practice to endow service/mobile robots with sensors capable of detecting and tracking people. The type and degree of autonomy in service robotics varies more with the specific tasks a robot is asked to perform and the environment in which it operates.

Some researchers have gone as far as devising techniques whereby robots learn about people's routine behavior and actively step out of the way when people approach. Personal robots tend to be aimed at low-cost markets. As a result, endowing a personal robot with autonomy can be significantly more difficult than its more expensive professional relative. These robots are used to build autonomous systems which provide robustness and flexibility that task-specific systems can never achieve.

While such average scales are appropriate to describe how autonomous a robot is, from an HRI point of view, a complementary way to consider autonomy is by describing to what level humans and robots interact and the degree to which each is capable of autonomy. The scale presented in Figure 2.5 puts an emphasis on mixed-initiative interaction, which has been defined as a "flexible interaction strategy in which each an agent (human or robot) contributes what it is best suited at the most appropriate time".

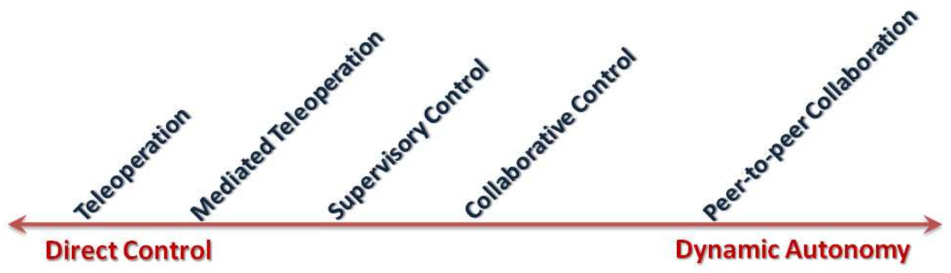


Figure 2.5: Levels of autonomy with emphasis on human interaction [44].

In order to achieve peer-to-peer collaboration, the robot must indeed be able to flexibly exhibit "full autonomy" at appropriate times. Moreover, it may need to support social interactions. As a result, peer-to-peer collaboration may be considered more difficult to achieve than full autonomy.

2.7.2 Information Exchange

Autonomy is only one of the components required to make an interaction beneficial. A second component is the manner in which information is exchanged between the human and the robot. Measures of the efficiency of an interaction include the interaction time required for intent and/or instructions to be communicated to the robot, the cognitive or mental workload of an interaction, the amount of situation awareness produced by the interaction (or reduced because of interruptions from the robot), and the amount of shared understanding or common ground between humans and robots.

There are two primary dimensions that determine the way information is exchanged between a human and a robot: the communications medium and the format of the communications. The primary media are delineated by three of the five senses: seeing, hearing, and touch. These media are manifested in HRI as follows [44]:

- Visual displays: They are typically presented as graphical user interfaces or augmented reality interfaces [64].

- Gestures: They include hand and facial movements and movement-based signaling of intent.
- Speech and natural language: It includes both auditory speech and text-based responses, which frequently emphasizes dialog and mixed-initiative interaction.
- Non-speech audio: It is frequently used in alerting.
- Physical interaction and haptics: They are frequently used remotely in augmented reality or in teleoperation to invoke a sense of presence especially in telemanipulation tasks, and also frequently used proximately to promote emotional, social, and assistive exchanges.

Recently, attention has focused on building multimodal interfaces [113, 120, 124] for robots to provide intuitive and pleasant interactions with non-expert humans. These interfaces are partly motivated by a quest to reduce workload and also a desire to make interactions more natural and easier to learn.

Regarding the robots that interact with humans, the robots' appearance and initial behavior must create in humans an appropriate mental model of the robots' abilities and intentions. The robots must be able to clear up inevitable misunderstandings, and they must adapt to the needs of different users [69].

2.7.3 Common Grounding

Two persons cannot even begin to coordinate on content without assuming a vast amount of shared information or common ground – that is, mutual knowledge, mutual beliefs, and mutual assumptions. To coordinate on process, they need to update their common ground moment by moment. All collective actions are built on common ground and its accumulation [27, 121]. In communication, common grounding cannot be properly updated without a process which is called grounding (least collaborative effort). The grounding process has been described within a framework that views communication as a form of collaborative action [10].

Effective communication between people and interactive robots will benefit if they have a common ground of understanding. The common ground principle of least collective effort can be used to predict and design HRI applications. Social cues lead people to create a mental model of a robot and estimates of its knowledge. People's mental model and knowledge estimate will, in turn, influence the effort they make to communicate with the robot. People will explain their message in less detail to a knowledgeable robot with which they have more common ground. This process can be leveraged to design interactions that have an appropriate style of robot direction and that accommodate differences among people.

The theory of common ground was developed to understand communication between people [26, 121]. Its main assumption is that communication between people requires coordination to reach mutual understanding, just as ballroom dancers and basketball teams do. The process of coordination relies on a large amount of shared knowledge between the parties, that is, common ground. One of the key postulates of the theory of common ground is least collaborative effort, that is, people in conversation minimize their collective effort to gain understanding. Achieving least collective effort should be an ultimate goal of successful HRI. The solution for achieving this most basic form of common ground is to create in people's minds an appropriate mental model of the robot automatically.

Kiesler [69] discussed research demonstrating the tendency for people to attribute knowledge to a strange robot based on their beliefs about the robot's origin, the tendency for people to communicate with a strange robot differently based on the robot's physical characteristics, and the ways that a robot could speak to a person depending on the expertise level of the person. She called on these lines of evidence to support the argument that the common ground principle is an important factor in HRI. The results showed that when the agent looked like a person, people cooperated with the person-like agent at the same level as they did with the real person. The results of her experiment also suggested that mental models are not just general beliefs (e.g., this robot is nice, funny, or respectful). Mental models also comprise a set of task-specific expectations of process, that is, how the system will work. She also argued that mental models are situation-specific, that is, that expectations of process can change depending on the situation. The more the robot can adapt to the person, the less the person needs to adapt to the robot.

2.7.4 The Structure of the Team

HRI problems are not restricted to a single human and a single robot, though this is certainly one important type of interaction. Robots used in search and rescue, for example, are typically managed by two or more people, each with special roles in the team. Similarly, managing Unmanned/Uninhabited Air Vehicles (UAVs) is typically performed by at least two people: a "pilot" who is responsible for navigation and control, and a "sensor/payload operator" who is responsible for managing cameras, sensors, and other payloads.

A question that has received considerable attention, but which is directly addressed by few scientific studies, is how many remote robots a single human can manage [44]. In general, the answer is dependent on factors such as the level of autonomy of the robot (e.g., teleoperation requires a great deal of direct attention from the human), the task (which defines not only the type and quantity of data being returned to the human but also the amount of attention and cognitive load required of the human), and the available modes of communication.

2.7.5 The Shape of The Task

Robotic technology is introduced to a domain either to allow a human to do a task that he could not do before, or to make the task easier or more pleasant for the human. Implicit in this assertion is the fact that introducing technology fundamentally changes the way that humans do a task. Task-shaping is a term that emphasizes the importance of considering how the task should be done and will be done when new technology is introduced. Compared to the other ways that a designer can shape HRI, there is little written about task-shaping.

There are formal processes for understanding how the task should be done and is currently done. These processes include goal-directed task analyses, cognitive work analyses, and ethnographic studies. Although frequently used to specify how a task is done and how it should be done, it is imperative to consider how the task will be done, including unintended consequences of design.

2.8 Summary

An essential aspect distinguishing robotics from other areas of artificial intelligence is their interaction with humans and their surrounding environments. The field of robotics is changing at an unprecedented pace. In the past, most robots operated in industrial settings, where they performed tasks such as assembly and transportation. Equipped with minimal sensing and computing, robots were created to perform the same repetitive task over and over again. Currently, researchers try to let robots provide services directly in human environments without applying any special settings.

In this chapter, we presented a brief introduction about humanoid robots and their applications. The main features of bipedal humanoid robots, which distinguish them from other types of mobile robots, were discussed. We also introduced the general issues of HRI and the accepted practices that are emerging in HRI. Then, the relationships and differences between HCI and HRI were presented. The different models of HRI and human roles in these models were illustrated. Finally, the basic effective attributes which affect the behavior of the HRI problem were elucidated.

CHAPTER 3 Route–Based Navigation for Mobile Robots

Since natural language is the easiest and most natural mode of communication for humans, it is desirable to use it to instruct the mobile robot and to generate easy-to-understand messages for the user. Using natural language to teach a task to a robot is an application of a more general instruction-based learning methodology. It can be used to instruct the robot with higher-level goals or to handle certain behaviors and modify their execution.

Spatial knowledge can be represented in various ways to increase the natural language interaction between humans and mobile robots. Autonomous mobile robots need to use spatial information about the environment in order to effectively plan and execute navigation tasks. The information can be represented at different levels of abstraction. One effective way is to describe the route verbally to the robot. This method can permit inexpert users to instruct their mobile robots, which understand spatial descriptions, to naturally perform complex tasks using succinct commands.

In this chapter, we mainly discuss the route-based navigation for the mobile robots. Firstly, the natural language interaction between humans and robots is represented. The spatial reasoning effect on the robot's ability to use human-like spatial language is described. Then, some current implementations of natural language interfaces for both mobile robots and simulated artificial agents are introduced. Afterwards, the mobile robot navigation and its broad categories are elucidated. Finally, the route description, symbolic, and environmental representations for the mobile robots are discussed.

3.1 Natural Language Interface for Artificial Agents

Natural language interaction is considered as a challenging problem, not only because it requires sophisticated speech recognition and language understanding, but also because it inevitably includes issues of mixed-initiative interaction, multimodal interaction, and cognitive modeling [44]. Natural language can express rules and sequences of commands in a very concise way. It uses symbols and syntactic rules to interact with robots that have knowledge represented at the symbolic level. Such symbolic communication can help robots to learn faster when they learn at the sensory–motor association level [87].

To accept mobile robots as cooperative partners, they must not only have the ability to understand perfectly clear and complete commands, but they must also resolve ambiguities and complement missing information that is inherent in information supplied by humans. Therefore, to build effective interactions, humans and robots should have a common ground of understanding [44, 5]. This common ground creates realistic expectations and forms the basis communications. From a robot's perspective, supporting effective interactions also requires establishing and maintaining common ground. An emerging approach to doing this is to create cognitive models of human reasoning and behavior selection. The goal is to create rich enough models either to allow the robot to identify a human's cognitive state and adjust information exchange accordingly, or to allow the robot's behavior to be generated by models that are interpretable by a human.

On the other hand, spatial reasoning on the natural language route is essential for both humans and mobile robots. Spatial reasoning gives robots the ability to use human-like spatial language and provides the human user with an intuitive interface that is consistent with his innate spatial cognition. It can also accelerate learning by using symbolic communication. A robot capable of understanding spatial language could be controlled by a novice user naturally to perform complex tasks using succinct, intuitive commands. Moreover, there is evidence that spatial reasoning underlies many parts of human cognition; a system that performs spatial reasoning may allow the user to bootstrap human understanding of many other facets of intelligence. By trying to build spatially competent machines, a useful system in itself is created and it provides insight into possible mechanisms for modeling human cognition [147, 135].

In the last three decades, there has been considerable research on spatial language and spatial reasoning. This motivates the research interest of using spatial language for interacting with artificial agents. Many researchers [93, 153, 147, 135, 100] have proposed frameworks using natural language commands in simulated or real-world environments to guide their artificial agents during navigation. For example, Tschander et al. [153] proposed the idea of a cognitive-oriented Geometric Agent (GA) which simulates instructed navigation in a virtual planar environment. This geometric agent can navigate on routes in its virtual planner environment according to natural language instructions presented in advance. The GA is implemented to study the interaction between the spatial information given in route instructions and the spatial information gained from perception. In their approach, Conceptual Route Instruction Language (CRIL) is used to represent the meaning of natural language route instructions. CRIL-expressions are constructed from a basic inventory of descriptive operators. On the one hand, CRIL-expressions specify the semantics of natural language expressions in the traditional method of formal semantics. On the other hand, CRIL is an internal language of GA that relates to perceptual objects and specifies actions carried out in GA. CRIL and formal reasoning based on CRIL-expressions can be used to test contrasting proposals for the semantics of spatial expressions regarding their consequences for the performance of an instructed navigator.

MacMahon [98] introduced the MARCO system for understanding and executing natural language route instructions in 3D large-scale virtual indoor environments. MACRO is an agent that follows free-form, natural language route instructions by representing and executing a sequence of compound action specifications that model which actions to take under which conditions. MARCO infers implicit actions from knowledge of both linguistic conditional phrases and from spatial action and local configurations. It performs explicit actions, implicit actions necessary to achieve the stated conditions, and exploratory actions to learn about the world.

Tellex and Roy [147] implemented spatial routines to control the robot in a simulator. They defined a lexicon of words in terms of spatial routines and used that lexicon to build a speech-controlled robot in a simulator. Their system is unified by a high-level module that receives the output from the speech recognition system and simulated sensor data, creates a script using the lexicon and the parse structure of the command, and then sends appropriate commands to the simulated robot to execute that command. However, their current implementation acts only on the current snapshot of sensor readings which leads to errors in the robot's behavior.

Levit and Roy [91] have developed components of an automated system that understands and follows navigational instructions. The system has prior knowledge of the geometry and landmarks of specific maps. This knowledge is exploited to infer complex paths through maps based on natural language descriptions. Their approach is based on an analysis of verbal commands in terms of elementary semantic units that are composed to generate a probability distribution over possible spatial paths in a map. An integration mechanism based on dynamic programming guides this language-to-path translation process, insuring that resulting paths satisfy continuity and smoothness criteria. In their implementation, parsing of text into semantic units is performed manually. Composition and interpretation of semantic units into spatial paths is performed automatically.

On the other hand, considerable research efforts are being made to develop various command sets for mobile robots and robotic wheelchairs [127, 148, 114, 133]. The mobile robot community has created systems that can understand natural language commands. Many research efforts [153, 135, 5, 150, 75] focus on using spatial language to control the robot's position and behavior, or to enable it to answer questions about what it senses. Torrance [150] implemented a system that is capable of mediating between an unmodified reactive mobile robot architecture and domain-restricted natural language. He introduced reactive-odometric plans (ROPs) and demonstrates their use in plan recognition. The communication component of this architecture supports a typewritten natural language discourse with people. This system was brittle due to place recognition from odometric data and the use of IR sensors for reactive motion control. The resulting ROPs do not contain error-reducing stopping conditions, and this had caused problems in some parts of the tested environment where hallways did not sufficiently constrain the reactive navigation system.

In [87, 82, 15], Instruction-Based Learning (IBL) is built to train mobile robots using natural language instruction to describe a navigation task. In this project, a robot is instructed on how to travel from one place to another in a miniature town. IBL uses unconstrained language in a real-world robotic application that learns prior to execution. In IBL, the user's verbal instructions are converted into new internal program code that represents new procedures. Such procedures become part of a procedure pool that robots reuse to learn increasingly complex procedures. Hence, the robot should be capable of executing increasingly complex tasks.

Skubic et al. [135] implemented robot spatial relationships combined with a multimodal robot interface that provides the context for the human-robot dialog. They showed how linguistic spatial descriptions and other spatial information can be extracted from an evidence grid map and how this information can be used in a natural human-robot dialog. With this spatial information and linguistic descriptions, they established a dialog of spatial language. To overcome the object recognition problem (the system does not support vision-based object recognition), they had defined a class of persistent objects that were recognized and named by the user.

Pradel and Hoppenot [116] proposed a method for symbolic trajectory description in unknown indoor environments. The chosen form uses a panoramic description called fresco. The method uses distance measurements from a 2D laser range finder, digitizes the robot's visibility area, eliminates superfluous data and reorients their presentation. The landmarks are then extracted and organized into the fresco which is validated by means of neighborhood rules. As the robot moves in the environment, the frescoes are created and both the amount of new information a fresco carries out and its position in relation to the preceding ones are evaluated. Only frescoes selected as enough informative are stored to describe the robot's route.

Kollar et al. [75] presented a system that follows natural language directions by extracting a sequence of spatial description clauses from the linguistic input and then infers the most probable path through the environment given only information about the environmental geometry and detected visible objects. They used a probabilistic graphical model that factors into three key components. The first component grounds landmark phrases in the perceptual frame of the robot by exploiting co-occurrence statistics from a database of tagged images. Second, a spatial reasoning component judges how well spatial relations describe a path. Finally, verb phrases are modeled according to the amount of change in orientation in the path.

3.2 Navigation Problem at a Glance

Spatial competence is a central aspect of human intelligence, and a crucial component of any system that needs to intelligently move itself or objects in the world. Spatial language is a window to spatial cognition: humans use spatial language to describe spatial situations, to refer to objects, and to request other humans to take actions. Humans possess the remarkable ability to give and follow natural language route instructions through large-scale spaces.

In navigation, a director describes the actions and observations along the route by recalling the environment's topology, metrical layout, and visual features. A follower interprets these descriptions, navigating by applying the instructions to the possibly unfamiliar environment. Furthermore, followers must account for mistakes, ambiguities, and omissions in the route description [148, 99]. In other words, despite the directors' best efforts, not all instructions are perfectly clear and reliable for reaching the goal. Often instructions contain ambiguous information, qualitative mistakes within the instruction, or metrical mistakes. Because of these failings, the follower must treat the instructions as guidance, not as strict commands.

Therefore, navigation is defined as the process that involves both planning and execution of agent's movements. It consists of two main components: locomotion and way-finding [105]. Locomotion is the movement of one's body around an environment, coordinated specifically to the local or proximal surrounds. Accordingly, it is guided both perceptually by current sensory information and cognitively by previously acquired information. There are various modes of locomotion which can be classified into two categories. Unaided by machines, people of different ages (or different states of mind or body) can roll, crawl, climb, slither, walk, hop, jog, or run. Aided by machines, there is the usual litany of planes, trains, and automobiles (and then some). Modes of locomotion are important because they determine much about the humans' way of acquiring and processing information as they locomote.

In contrast to locomotion, way-finding is the goal-directed and planned movement of one's body around an environment in an efficient way. It is planning and decision making coordinated to the distal as well as local surrounds. In general, way-finding requires controlled, explicit strategies and working-memory processes when people are in unfamiliar places, including when they are lost. Way-finding is composed of three activities: knowledge storage and access (i.e., the cognitive map), decision making for planning actions, and decision execution to turn decisions into behaviors.

The great majority of acts of navigation involve locomotion and way-finding components to varying degrees. Perhaps the most interesting and important cases where the mapping of declarative/non-declarative onto way-finding/locomotion is problematic concern instances where spatial inferences are made in relatively immediate surrounds.

Consequently, successful navigation means that agents reach their goal in an efficient and accident-free manner. To do so, it is necessary that as agents move, they maintain a sense of where they are relative to their goal, where places and objects they should avoid are located, and so on [105]. A variety of sensory and motor systems provides information for updating during locomotion. Humans recognize landmarks primarily visually, because vision is the most precise channel for spatial and pattern information, particularly at a distance, but landmark recognition may be based on audition, olfaction, radar or satellite signals, and so on.

3.3 Mobile Robot Navigation

In mobile robotics, navigation has always been an interdisciplinary topic of research, because mobile agents of different types are inevitably faced with similar navigational problems. Therefore, human navigation can readily be compared to navigation in other biological organisms or in artificial mobile agents like autonomous robots. The set of navigational strategies found in mobile robots mirrors the complexity of navigational strategies employed by biological organisms. In many instances, the close resemblance between biological and artificial navigation is not a mere coincidence. Artificial navigation often mimics evolutionary proven strategies in an attempt to build robust technologies. In some instances, artificial agents are even specifically designed to test biological models of navigation [158].

There are, however, a number of important differences between biological and artificial agents. First, modern technology makes a wide range of very accurate sensors possible, such as laser range finders, radar and ultrasonic sensors, global positioning systems, etc., providing information about the environment or the position of an agent. Unlike biological organisms, who might possess a few highly developed sensory systems for particular sources of information (e.g., the innate compass for the desert ant); the designer of a robot system is free to use an arbitrary combination of different sensors. In addition, robots sometimes are equipped from the start with a precise representation of their spatial environment (e.g., a map).

In many cases, the environment is also specifically designed to match the robot's navigational or sensory abilities, e.g. by placing markers at important points of a route or using well-demarcated paths along which the robot travels. The basic tasks of spatial navigation, however, remain the same for both robots and biological organisms. According to [151, 158], four different, broad categories to classify natural and artificial navigation can be distinguished:

Guidance: It is mainly concerned with directly leading an agent by external cues – either by following a particular gradient or moving to match the current sensory image with a stored image of the target or of the surroundings. In all these cases, the agent tries to locally maximize a predefined criterion without knowledge of spatial relations in the environment or about its own position. A robot could, for example, try to follow a wall of a corridor by keeping a constant distance to it.

Place Recognition–Triggered Response: For place recognition based strategies, complex spatial behaviors are triggered at distinct points in space. Once the correct place is recognized, the associated action (e.g., movement in a particular direction or guided behavior) will lead to complex trajectories. The main problem of this strategy obviously consists in the correct identification of a place.

Topological Navigation: It describes navigation based on topological networks and is thus a more flexible extension of place-triggered navigation. The basic elements of this type of network are places and the connections between these places. In this case, the agent possesses knowledge how to get from one place to a second place which is connected to it. The agent's spatial knowledge is confined, however, to the topological network. New places not included in the network, or new connections between two places cannot be found by navigating on the basis of a topological network.

Metrical Navigation: Unlike the last two approaches, which divide space in a small number of distinct places and the space in between, metrical navigation does not require such a distinction in principle. The metric most frequently used is Euclidean, thus distances and angles are well defined and can be used to drive spatial navigation. Pre-existing maps, which specify the metrical relations between objects in the environment of the agent, are often supplied directly or are autonomously constructed by triangulation and integration of sensory information. A coarse version of metrical navigation can be seen in world representations such as occupancy grids. Unlike navigation based on topological networks, the agent can determine its position in space and its spatial relation to each other object within the same metrical space for each position. However, knowledge of its position does not, as in topological navigation, automatically trigger the correct action. Instead, the action usually has to be computed for the current situation.

Table 3.1 summarizes the previous four levels of navigation strategies according to the following three criteria: (i) the information structure and content; (ii) the movement selection procedure; and (iii) the behavioral repertoire of the navigation strategy.

For mobile robots, their capability to autonomously go from one point to another is considered as an important issue for their navigation in the surrounding environment. Autonomous navigation is based on three main concepts: planning, navigation, and environment representation [116]. Firstly, planning computes a trajectory between the start and target points. Secondly, navigation gives motion orders to the robot to follow the computed trajectory. It is based on three principal kinds of techniques: map-based navigation using predefined geometric and/or topological models, map-building-based navigation where the robot construct geometric and/or topological models on their own, and mapless navigation using only object recognition and actions associated to these objects. Finally, environment representation permits the robot to know if it goes in the right direction or not.

In general, robot navigation can be processed in four basic steps. First, the robot should perceive its environment by using its sensors, such as a laser range finder, IR, and stereo cameras. The second step is to build the digital representation of the environment. The third step is to extract the landmarks from the environment. Finally, the robot specifies paths and locations of landmarks [83].

Bipedal humanoid robots, as discussed in chapter 2, have unique characteristics distinguishing them from other types of mobile and service robots. They can be integrated in the human environment without applying any special settings. They have the ability to recognize and avoid different types of obstacles in their surrounding environment. To let the bipedal humanoid robot navigate autonomously in human environments, it needs to solve the last mentioned tasks and handle objects (landmarks) with respect to their characteristics [159].

3.4 Route Instructions for Robot Navigation

Every day, people use route instructions to travel along previously unknown routes. Route instructions can be defined as an instruction set intended to guide a mobile agent toward a spatial destination. They are also referred to as “route descriptions”, “route directions”, “route guidance”, or simply “directions”. Route instructions are a special case of verbal instructions, which include recipes, assembly instructions, and usage manuals. They are

	Name	**Stored spatial information**	**Procedure**	**Characteristics**
0	**Target Approaching**	None	Taxis	Basic requirement for navigation.
1	**Guidance**	Identity of the landmark configuration. Raw state of the sensory inputs at goal location.	Minimize the mismatch between the perceived configuration and the memorized configuration (approach).	Local navigation. Only when direct perception is available.
2	**Place Recognition–Triggered Response**	Landmark configurations defining places. A local directional reference frame for each. The direction of movement that leads to the goal from each place.	Self-localize by recognizing the current places as an already experienced place. Orient relative to it. Move in the goal-associated direction.	Way-finding. Stimulus-response type of behavior.
3	**Topological Navigation**	A set of landmark configurations linked by topological relationships.	Search for the sequence of places linked by experienced routes from the current place to the goal.	Way-finding. Stimulus-response-stimulus type of behavior. Topological detours (path section).
4	**Metric Navigation**	A set of landmark configurations linked by metric relationships.	Plan a trajectory which will be followed by lower level strategies. The resulting path is not necessary a previously taken one.	Way-finding. Metric detours, metric shortcuts, novelty.

Table 3.1: The hierarchy of navigation strategies [151].

limited enough to be tractable and applicable to useful real-world tasks with clear criteria for evaluation.

Route instructions are considered as one of the more important natural language interfaces between humans and mobile robots for applying an effective HRI. They are an interesting combination of robotics, artificial intelligence, cognitive psychology, and natural language processing [98, 100]. Route instructions are easily evaluated, despite the complexity of integrating modules doing linguistic modeling, abstract spatial reasoning, and moving a robot through a world. They frequently involve two kinds of expressions that connect to spatial structure: a verb of motion (such as go, turn, enter, and throw) and a directional adverb or a directional prepositional phrase (such as into the zoo, through the park, back, and straight on) [36].

Therefore, route instructions specify spatial information about the environment of the route and temporal information about the actions (movements, turns) to be performed [153]. Human route instructions are usually conveyed either verbally in spoken discourse or written texts, or by graphical means, i.e. by illustrating the route on a map, or drawing sketch-maps. Whereas verbal route instructions focus on the actions to be performed and take the spatial environment as the frame for these actions, maps and other pictorial representations foreground the spatial environment without possessing adequate means for representing the actions. The major deficit of maps as means for route instruction is that they do not focus on the sequence of actions to be performed but on the spatial environment. A third possibility is to combine these two kinds of external representations leading to multimodal route instructions. Multimodal route instructions combine natural language route descriptions and visualizations of the route to follow, such that the strengths of both means for communication route knowledge are brought together[49, 162, 161].

Nevertheless, all types of route instructions have to provide correlated actions, paths, tracks, positions and landmarks to describe the navigation path to the agent. All of these route instruction components can be classified and categorized into main groups to facilitate the analysis of the navigation task [153].

Verbal route instructions represent knowledge about spatial actions and spatial layouts. A route instruction set is useful if it reliably guides followers to the intended destination. Verbal route instructions are explanations given by a director intended to guide a mobile agent, the follower, toward a specific spatial destination. When following route instructions, the follower must parse and interpret the text, model the instruction's actions and descriptions, and enact the instructions in the world, by performing these actions and recognizing the descriptions. Typically, a follower cannot simply execute instructions without inference, since the necessary actions are not completely specified. Instructions often provide just a skeletal plan of action. A follower can resolve the ambiguities and omissions by using knowledge of language, an understanding of spatial actions and relations, and a model of the environment. The core measure of a set of instructions for a route is simple – did the follower end up at the intended destination?

A property characteristic for in advance route instructions is that neither the instructor nor the follower perceives the relevant environment, the critical landmarks, or the tracks, e.g., the roads, completely and directly. During the instruction phase an instructor, who possesses knowledge about the environment in question, produces a route instruction. In comprehending the instruction, the follower builds up conceptual, mental representations of the route. These representations, which contain spatial information about the route and the sequence of actions to be performed, have to be stored in memory. Later, in the navigation phase, the instructed navigator has to match the internal representations against the perceived scenes. This process involves the recognition of spatial configurations of landmarks, tracks, and positions in accordance with the spatial relations specified in the instruction.

The overall criterion for the adequacy of a route instruction is whether it enables navigator agents to find their way. Thus, adequacy depends on a wide spectrum of parameters. For example, epistemological parameters, such as the knowledge of the participants (the instructor and the follower), or perceptual parameters, which concern the navigator's perception of the environment and the perceptual salience of landmarks, can influence the performance of the navigator. Since not all objects or spatial configurations that the navigator will perceive on the route can be specified in the instruction, the type and amount of information, e.g. concerning landmarks, provided by the instructor is crucial for successful route instructions.

Good route instructions should contain adequate information about the two aspects. First, the aspect concerns navigation actions, in particular locomotion actions and perception actions, which are performed by the robot to reach its destination. The second is the spatial environment in which the intended locomotion of the robot will take place. The instructor's primary task is to choose a good combination of communicational means to transfer the relevant information concerning both aspects to the robot [49].

MacMahon [98] proposed four basic actions to be used in following verbal route instructions: turning in place, moving from one place to another, verifying a view description against an observation, and terminating a current action. The primary characteristic of a path is the change of location. Turns can be viewed as changes in orientation. These considerations led to four basic types of navigational information: moves, turns, positions and orientations. Altogether, moves and turns can be subsumed under the general notion of actions, and positions and orientations can be viewed as verifications.

Another important issue which should be considered in describing route is the reference systems. Reference systems can be classified into relative, intrinsic, and absolute [105, 127, 91]. Relative systems are essentially egocentric. The objects in the route description are described from the speaker's point of view or some other viewer. Intrinsic systems code direction relative to the asymmetric shape of a feature in the environment, i.e. from the object's point of view. Absolute systems code directions relative to global features that function over large areas. These frames of reference can be used to construct or describe spatial relationships in the environment. The use of different frames of reference in different languages indicates that language may restructure the spatial representations of the language speaker, rather than the existence of innate and universal spatial concepts.

3.5 Perception–Based Symbolic Representations

Extracting basic instruction elements from sentences containing navigational information and grounding them in action primitives is a common strategy for understanding systems (for more details, see Appendix B). One approach to solving the problem of vision-based robot navigation is to model both of the 3D environment and the verbal route instructions as symbolic data and to process all data input on this symbolic level. This approach will allow the robot to perceive and interpret its environment if the symbolic representation of both the environment (i.e., landmarks) and the route description can be mapped to their corresponding real world and verbal description, respectively. This abstract representation will reduce the problems of machine vision and artificial intelligence into a subset of the general symbolic data [146].

Building the symbolic description of the route followed by a robot is a threefold problem. The first problem is how to build the qualitative descriptions in accordance with the robot's sensors. The second one is how to describe the route by a sequence of the most pertinent symbols. Finally, there is the question of how to use these symbols with the control-command level of the robot [116]. Tedder and Hall [146] described the symbolic processing methods in detail. The general method for perception and interpretation is proposed to symbolically represent and manipulate data in a mapping process. They solved the problem in modeling the 3D environment as symbolic data and in processing all data input on this symbolic level. The results of obstacle detection and avoidance experiments demonstrate that the robot can successfully navigate the obstacle course using symbolic processing control.

3.6 Environment Map-like Representations for Mobile Robots

Building a representation of the environment is an important task for a mobile robot that aims to move autonomously in the surrounding space. Over the past decades, the problem of building maps and navigating indoor environments has received significant attention in the mobile robotics community. The problem of building maps is the problem of determining the location of certain entities, such as landmarks or obstacles, in a global frame of reference. To build a map of the environment, a robot must know where it is. Since robot motion is inaccurate, constructing maps of large indoor environments requires a robot to solve an inherent concurrent localization problem.

In robotics, the common descriptions of the space are metric and topological maps. A metric map represents the environment according to the absolute geometric position of obstacles. A topological map is a more abstract representation that describes relationships among features of the environment without any absolute reference system. Topological maps are usually represented in graph form [160, 80, 105].

On the one hand, metric maps generate fine-grained, metric descriptions of a robot's environment. They include distance, direction, and shape, organized in a global allocentric reference system. These are modeled best by statistical estimation theory, such as Bayesian modeling [105]. In these representations, the robot's environment is defined by a single global coordinate system, in which all mapping and navigation takes place. Typically, the map is a grid with each cell of the grid representing some amount of space in the real world. These grids became quite sophisticated at representing the spatial structure of the world. These approaches typically work well in bounded environments, with little consistent structure and where the robot has opportunities to realign itself with the global coordinate system using external markers.

On the other hand, topological maps generate coarse, graph-like descriptions of environments, where nodes correspond to significant, easy-to-distinguish places or landmarks, and arcs correspond to actions or action sequences that connect neighboring places. Topological maps are qualitative descriptions of the robot's workspace, in which the environment is represented as places and connections between places. Indeed, the idea of a map that contains no metric or geometric information, but only the notions of proximity and order, is very attractive because such an approach eliminates the inevitable problems of dealing with movement uncertainty in mobile robots. Movement errors do not accumulate globally in topological maps as they do in maps with global coordinate systems since the robot only navigates locally, between places. Topological maps can also be more compact in their representation of space, in that they represent only interesting places and not the entire environment. Topological maps have become increasingly popular in mobile robotics. Being more abstract, a topological map has the advantage of being more compact and more stable with respect to sensor noise and to small changes in the environment. Unfortunately, the semantics associated to topological maps are still somehow ambiguous [80].

In principle, topological maps could be scaled to the size of large-scale indoor environments better than metric maps could, because a coarse-grained, graph-structured representation is much more compact than a dense array, and more directly suited to problem solving algorithms. However, purely topological maps have difficulty in distinguishing adequately between different places, and have not been applied to large environments in practice. Recent progress in metric mapping has made it possible to build useful and accurate metric maps of reasonably large-scale environments, but memory and time complexity pose serious problems [79, 160].

3.7 Summary

Natural language is considered as the easiest and most natural mode of communication for humans. It is desirable to use it to instruct the robot and to generate easy-to-understand messages for the user. Using natural language to teach a route for a robot is an application of a more general instruction-based learning methodology.

In this chapter, we discussed the main features of the natural language interfaces for artificial agents and the general issues of mobile robot navigation. The spatial reasoning effect on the robot's ability to use human-like spatial language is described. Then, some current implementations of natural language interfaces for both mobile robots and simulated artificial agents are introduced. Afterwards, the mobile robot navigation and its broad categories are elucidated. The importance of generating symbolic representations for both route descriptions and 3D environments is discussed. Finally, the representations of the robot's environments are elucidated.

CHAPTER 4

Vision–Based Robot Navigation

For many robot domains, vision provides the ideal sensor for robots due to its low cost, wide availability, high data content and information rate, and suitability for human environments. For autonomous navigation in unknown or dynamic environments, being able to extract significant information about the world is crucial to operate effectively, making vision an attractive sensor for many robot platforms. For vision-based autonomous robots, it is important that vision processing algorithms are fast in addition to being robust. Selecting which algorithms should be used by a mobile robot is a decision that is usually made a priori by the system developer. The choice is based on past experience and intuition to learn which algorithm should be used in execution time. Consequently, robust techniques for object detection, image segmentation, object recognition, and pose estimation are essential for robots that work in human environments.

This chapter discusses the mobile robot navigation based on vision sensors. In the next section, the general issues and classes of vision-based robot navigation are introduced. Then, the basic stages of robot stereo vision, which are used to calculate the landmarks' positions and range information from the environment, are elucidated. Finally, the object detection and recognition techniques which are suitable to mobile robotics scenarios are presented.

4.1 Autonomous Navigation Based on Vision Sensors

Vision is one of the most powerful and popular sensing methods used for autonomous robot navigation that continues to demand a lot of attention from the mobile robot research community. When vision sensors are compared with other on-board sensing techniques, the vision sensor has the ability to provide detailed information about the environment which may not be available using combinations of other types of sensors. The past decade has seen the rapid development of vision-based sensing for indoor mobile robot navigation tasks.

For mobile robot navigation, autonomous robots operating in unknown and uncertain environments have to cope with dynamic changes in these environments. The major challenge of the autonomous robot navigation is to navigate successfully to its goal while avoiding both static and dynamic obstacles in its surroundings. In other words, when the robot is in an area it does not know, it becomes necessary for it to identify and learn a useful set of landmarks in its surroundings so that it can return through the area, knowing exactly where to go and what is around it. Throughout this learning process the robot must identify potential landmarks based on sensory information, learn where those landmarks are, and then be able to predict, with reasonable accuracy, where to expect to sense the next landmark. Yet most robotic applications tend to be one of the two [134]: Either entirely blind to everything in their surroundings except what is required merely for navigating through the environment, or else designed to function in a fixed setting, where they have a well-known and unchanging frame of reference that they can relate objects to.

On the other hand, vision-based robot navigation can be classified into three main groups with respect to the familiarity of the robot with its environment [47]. The first class is the map-based navigation which consists of providing the robot with a model of the environment. These models may contain different degrees of detail, varying from a complete CAD model of the environment to a simple graph of interconnections or interrelationships between the elements in the environment. The second group is the map building based navigation. In this approach, a 2D or 3D model of the environment is first constructed by the robot using its on-board sensors after which the model is used for navigation in the environment. Finally, the third group is the mapless navigation. This category contains all systems in which navigation is achieved without any prior description of the environment. The required robot motions are determined by observing and extracting relevant information about the elements in the environment, such as walls, desks, doorways, etc. It is not necessary that the absolute position of these objects is known for further navigation to be carried out. Mapless navigation falls into three sub-groups: Navigation using optical flow, navigation using appearance-based matching, and navigation using object recognition.

For robot navigation, object recognition has the ability to learn and detect hundreds of arbitrary objects in images from uncontrolled environments which can be considered as a major breakthrough for many intelligent robotics applications. Object recognition has many applications in mobile robotics, such as self-localization of mobile robots and object manipulation of autonomous robots. Object recognition enhances the representation of the environment that robots will use for their reasoning processes. The capacity to perceive and understand the environment is an important limitation for designing robots suitable for being deployed in human environments to perform domestic tasks or help the disabled. Also, being able to communicate at human-level semantics with its owners would make these robots much easier to control for a non-technical end user.

Object recognition in real scenes is deemed one of the most challenging problems in computer vision. The visual appearance of objects can change enormously due to viewpoint variation, occlusions, illumination changes, or sensor noise. Furthermore, objects are not presented alone to the vision system, but they are immersed in an environment with other elements which clutter the scene and make recognition more complicated. In a mobile robotics scenario, a new challenge is added to the last list: computational complexity [117]. In a dynamic world, information about the objects in the scene can become obsolete even before it is ready to be used if the recognition algorithm is not fast enough to handle the current robot's view.

4.2 Stereo Vision of Mobile Robots

Stereo vision is an important mechanism in animal and machine vision, allowing judgments to be made based on disparity between the images captured by each eye/camera. It is based on the human visual apparatus that uses two eyes to gain depth information. Stereo vision produces a doubling of the processing time in comparison to a monocular visual apparatus, because two images must be analyzed. Therefore, it is recommended to implement stereo approaches with parallel algorithms [39]. In other words, stereo vision is a technique for inferring the 3D position of objects from two or more simultaneous views of a scene [61, 163, 165]. Mobile robots can take advantage of a stereo vision system as a reliable and effective way to extract range information from the environment. Accuracy of results is usually adequate for applications such as navigation and map building. Moreover, stereo vision is a passive sensor and there is no interference with other sensor devices (when multiple robots are present in the environment). Finally, it can be easily integrated with other vision routines, such as object recognition and tracking.

To calculate the position of objects and obstacles in the robots' environment, the robot should be equipped with at least two cameras to retrieve the depth information. Stereo vision requires that two lines of sight intersect in the scene point X for which the depth information is to be processed. Figure 4.1 shows the general geometry of a stereo vision system that is used in mobile robotics. The two optical centers F and F' are associated with a baseline. The lines of sight belonging to F and F' intersect at the point

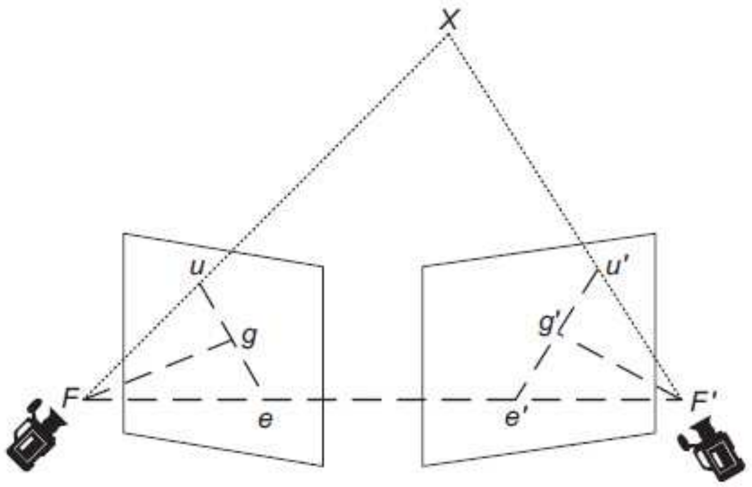


Figure 4.1: Geometry of stereo vision in mobile robotics [39].

X and generate a triangular plane that intersects every image plane in the epipolar lines g and g'. The projections u and u', respectively, of the scene point X can be found on these two lines. All possible positions of X lie on the line FX for the left image and on the line $F'X$ for the right image.

To obtain the depth information and reconstruct the 3D world by the use of at least two cameras, the stereo vision is processed in four essential stages. The first phase is the calibration of the cameras to determine each camera's line of sight. It calculates the internal and external parameters of the cameras as well as eliminating the distortion of the lenses. The second stage is to rectify the captured images by using epipolar geometry. It simplifies the problem of finding correspondences between pairs of pixels in the two images by transferring the search space from two-dimensions to one-dimension search. The third step is to find the corresponding features between the captured images and compute the disparity map (distance in pixels) of the stereo pair. The final step is to calculate the triangulation. Given the disparity map, the focal distance of the two cameras and the geometry of the stereo setting (relative position and orientation of the cameras), the world coordinates of all points in the images are computed to determine the object positions in the 3D environment. The stereo vision stages will be discussed briefly in the next subsections.

4.2.1 Camera Calibration

Camera calibration is an important step in the initialization of many stereo vision systems. The accuracy of camera parameters has a marked effect on the data obtained from a stereo vision system. It computes the mapping between points in the real world and where they arrive in the captured image. This allows graphics to be rendered into an image in the correct position. Given this information for a pair of stereo cameras, it is possible to reverse the process to compute the 3D position of a feature given its position in each image.

In stereo vision, calibration is usually dealt with by calibrating each camera independently and then applying geometric transformation of the external parameters to find out the geometry of the stereo setting. Camera calibration calculates two categories of parameters. First, it determines the intrinsic parameters (internal calibration) of the camera. These parameters are used to relate the ideal pinhole model of the camera with an actual imaging device. The calculated internal parameters of a camera are: the focal distance of the lens (f), the pixel dimension P_x, P_y, the r^{th} coefficient of radial distortion (k_1), the center of the image (O_x, O_y), and the scale factor (α_x). Second, it determines the extrinsic parameters (external calibration) that define the position and orientation of the camera within an arbitrarily defined three-dimensional coordinate system. They are needed for both the correspondence problem (determining the epipolar lines for determining point correspondences) and triangulation (for reconstruction). The world reference system is chosen to be the left camera, so the parameters to be found are the translation vector T and rotation matrix R of the right camera with respect to the left one [61, 39].

Therefore, camera calibration is a fundamental step for 3D reconstruction and in particular for stereo vision analysis. It allows not only for determining the geometry of the stereo setting (needed for triangulation), but also for removing distortions provided by common lenses to avoid errors in range determination. Two kinds of distortion can be observed in camera lenses: radial distortion and tangential distortion. Radial distortion bends the camera's line of sight and de-centering shifts the principal point from the principal axis. Tangential distortion is caused by defects in the manufacturing process.

Many calibration approaches are developed to calculate the internal and the external camera parameters. They can be classified into two categories: test-area-calibration and self-calibration approaches [39]. Test-area-calibration approaches use images where the three-dimensional world coordinates are known to derive the internal and external camera parameters. Very precisely measured test areas or reference objects are often used. The manufacturing and measuring of such reference objects requires much effort and is often afflicted with errors. The self-calibration approaches determine the external and internal camera parameters as well as the world coordinates of the reference points.

A well-known method for calibrating a camera taking into account lens distortion has been proposed by Tsai [152, 90]. The method is based on the knowledge of the position of some points in the world and the correspondent projections on the image. It requires the user to use a calibration

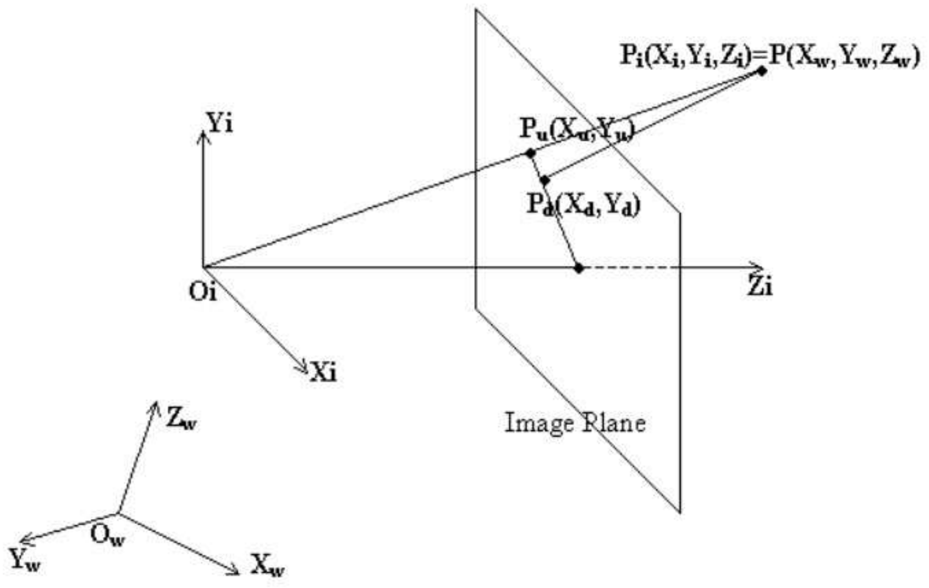


Figure 4.2: Tsai Camera re-projection model with perspective projection and radial distortion [152, 90].

grid (that must be accurately prepared) and to individuate the projections of calibration points in the image. Figure 4.2 illustrates the Tsai Camera re-projection model with perspective projection and radial distortion. The Tsai model calculates the focal length of the camera (f), the radial lens distortion coefficient (k), the coordinates of the center of radial lens distortion (C_x, C_y), the scale factor to account for any uncertainty due to imperfections in hardware timing for scanning and digitization (S_x), the rotation angles for the transformation between the world and camera coordinates (R_x, R_y, R_z), and the translation components for the transformation between the world and camera coordinates (T_x, T_y, T_z).

The origin of the three-dimensional camera coordinate system is determined by the focal point of the camera. If object coordinates are actually known in the camera coordinate system, it is possible to derive the world coordinates. The determination of the coordinates can use a stereo technique. At least two images from different positions are necessary for these purposes [39]. The mapping between 3D coordinates of features and 2D positions of their corresponding image points is given by a 4x3 projection matrix P. A graphics renderer applies a projection matrix to a feature in 3D space in order to discover where to draw the feature in the image. A real camera effectively does the same.

Many other calibration techniques exist, including techniques that rely on a mix of estimation and non-linear search, such as the algorithm described by Zhang [166]. A popular area of research involves methods of calibration which do not require a purpose-built calibration pattern, and use multiple images taken using a moving camera as input data.

On the other hand, while it is tempting to rely on algorithms for uncalibrated stereo vision, or those calibration algorithms which provide only intrinsic parameters, there are shortcomings to these approaches. Without information about the relative positions of the two cameras, stereo vision becomes less useful and more complicated. The distance between the cameras is crucial to the recovery of accurate depth information. Perhaps more importantly, the relative positions of the two cameras are necessary for rectification of the images [118].

4.2.2 Epipolar Rectification of Stereo Pairs

Epipolar rectification (or simply rectification) is a classical problem of stereo vision. It is a transformation of the coordinate systems of the two cameras, such that the image planes of the cameras are made coplanar and the scan-lines in each re-projected image are parallel to the corresponding scan-lines in the other image. The rectified images can be thought of as acquired by a new stereo rig, obtained by rotating the original cameras. This simplifies the problem of finding correspondences between pairs of pixels in the two images. In order to find a pixel in one image which corresponds to a pixel in the other, it is only necessary to search along a single horizontal scan-line of the rectified image. Without extrinsic parameters of the camera, the information required to rectify the images must be estimated.

Therefore, this is almost twice as fast as the simpler algorithm, since scanning the whole image from each camera is slower than scanning the whole of one image followed by one line in another [54]. However, it does rely on the feature being accurately located in camera 1. If a feature projects to a point (x, y) in one camera view, the corresponding image point in the other camera view must lie somewhere on an epipolar line in the camera image. An image point in camera 1 corresponds to an epipolar line in camera 2 and vice versa. If the feature is mis-located, the epipolar line is likely to be wrong. Finding the best match on this incorrect epipolar line might give a reconstructed point which is significantly inaccurate.

Fusiello et al. [42] have proposed an algorithm for stereo rectification. They assume that the stereo rig is calibrated, i.e., the cameras' internal parameters, mutual position, and orientation are known. They considered a stereo rig composed of two pinhole cameras (see Figure 4.3). C_1 and C_2 represent the optical centers of the left and right cameras, respectively. A 3D point W is projected onto both image planes, to points M_1 and M_2, which constitute a conjugate pair. Given a point M_1 in the left image plane, its conjugate point in the right image is constrained to lie on a line called the epipolar line (of M_1). Since M_1 may be the projection of an arbitrary point on its optical ray, the epipolar line is the projection through C_2 of the

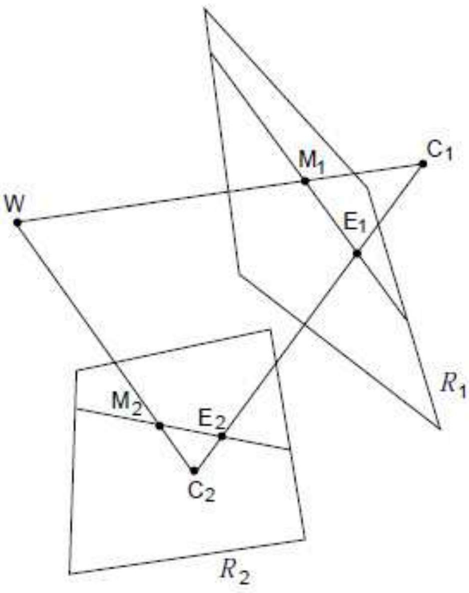


Figure 4.3: Epipolar geometry. The epipole of the first camera E is the projection of the optical center C_2 of the second camera (and vice versa) [42].

optical ray of M_1. All the epipolar lines in one image plane pass through a common point (E_1 and E_2, respectively) called the epipole, which is the projection of the optical center of the other camera.

When C_1 is in the focal plane of the right camera, the right epipole is at infinity, and the epipolar lines form a bundle of parallel lines in the right image. A very special case is when both epipoles are at infinity, which happens when the line C_1C_2 (the baseline) is contained in both focal planes, i.e., the retinal planes are parallel to the baseline. Epipolar lines, then, form a bundle of parallel lines in both images. Any pair of images can be transformed so that epipolar lines are parallel and horizontal in each image.

4.2.3 Stereo Correspondences

The computation of correspondences between features in different views is a necessary precondition to obtain depth information. Features that represent the same in different images must be found and thereafter geometrically analyzed. Approaches to the correspondence problem can be broadly classified into two categories: the intensity-based matching and the feature-based matching techniques. In the first category, the matching process is applied directly to the intensity profiles of the two images, while in the second, features are first extracted from the images and the matching process is applied to the features.

As shown in the previous section, the epipolar lines coincide with the horizontal scan-lines if the cameras are parallel, the corresponding points in both images must therefore lie on the same horizontal scan-line. Such stereo configurations reduce the search for correspondences from two-dimensions (the entire image) to one-dimension. In fact, a close look at the intensity profiles from the corresponding row of the image pair reveals that the two intensity profiles differ only by a horizontal shift and a local foreshortening.

Generally it is possible to find matching features in the captured right and left images by using many approaches. These approaches use several features to reduce ambiguities during matching process. Features can be derived from geometry, which affects the image taking, photometry, or from the object attributes. Many approaches to calculate correspondences between pixels have been developed. Some of them are outlined at this point [39]:

1. Epipolar constraint: It is used to find the corresponding pixel in the second image. The epipolar line is used to find the corresponding pixels in the second image. The search space is diminished from 2D space to 1D space.

2. Photometric compatibility constraint: The gray values of the pixels are used. It can be assumed that the gray values of corresponding pixels are nearly equal. They are probably not completely equal, because the luminosity differs due to different positions from which the images are taken.

3. Geometric similarity constraints: Geometric attributes of objects like length of lines, contours, or regions are used to get corresponding pixels. It is supposed that the attribute values are equal.

In order to minimize false matches, some matching constraints must be imposed. Below is a list of the commonly used constraints [39]:

1. Disparity smoothness constraint: This method is based on the assumption that the amount of disparity differences between adjacent pixels is similar in both images. Let P_{L1} and P_{L2} be the coordinates of two adjacent pixels in the left image. P_{R1} and P_{R2} are the corresponding coordinates in the right image, then the absolute difference computed with the following formula is small, on condition that the two images were taken from cameras arranged in parallel.

$$\| P_{L1} - P_{R1} \mid - \mid P_{L2} - P_{R2} \|$$

2. Figural continuity constraint: Additionally to the fulfillment of the disparity smoothness constraint it is required that the neighboring pixels lie on an edge in both images.

3. Disparity limit: In psychophysical experiments it has been verified that stereo images can only be merged if the disparity does not exceed a limit.

4. Ordering constraint: The corresponding pixels of objects that have similar depths have the same order on the epipolar lines of both images. This is not valid if an object point is close to the camera and additionally has a large depth difference to other objects in the scene. In this case the corresponding pixels on the epipolar lines can be sorted differently on both lines.

4.2.4 Triangulation

The triangulation problem is a small cog in the machinery of computer vision, but in many applications of science reconstruction it is a critical one on which ultimate accuracy depends. Triangulation is the problem of finding the position of a point in space given its position in two images taken with cameras with known calibration and pose. This process requires the intersection of two known rays in space. In the absence of noise, this problem is trivial. When noise is present, the two rays will not generally meet, in which case it is necessary to find the best point of intersection. This problem is especially critical in affine and projective reconstruction in which there is no meaningful metric information about the object space. It is desirable to find a triangulation method that is invariant to projective transformations of space.

Hartley and Sturm [53] solved that problem by assuming a Gaussian noise model for perturbation of the image coordinates. The triangulation problem is formulated as a least-squares minimization problem. They presented a non-iterative solution which can find the global minimum. They also show that in certain configurations, local minima occur which are avoided by their method.

4.3 Object Processing in Robot Navigation

In the vision-based robot navigation systems, human-robot interactions take place in three main modes: manual, autonomous, and semi-autonomous modes [45]. In the manual mode, the user can determine the objects of interest and these objects are recognized automatically by the robot. If the robot fails to recognize the object, then it asks the user for help. The user can interact with the robot vocally via the robot's speech recognition capability or by using any other communication tool. In the autonomous mode, the robot automatically detects the landmarks that have salient features. It then records images of the landmark from different perspectives for object recognition. In the semi-autonomous mode, the robot identifies some potential objects of interest that are distinguishable in the environment and asks if the user is interested in these landmarks or not.

Object processing in vision-based robot navigation is used to detect and classify landmarks during navigation. It is also used to localize the current position of the robot with respect to the detected landmarks. This process is called the robot's self localization. It is defined as the process of estimating the initial position of the robot with respect to a global coordinate system or according to recognized landmarks [45]. Landmark-based localization techniques are common in robotics. They can be classified into active and passive landmarks. Active landmarks are captured and transmitted to the robot for sensing and analysis, such as images acquired by the robot's cameras. Passive landmarks are detected by the robot without any transmitted signals, such as the output of a laser sensor.

As described in the last section, the landmark position in the 3D environment can be calculated by using triangulation. In the following subsections, the object detection and recognition techniques, which are suitable to the mobile robot scenarios, will be discussed in detail.

4.3.1 Object Detection and Segmentation

Most object recognition methods typically assume that the object is either already segmented from the background or that it occupies a large portion of the image. In robotic applications, there is often a need for a system that can locate objects in the environment. This means that neither of the above assumptions is valid anymore since the distance to the object and thus its size in the image can vary significantly. Therefore, the robot has to be able to detect objects even when they occupy a very small part of the image. This requires a method that evaluates different parts of the image when searching for an object. This step is denoted as object detection.

In some real-time mobile robot applications, vision systems are employing region segmentation by color to detect objects or landmarks during interaction with humans or navigating in a dynamic world. Traditionally, systems employing real-time color-based segmentation are either implemented in hardware, or as very specific software systems that take advantage of domain knowledge to attain the necessary efficiency. These detected image regions or the whole image can be decomposed into segments. All contained pixels must be similar in these segments. Pixels will be assigned to objects in the segmentation phase. If objects are isolated from the remainder of the image in the segmentation phase, feature values of these objects must be acquired in the detection phase. The features determined are used in the recognition phase to perform the object classification.

For example, Bruce et al. [13] implemented a segmentation system capable of tracking several hundred regions. It can classify each pixel in a full resolution captured color image, find and merge regions of up to 32 colors, and report their centroid, bounding box and area at 30 Hz. Michel et al. [102] built an occupancy grid from the synthesized top-down floor view; a step of color segmentation was performed in YUV space, which provides robustness to color intensity changes due to variability in environmental lighting. They defined segmentation thresholds by sampling pixel values offline for obstacles placed in a variety of locations on the floor. To eliminate noise, a pass of erosion/dilation using a rectangular structuring element is performed, followed by a step of connected component labeling that groups pixels belonging to the same obstacle.

4.3.2 Object Recognition

In order to make robots useful assistants to people in everyday life, the ability to learn and recognize objects is of essential importance. Object recognition in real scenes is one of the challenging problems in computer vision, as it is necessary to deal with difficulties such as viewpoint changes, occlusions, illumination fluctuations, background clutter, or sensor noise. Furthermore, in a mobile robotics scenario a new challenge is added to the list: computational complexity. All these complications make object recognition in real scenes a difficult problem that will demand a significant research effort in the coming years.

Object recognition algorithms are typically designed to classify objects into one of several predefined classes assuming that the segmentation of the object has already been performed. In general, object detection tasks are much more difficult. Their purpose is to search for a specific object in an image while not knowing beforehand if the object is present in the image or not. Most of the object recognition algorithms may be used for object detection by scanning the image for the object. Regarding the computational complexity, some methods are more suitable for searching than others.

In general, approaches to solving the recognition problem can be classified into two categories: appearance-based (or so-called global) methods, and model-based (or so-called local) methods [119]. Appearance-based methods are based on the overall visual appearance of the object. They often represent the object with a histogram of certain features extracted during the training process, such as a color histogram which represents the distribution of object colors. Whereas model-based methods rely on specific geometric features of the object such as small texture patches or particular features. For the robot to recognize an object, the object must appear large enough in the camera image. If the object is too small, local features cannot be extracted from it. Global appearance-based methods also fail to recognize the object, since the size of the object is small in relation to the background, which commonly results in a high number of false positives. A more natural approach in terms of computational efficiency is the use of appearance-based methods for providing a rough initial estimate followed by a refinement step using model-based methods, to estimate the full pose of the object [31, 33]. In addition, this proposed method shows a significant robustness with respect to scaling and translations.

Recently significant work has been done in visual object classification, but few of the results actually scale to the demands posed by mobile robot scenarios. In robotic applications, the robots should have lightweight, fast, and robust object perception methods that allow them to interact with the environment in real-time. In the remaining part of this section, we will represent some recent object and landmark recognition methods which are used and suitable for mobile robot applications.

Scale-Invariant Features Transform

Lowe [95] proposed an object recognition method that uses Scale Invariant Features Transform (SIFT). SIFT is an approach for detecting and extracting local feature descriptors that are reasonably invariant to changes in rotation, scaling, small changes in viewpoint, illumination, and image noise [132]. This object recognition approach is a single-view object detection and recognition system with some interesting characteristics for mobile robots, most significant of which is the ability to detect and recognize several objects at the same time in an un-segmented image.

Another interesting feature is the Best-Bin-First algorithm used for approximating fast matching, which reduces the search time by two orders of magnitude. SIFT can be used to detect and classify the objects in real-time. Its average recognition time is approximately one second. On the other hand, this algorithm has two significant drawbacks. First, it performs poorly when recognizing sparsely textured objects [3]. Second, with repetitive textures of the methods based on local features such as SIFT, which can only find the reliable features when the object occupies a significant part of the image, it is very hard to recognize objects that are far away from the camera.

With respect to the efficiency of the SIFT approach, many researchers have used it in robotic systems. For example, the authors in [34, 32, 33] use Receptive Field Co-occurrence Histograms (RFCH) for generating hypotheses of object locations and then use a SIFT-based method for object recognition once the object is zoomed-in. RFCH is an appearance-based method capable of detecting objects far away from the camera. Once a hypothesis is zoomed in, they again use RFCH for matching. If the match value exceeds the threshold, they perform SIFT-matching to verify the hypothesis. The more SIFT-matches found in an image, the more likely it is that the image contains the object. If the number of matches exceeds an object-dependent threshold, the object is considered recognized.

Sjö et al. [134] presented a method for search and localization of objects with a mobile robot using a monocular camera with zoom capabilities. They showed how to overcome the limitations of low resolution images in object recognition by utilizing a combination of an attention mechanism and zooming as the first steps in the recognition process. The attention mechanism is based also on RFCH and the object recognition on SIFT feature matching. The authors presented two methods for estimating the distance to the objects which serve both as the input to the control of the zoom and the final object localization.

Ribes et al. [118] used SIFT with some improvements in order to obtain better results by using panoramic images from a mobile robot. They used one training image per object in the library, storing its descriptors along with their object ID in a descriptor database. All training images are taken from a 1 Mpx digital camera. Testing images are panoramas built from 36 images acquired with the pan-tilt camera mounted on top of a mobile robot, storing the set of SIFT descriptors extracted from image regions used to construct the panorama. The position of each descriptor is relative to the total panorama size, not to the individual images. The panoramic image is used only to show recognition results, as the internal representation used is the descriptor set. Their implementation of the SIFT object recognition method differs from the original in the key point matching method.

In [117], the authors make an evaluation of the SIFT object recognition method in a challenging dataset, focusing on issues relevant to mobile robotics. The method presents robustness to the typical problems of images acquired in the robotics domain, but its good performance was limited mainly to well-textured objects. Several modifications and improvements of the original method are proposed in order to adapt it to the domain of mobile robotics.

Bag of Features Approach

On the other hand, Nistér and Stewénius [110] developed the bag of features approach (BoF) to recognizing segmented objects. This algorithm comes from the text categorization domain, where the occurrence of certain words in documents is recorded and used to train classifiers that can later recognize the subject of new texts. A histogram of descriptor occurrences is built to characterize an image. In order to limit the size of the histogram, a code-book or vocabulary computed by applying a clustering method to the training descriptors is used. A hierarchical vocabulary tree is used, as it allows the coding of a larger number of visual features and simultaneously the reduction of the look-up time in proportion to the number of leaves. The vocabulary tree is built using hierarchical k-means clustering, where the parameter k defines the branch factor of the tree instead of the final number of clusters.

One of the drawbacks of the bag of features method is that it is designed to work with an accurate segmentation stage prior to classification which can be very time consuming [3, 89]. The second drawback is that if one image contains background with a value greater than a certain threshold, the probability of miss-classification increases. The third drawback is that if a particular image contains two objects, there is no way to recognize both.

Color-Based Techniques

Color-based object recognition, where objects of interest are colored in a uniquely identifiable and known way, is a technique that has found wide use in the robotics community. Correspondingly, there are now a number of fast color segmentation vision libraries that are available such as CMVision [12] and OpenCV [9].

On the other hand, there are many researchers who implemented their mobile robot vision applications by using color-based object recognition methods. For example, the authors in [31] proposed a recognition scheme that is based on the color co-occurrence histograms (CCHs). It is used in a classical learning framework that facilitates a "winner-takes-all" strategy across different scales. The detected "windows of attention" are compared with training images of the object for which the pose is known. The orientation of the object is estimated as the weighted average among competitive poses, in which the weight increases proportional by the degree of matching between the training and the segmented image histograms. The optimal color scheme for an object histogram is determined by K-means clustering, in which cluster centers are distributed according to pixel density in color space.

Fasola and Veloso [37] described an approach that performs visual object detection in real-time by combining the strength of processing the color segmented image along with that of the grayscale image of the same scene. They used color segmented images for producing initial hypotheses for the location of robots in the image, and grayscale images for final classification purposes. Using both representations to process a scene allows each to make up for the deficiencies of the other, and provides a good balance between fast processing time and high detection accuracy.

In [70], the object tracking system is implemented by using moving color and shape information. A group of candidates for objects is extracted using the color distribution and the motion information. The system decides final object regions using a signature parsing algorithm. Then, it suggests the tracking method for detected object regions. The basic stages of the system can be summarized as follows. First, the normalized RGB color distribution and moving color information are combined for robust separation between the object and the background. Second, this method shows that the objects are segmented and extracted well using the signature parsing method, regardless of the shape variation. Third, recovering noises and unexpected variation is important for robust object tracking, the major control points of shape information are defined to the boundary region of the moving object to guarantee the tracking performance. Finally they show one application of a mobile robot avoiding obstacles and tracking the special object.

Artificial Neural Network

Artificial neural networks (ANNs) are among the most powerful object classifiers. They only use object shape information as input. In mobile robotics, there are a number of existing algorithms that have been developed using ANNs to recognize landmarks. For example, Gopalakrishnan et al. [45] implemented a vision-based system for semi-autonomous mobile robot navigation. Initially, the robot can localize itself in an indoor environment with its laser range finder. Then, the user can teleoperate the robot and point out the objects of interest via a graphical user interface. In addition, the robot can automatically detect potential objects of interest. The objects are automatically recognized by the object recognition system using Neural Networks. If the robot cannot recognize an object, it asks the user to identify it. The user can ask the robot to navigate back autonomously to an object recognized or identified before. The human and the robot can interact vocally via an integrated speech recognition and synthesis software component. Their ANN-based object recognition system is initially trained with a group of known object types such as ball, cylinder, trashcan, fire extinguisher, cube, or cone. In this technique a library of images of the objects of interest is created and used for training the ANNs.

In [30], the authors addressed the problem of automatically selecting and predicting landmarks for use in way-finding on a mobile robot. They employed back-propagation to teach a multilayer neural network to predict the image location and appearance of future landmarks based on the appearance and image location of currently visible landmarks. The authors used a hybrid reactive/deliberative architecture. The reactive component collects candidate landmark feature measurements while the robot explores its (previously unknown) environment. For the deliberative component, they used a neural network running on a Beowulf cluster to process the large amount of data being collected from the camera.

4.4 Summary

Vision is one of the most powerful and popular sensors used for mobile robot navigation that continues to demand a lot of attention from the mobile robot research community. For autonomous navigation, vision-based robotic systems are considered as one of the most challenging problems because the visual appearance of objects can change enormously due to viewpoint variation, occlusions, illumination changes, or sensor noise.

To obtain depth information and reconstruct the 3D world by the use of at least two cameras, the stereo vision is processed in four essential stages. The first step is the calibration of the cameras to determine each camera's line of sight and also to retrieve the camera's parameters. Then, the captured images are rectified by using epipolar geometry. The third phase is to find correspondence features between the captured images and compute the disparity map of the stereo pair. Finally, the triangulation is calculated to determine the position of the landmarks in the 3D environments.

To recognize landmarks or obstacles during robot navigation, there are many object recognition techniques which can be robustly used for the mobile robot applications in real time. SIFT, bag of features, color techniques, and neural networks are some of the most often used approaches in classifying objects in robotic systems.

CHAPTER 5

Humanoid Robot Motion Planning

Over the years, motion planning has become a major research theme in robotics. Its main goal is to enable robots to automatically compute their motions from high-level descriptions of tasks and models acquired through sensing. It is used in building many obvious applications within robotics. Museum tour guides, search and rescue, medical surgery, assembly and disassembly, and planetary exploration are some of many examples of robotics applications that need motion planning. Nowadays, motion planning is no longer restricted to just robotics applications. It also plays an important role in animation, virtual environments, computer games, computer aided design and maintenance, and structural analysis and fold proteins in biology. Beside the robotics applications, these new applications increasingly motivate the progress in motion planning techniques and algorithms.

In recent years, many motion planning algorithms have been introduced. They have had remarkable success in solving challenging motion planning problems, such as handling many degrees of freedom (DOFs), considering physical and dynamic constrains of the real robots, and dealing with uncertainties in both motion and sensors. In this chapter, an overview of the current research efforts in motion planning for mobile and humanoid robots is discussed. Then, the obstacle avoidance problem and some issues related to the real robot motion planning are presented. Finally, the sampling-based motion planning algorithms are elucidated.

5.1 Robot Motion Planning

Motion planning is considered as one of the most important issues of building autonomous or semi-autonomous robotic systems [143]. The motion planning problem has been studied for several decades and there are many algorithms that have been described in the literature [85, 25, 88]. It can be characterized by the ability of computing a collision-free feasible path for a mobile robot from a given initial position to a destination position through a workspace populated with stationary or moving obstacles. In some applications, the motion planning problem can be defined to maintain a set of constraints in the state of the world such as following a target, achieving knowledge about the world, or exploration in an unknown environment. Therefore, there are many different aspects to the motion planning problem, such as optimal path planning among rectangular obstacles, optimal path finding among weighted regions, and path planning to traverse narrow passages [154].

In the past, research on motion planning used to be neatly divided between theory and practice. Today, this distinction has largely disappeared. Most recent contributions to the field combine effective algorithms tested on significant problems along with some formal guarantees of performance. On the other hand, in the '80s and part of the '90s, finding collision-free paths was the main or only goal of the motion planning problem. Today, while obstacle avoidance remains a key issue, other important constraints are considered as well such as visibility, coverage, kinodynamic, optimality, equilibrium, and uncertainty constraints [25]. These constraints make motion planning problems more interesting and entail the implementation of more useful algorithms for real mobile robots.

In the last ten years, the significant progress in stable dynamic bipedal walking has been leading to an increased research interest in developing autonomous navigation strategies tailored specifically to humanoid robots. As autonomous navigation becomes an increasingly important research topic for humanoid robots, efficient approaches to perception, mapping, and motion planning, which are suited to their unique characteristics, will be required to integrate them easily in their typical operating environments. The ability of legged robots to step not only around but also over and onto some obstacles makes them particularly well-suited for environments designed for humans, which often contain a wide variety of objects and obstacles such as furniture, stairs, doors, and uneven ground [102]. In addition to the bipedal walking of the humanoid robots, the large amount of their degrees of freedom (DOFs) should be considered while developing practical motion planning techniques for them. Typically, humanoid robots have 20 or more DOFs which must be controlled very carefully in order to maintain overall static and dynamic stability. These constraints severely restrict the set of allowable configurations and prohibit the direct application of existing motion planning techniques [77]. Motion planning techniques for humanoid robots should pose these particular challenges during the design phase.

There exists an extensive literature on the motion planning problem in 2D static environments. Previous research with wheeled robots usually modeled a robot as a 2D circle and planned a path on a 2D grid map. These robots use a laser range sensor or a stereo vision sensor to generate a 2D map for planning. For example, Jan et al. [62] used a single circle and multiple circle modes for the robot to solve the narrow passages piano mover's problem in a 2D simulated environment. In this approach, a single cell object travels along the optimal path in a grid plane without any collision. For the multiple circles model, the checkpoints of the single cell object are ensured to be collision-free regarding all of the obstacles. The concepts involved in this algorithm are simple and can be implemented in the image plane or grid plane.

On the other hand, some attempts are reported in 3D motion planning for simulated and real humanoid robots. For example, Lau and Kuffner [86] presented a behavior planning approach to automatically generate realistic motions for animated characters. Motion clips are abstracted as high-level behaviors and associated with a behavior finite-state machine (FSM) that defines the movement capabilities of a virtual character. During runtime, motion is generated automatically by a planning algorithm that performs a global search of the FSM and computes a sequence of behaviors for the character to reach a user-designated goal position. Shiller et al. [129] implemented a practical motion planner for animated human figures that can also be used for humanoid robots. They focused on path planning and sensor-based recognition of the environment. The human motions are modeled as a sum of rigid body and cyclic motions. They identified body postures that represent the rigid-body part of typical motion patterns. This leads to a model of the configuration space that consists of a multi-layered grid, each layer corresponding to a single posture. A global search through this reduced configuration space yields a feasible path and the corresponding postures along the path.

Numerous research groups worldwide concentrate on the design and implementation of various motion planning algorithms that consider the bipedal capabilities of humanoid robots. For example, Bourgeot et al. [8] proposed a method to generate footprints from a reference path. This method finds a path and footprints in terrain under robot stability and motion continuity between starting and goal positions. The path is planned in a 3D simulated environment. The researchers studied the biped walking problem on horizontal flat and sloping grounds. Lorch et al. [94] have developed a sensor-based planning system using a biped robot with a stereo vision sensor. They proposed a footprints planner using a local environment map based on visual sensor inputs. This planner finds the step sequence while the robot is walking in a straight line. They recognized an obstacle under the assumption that any object in the scene is a rectangle. Okada et al. [111] described the vision-based navigation system for humanoid robots with vision based floor recognition and path planning using a multi-layered body image. They utilized an RRT-Connect Planner as a path planning method. This method generates a path which connects a start position to a goal position by using randomly growing trees in configuration space. Path smoothing is performed on the final path to reduce jaggedness.

Kuffner et al. [77] developed a footprints and leg trajectory planner for humanoid robots using a global method. This method enables the robot to step over obstacles and consider global information to cope with local minima. Their approach adapts a variation of the randomized planner to compute full-body motions for humanoid robots that are both dynamically stable and collision-free. The first phase computes a statically stable, collision-free path and the second phase smoothes and transforms this path into a dynamically stable trajectory for the entire body. They employed a randomized search strategy based on Rapidly-exploring Random Trees (RRTs) [38]. Their algorithm has some limitations. First, their current implementation of the planner can only handle a fixed position for either one or both feet. Second, the effectiveness of different configuration space distance metrics needs to be investigated. Finally, they currently have no method for integrating visual or tactile feedback.

5.2 Configuration Space

To create motion plans for robots, the position of the robot must be specified precisely. More specifically, a specification of the location of every point on the robot must be calculated to ensure that no point on the robot collides with the obstacle. Most of the current approaches for robot path planning are based on the concept of configuration space introduced by Lozano-P*é*rez [96]. Therefore, the concept of the configuration space is first introduced and then the difference between the path and motion planning is discussed.

The configuration of the robot system is defined as a complete specification of the position of every point of that system. The configuration space of the robot system (C-space or Q) is the space of all possible configurations of the system. Thus a configuration (c) is simply a point in this abstract configuration space. The number of DOFs of a robot system is the dimension of the configuration space, or the minimum number of parameters needed to specify the configuration [25]. To illustrate these definitions, consider a circular mobile robot that can translate without rotating in the plane. A simply way to represent the robot's configuration is to specify the location of its center, (x, y), relative to some fixed coordinate frame. If the radius r of the robot is known, then the set of points occupied by the robot can be easily determined from the configuration $c = (x, y)$ as follows:

$$R(x, y) = \{(\acute{x}, \acute{y}) | (x - \acute{x})^2 + (y - \acute{y})^2 \leq r^2\} \tag{5.1}$$

As seen from the last equation, these two parameters, x and y, are sufficient to completely determine the configuration of the circular robot. Therefore, for the circular mobile robot, the configuration space can be represented by $\mathbb{R}^2$ once a coordinate frame in the plane has been chosen. Hence, robots that move in a two- or three-dimensional Euclidean ambient space are represented by $\mathbb{R}^2$ or $\mathbb{R}^3$, respectively. This ambient space is referred to as the workspace (W) [25]. Consequently, a robot with k $DOFs$ can be described by k values, which can in turn be considered as a single point in a k-dimensional C-space of the robot. This configuration is considered free if two parts touch and blocked when two parts overlap [143].

According to the last definitions of configuration and configuration space, the obstacle configuration space (C_{obst}) is defined as a set of all configurations in C-space at which the robot is in collision with some obstacle in the workspace, i.e.:

$$C_{obst_i} = \{c \in C | R(c) \cap W_i \neq \emptyset\} \tag{5.2}$$

Otherwise, the free space or free configuration C_{free} is the set of configurations at which the robot does not intersect with any obstacle, i.e.:

$$C_{free} = C \diagdown (\cup_i C_{obst_i}) \tag{5.3}$$

With this notation, a free path is defined to be a continuous mapping $p : [0,1] \rightarrow C_{free}$, and a semi-free path to be a continuous mapping $p : [0,1] \rightarrow cl(C_{free})$, in which $cl(C_{free})$ denotes the closure of C_{free}. A free path does not allow contact between the robot and obstacles, while a semi-free path allows the robot to contact the boundary of an obstacle [25]. Therefore, The motion planning problem can be defined as follows [154]:

> **Given an initial and a goal configuration** $c_{start}, c_{goal} \in C_{free}$**, find a continuous path** $p : [0,1] \rightarrow C_{free}$ **where** $p(0) = c_{start}$ **and** $p(1) = c_{goal}$**.**

The above definition describes the geometrical version of the motion-planning problem. It is usually known as path planning. This is because the planning algorithm is only asked to return a path without considering the robot's ability to implement that path. This is not an essential issue if a robot is moving slow enough and the dynamic constraints such as friction, gravity, etc. can safely be ignored. Otherwise, there is an increasing interest in planning problems where the dynamic constraints can no longer be ignored. For those cases, planning algorithms need to come up not only with a geometrical path, but rather with what is called a motion planning, i.e. a complete description of what controls need to be applied so the robot can execute a feasible and collision-free trajectory to reach its goal [154].

5.3 Obstacle Avoidance

Global navigation strategies for mobile robots can usually be obtained by searching for a collision-free path in a 2D environment. Because of the low dimensionality of the search space, very efficient and resolution-complete algorithms can be employed. Global path planning and obstacle avoidance strategies for mobile robots and manipulators have a large and extensive history in the robotics literature [67, 35, 62, 123, 111, 78]. In the configuration space, the robot is reduced to a point. Hence, the motion planning problem becomes a question of finding a path for a point from an initial point to a goal point in the configuration space. This contrasts with previous methods which plan directly in the workspace, using methods such as swept volumes to determine whether or not a path was feasible (i.e., did not collide with an obstacle). However, planning in C-space poses a problem: unlike the obstacles in the workspace, which are well-defined, how does one represent invalid configurations in C-space (i.e. C_{obst})? [92].

For wheeled robots, many solutions on this subject have been presented using ultrasonic sensors or laser range finders and they mainly detect walls and relatively large obstacles around the robot. The traditional collision avoidance approach is based on expanding the obstacles by a radius r equivalent to the robot's largest dimension, hence planning as if the robot could navigate as a point in the environment. This over-simplification, however, is not suitable for the case of large robots in constrained spaces, as expanding the obstacles along narrow passages will effectively block the passage. A more suitable solution is proposed in [143] by finding the largest possible expansion radius R that allows the robot to pass through the narrowest path and then divide the area of the robot into circles of that radius. The center points of those circles will then be used to check for obstacle collision.

From another point of view, solving the problem of obstacle avoidance for a humanoid robot in unstructured or unknown environments is a big challenge, because the robot can easily lose its stability or fall down if it hits or steps on an obstacle. Otherwise, biped humanoid robots have the unique ability to traverse obstacles by stepping over or upon obstacles in a cluttered terrain [103, 24, 76].

Conservative global navigation strategies for biped humanoid robots can be obtained by choosing an appropriate bounding volume (e.g. a cylinder), and designing locomotion gaits based on stereo vision system outputs for following navigation trajectories computed by a 2D path planner. However, this always forces the robot to circumvent obstacles. As the humanoid robots are non-holonomic, they cannot turn in place without requiring additional space. The trajectory of the body center describes a curve similar to a car-like robot, although the turn radius is generally quite small. It is possible to account for the extra turning space in a cylinder model of the robot enlarged by twice the turn radius [123]. In the cylinder model, the shape of the robot is approximated by two cylinders sharing the same axis. Cylinders allow for fast collision checking and the smaller cylinder for the legs enables the robot to pass close to low obstacles where upper body and arms are above the obstacle [46]. Cylinder models of robots make it possible to perform this check in constant time if the distance to the nearest obstacle is known; the distance is simply compared to the radius of the cylinder.

5.4 Robot Motion Planning Constraints

In the area of mobile robotics, it is an interesting and challenging goal to try and embed a motion planner in a real robot as a black box that can automatically drive a robot to wherever its goal might be. For such functionality in real-life scenarios, there are various constraints and difficulties that need to be addressed on top of the basic geometrical motion-planning problem. Robot dynamics, time-changing workspaces, real-time planning, dealing with uncertainty in motion and sensors, and consequently problems in localization and mapping are important constraints which should be considered when dealing with real robots [43, 154]. This section tries to identify some of those issues, and show how motion planners are being adapted to deal with these problems.

5.4.1 System Dynamics

Some motion planning algorithms have the underlying assumption that the robots are 2D circles moving in 2D static environments. Other planning algorithms consider the robots as free-flying objects with 6DOFs (three translational and three rotational) in a 3D workspace moving in a static workspace. One crucial extension towards more physical realism is to try and take into account dynamic constraints of the robots. A real robot is not a circle or a "free-flying" object. It has motor limitations that cannot generally be ignored. These limitations, which are called kinodynamic constraints, impose bounds on its maximum velocity and acceleration [154]. These constraints can significantly increase the complexity of motion planning, as the robot might be incapable of implementing certain collision-free paths (i.e. infeasible paths). Furthermore, real robots are subject to other physics-based constraints such as gravity and friction that can and sometimes need to be taken into account. Motion planning algorithms that account for system dynamics (i.e. robot and physical constraints) typically approach the problem in one of two ways: (1) decoupling the problem by first computing a kinematic path, and subsequently transforming the path into a dynamic trajectory, or (2) searching the system state-space directly by reasoning about the possible controls that can be applied [77].

To handle kinodynamic constraints, a very common approach, which is called a decoupled trajectory planning, is used to solve motion-planning problems [14, 71]. First, a path-planning algorithm computes a collision-free trajectory ignoring system dynamics. Then, a controller is needed to compute appropriate controls that will implement the desired path to generate feasible motions. There are a number of issues in this approach which should be considered. Typically, controllers alone cannot avoid obstacles in the environment, and that is why an obstacle-free path must be found in another way first. Moreover, the produced geometrical paths may be infeasible for a real robot and even when the controller manages to follow a desired path, this may require the robot to move slowly to minimize the influence of dynamic and physical constraints. Finally, controllers are system-specific, and as today's robots become increasingly complex it becomes very hard to develop good controllers.

In addition to the previous dynamic constraints, different directions should be addressed in the motion planning algorithms depending on the shape and walking style of the robot. Recently, the biped humanoid robots have made considerable progress in stable dynamic walking, stable walking trajectories, and dynamic balance and control [24]. These research problems have been developed to generate complete global navigation strategies for biped robots. In particular, the ability to autonomously select footstep placement positions to avoid obstacles while walking is an important step towards improved navigation autonomy for humanoid robots.

5.4.2 Time-changing and Unknown Workspaces

Another extension is to relax the static known workspace assumption. This is another important extension that is necessary for robots that are not restricted to operating in a highly-controlled, stationary environment. The difficulty of the motion planning in such cases can vary based on what is known about the environment and the moving obstacles. In the best case, the obstacles are executing repetitive motions and information about their maximum velocity or acceleration is available. Then, the obstacles can be handled in the motion planning algorithm. However, it could be that the moving obstacles are unpredictable or even malevolent and moving arbitrarily fast. In these cases, guaranteeing overall collision avoidance for a robot may be impossible.

In the time-changing or unknown workspaces, the robot first builds a roadmap in the state x time space of the environment and finds an initial plan that takes any known dynamic obstacles into account. Then, as the robot starts executing the plan, it is possible that new obstacles might be observed that invalidate the plan. There are two main directions for addressing this problem. One assumes the movement of obstacles to be predictable, in which case time can be considered an extra parameter of the configuration space. The other direction is to use roadmaps that permit updates. This is a more general method but raises the problem of updating the roadmap in a useful and efficient manner [154].

5.4.3 Real-time Motion Planning

Real-time planning in a dynamically changing environment is considered as one of the most interesting classes of motion planning problems. A robot moving in an unknown and/or changing environment needs to change its plan rapidly, depending on the latest sensor inputs. Therefore, in environments that are changing in time, the robot is expected to react to these changes and re-plan in real-time while moving. For this reason, robots have to gather new sensory information periodically, and then re-plan using the latest available information. Finally, all these considerations become more important when the robot's dynamics are also taken into account. Therefore, if the robot is limited by its dynamic constraints, it cannot instantaneously change its behavior. All of these considerations have brought up the issue of safety. It is no longer enough to simply produce feasible trajectories that are collision-free with respect to static or moving obstacles. The trajectories also have to be safe so that the robot never finds itself in what is called an Inevitable Collision State or ICS [154]. Being in ICS means that due to dynamic constraints, the robot will collide with an obstacle in the future no matter what controls are applied from that state on.

5.4.4 Handling Uncertainties

In the last three decades, many approaches to planning robot motion with uncertainty were developed [144, 7, 19]. As these approaches deal with large cumulative uncertainty, environment sensors are often used by higher-level, slower-rate control loops to reduce uncertainty. Then, a separate estimate of the current robot's configuration is computed by matching the incoming sensory data against a prior model of the environment. This estimate is called the sensed configuration [144]. As environment sensing is not perfect, the sensed configuration is not error-free either. However, unlike the dead-reckoning estimate, the error in the sensed configuration does not depend on past history, nor does it result in cumulative uncertainty. It mainly depends on the portion of the environment that is currently perceptible by the sensors (in addition to being a function of the intrinsic quality of the sensors). If appropriate environment features can be identified and localized by the sensors from the current actual robot configuration, a sensed configuration with relatively small uncertainty can be computed.

5.5 Sampling-based Motion Planning

Motion planning has been studied for several decades and many motion planning algorithms have been proposed in the literature [85, 25, 88]. Formerly, robot motion planning algorithms were classified with respect to the processing scope into either being global or local planners [59]. A global planner is one that assumes complete knowledge about its environment, whereas a local planner assumes partial knowledge about its surrounding environment. Lately, researchers have started to look at the problem in a more general and realistic form by considering some difficulties such as navigation in higher-dimensional spaces, robot dynamics, and time-varying environments.

A number of algorithms have been introduced and they have had remarkable success in solving the motion planning problem, like grid-based search approaches, geometric methods, potential field algorithms, and sampling-based planners. Low-dimensional problems can be solved with grid-based algorithms that overlay a grid on top of the configuration space, or geometric algorithms that compute the shape and connectivity of C_{free}. Otherwise, exact motion planning for high-dimensional systems under complex constraints is computationally intractable. Potential-field algorithms are efficient, but fall prey to local minima (an exception is the harmonic potential fields). Sampling-based algorithms avoid the problem of local minima, and solve many problems quite quickly. They are unable to determine that no path exists, but they have a probability of failure that decreases to zero as more time is spent.

Sampling-based algorithms are currently considered state-of-the-art for motion planning in high-dimensional spaces [92, 125, 18]. Therefore, they have been applied to problems which have dozens or even hundreds of dimensions, such as robotic manipulators, biological molecules, animated digital characters, and biped humanoid robots. In the remaining part of this section, the sampling-based motion planning algorithms will be discussed in detail.

For practical purposes, complete algorithms such as cell decomposition and visibility roadmaps turn out to be computationally expensive and hard to implement. Adding various restrictions to the problem made the use of complete algorithms possible. For the general case of the problem, a breakthrough was achieved with the development of sampling-based motion planners [88]. The ultimate goal for such methods is to generate plans that can be executed with few modifications in real mobile robot platforms [154]. These algorithms quickly became popular for various reasons. Many previously considered hard problems could be solved using sampling-based motion planners, while the fundamental ideas behind these planners were in general easy to describe and implement. The increased performance of these algorithms comes at the cost of sacrificing completeness, which is due to the fact that a set of sampling points are used to represent the C-space that is used in constructing solutions [143]. These algorithms can only guarantee probabilistic completeness instead. A probabilistically complete algorithm will eventually find a solution if there is one, but it will run forever if no solution exists.

From another point of view, sampling-based motion planning is thus fundamentally different from earlier approaches to motion planning since its model of available information about C-space is substantially restricted. It uses only information from a collision detector as it searches the configuration space. This restriction eliminates many of the problems encountered in methods that constructed a representation of C_{obst}. Since there is no explicit model of C_{obst}, there is no need to characterize all possible contact conditions for particular classes of problems, nor to compute the contacts to solve a given problem. Also, a sampling-based motion planner can apply to a broad class of motion planning problems because it treats collision detection as a separate module, which may be tailored to a particular kind of problem. For these reasons, sampling-motion planning algorithms often seem strikingly simple in comparison to combinatorial motion planners. The simplicity and generality of these planners, along with increases in computation power and the development of efficient collision detection algorithms, have resulted in the introduction of a number of powerful motion planning algorithms, capable of solving challenging problems with many DOFs [92].

The sampling-based algorithms can be divided into two types: single-query and multiple-query approaches [143]. Multiple-query approaches start with a pre-processing step that usually takes a large amount of time but makes solving path planning problems in the same environment faster. The Probabilistic Roadmap planner (PRM) [25] is an example of a multi-query approach that initially used uniform sampling in constructing the path. This method was problematic because the entire C-space will be sampled with a density required by the most complex area of the environment, such as a narrow passage area [58]. Nowadays, PRMs are moving into non-uniform methods for sampling, such as the Gaussian sampling method and the bridge test, to insure that most of the configurations in C-space are actually close to obstacles or inside a narrow passage, thus reducing the unnecessary samples and decreasing the computational time.

Single-query methods were developed to avoid the large pre-computational time that the multi-query methods take, and they have been proved to be efficient. Randomly-exploring Random Trees (RRTs) [88, 38] are mainly based on single-query methods. They have gained popularity for their good performances, which has led to a number of extensions specifically targeting the solution to complicated geometrical problems. The next subsections introduce the basic ideas presented in almost all sampling-based motion planners and describe improvements in the aforementioned two categories of algorithms.

5.5.1 Roadmap-based Planners

Roadmap-based planners are typically used as multi-query planners. As their name implies, they maintain a roadmap that can be used to answer different planning queries. The main data structure being used in these planners is a graph whose nodes are points in the configuration space. Edges in this graph exist between configurations that are close to one another, and the robot can move from one point to the other without collisions [154]. Therefore, the placement of nodes (vertices) is seen as constructing a reusable roadmap in the roadmap-based planners, not as generating query sub-goals. Second, these methods attempt to connect to a more carefully-chosen subset of new nodes (sub-goals), which are typically the k nearest nodes from each connected component, or all sub-goals within some specified radius. Third, the roadmap-based planner uses a simpler local planner, often either straight-line or rotate-at-s. Finally, methods are used to identify difficult regions of C-space and sample in those regions (the "roadmap enhancement" phase). Along with the use of more sophisticated collision detection methods, these factors make the roadmap-based planners more effective for challenging motion planning problems [92].

A popular roadmap-based planning technique is the probabilistic roadmap planner (PRM) which is developed independently at different sites [25, 88]. It turns out to be very efficient, easy to implement, and applicable for many different types of motion planning problems. The basic PRM approach leaves many details to be filled in, in particular how to sample the space, what local planner to use and how to select promising pairs. Over the past decade, researchers have investigated these aspects and developed many improvements over the basic scheme [43]. The typical PRM approach consists of a preprocessing phase and a query phase. In the preprocessing phase a roadmap graph $G = (V, E)$ is constructed. In the query phase, the start and goal configurations are connected to the graph. As C_{free} denotes the part of C-space that consists of feasible configurations, then C_{free} is sampled for collision-free placements that are added as nodes to the graph G. Pairs of promising nodes are chosen in the graph and a simple local motion planner (normally a straight-line motion) is used to try to connect such placements with a path. If successful an edge is added to the graph. This process continues until the graph covers the connectedness of C_{free}. The complete list of the basic PRM planner is shown in Algorithm 1.

In a sampling-based motion planner, one of the core issues is the sampling strategy which can be considered as the most time-consuming step in PRM. It samples the configuration space of the moving object to retrieve C_{free}. Sampling is defined as the process by which new configurations are randomly selected to be added to the roadmap. Lately, there are multiple possible directions for improving sampling [154]. Some of the previous work focuses on sampling important areas of the configuration space using workspace information to derive what the important areas are. A well-known example is sampling in the areas of narrow passages. Increasing the density of sampling around narrow passages increases the chances of finding samples in areas that are hard to reach and are likely to be needed for finding a solution. Table 5.1 lists some uniform and non-uniform sampling

Algorithm 1: Probabilistic Roadmap Planner (PRM)

```
input  : n← the number of nodes in the roadmap
         k← the number of the closest neighbors to c
         $c_{init} \leftarrow$ the initial configuration
         $c_{goal} \leftarrow$ the goal configuration
output: A roadmap $G = (V, E)$
         $P \leftarrow$ a path from $c_{init}$ to $c_{goal}$

$V \leftarrow \{\}$, $E \leftarrow \{\}$;
repeat
    $c \leftarrow$ a random configuration in $C_{free}$;
    $V \leftarrow V \cup \{c\}$;
until $|V| > n$ ;
forall $c \in V$ do
    $N_c \leftarrow$ the k nodes of c from $V$;
    forall $\acute{c} \in N_c$, in order of increasing distance from c w.r.t. dist do
        if $(c, \acute{c}) \notin E$ and $\triangle\ (c, \acute{c}) \neq NIL$ then
            $E \leftarrow E \cup \{(c, \acute{c})\}$;
$N_{init} \leftarrow$ the k nodes of $c_{init}$ from $V$;
$N_{goal} \leftarrow$ the k nodes of $c_{goal}$ from $V$;
$V \leftarrow V \cup \{c_{init}\} \cup \{c_{goal}\}$;
$\acute{c} \leftarrow$ the closest neighbor of $c_{init}$ in $N_{init}$ ;
repeat
    if $\triangle\ (c_{init}, \acute{c}) \neq NIL$ then
        $E \leftarrow E \cup \{(c_{init}, \acute{c})\}$;
    else
        $\acute{c} \leftarrow$ the next closest neighbor of $c_{init}$ in $N_{init}$;
until connected or $N_{init} = \emptyset$ ;
$\acute{c} \leftarrow$ the closest neighbor of $c_{goal}$ in $N_{goal}$ ;
repeat
    if $\triangle\ (c_{goal}, \acute{c}) \neq NIL$ then
        $E \leftarrow E \cup \{(c_{goal}, \acute{c})\}$;
    else
        $\acute{c} \leftarrow$ the next closest neighbor of $c_{goal}$ in $N_{goal}$;
until connected or $N_{goal} = \emptyset$ ;
$P \leftarrow$ shortest path $(c_{init}, c_{goal}, G)$;
if $P \neq \emptyset$ then
    return P;
else
    return failure ;
```

strategies which are used in sampling-based motion planners [88, 43].

After applying a sampling strategy in C-space, the preprocessing phase of the PRM planner is processed to connect the resulting samples to each other to generate the roadmap graph (G). As connected samples play an important role in retrieving the shortest path, different techniques have been suggested to favor connections to samples [88, 25]. While the standard nearest-k method tries to connect to the nearest k nodes, the component method tries to connect the new configuration to the nearest node in each connected sample that lies close enough. The component-k method is a combination and tries to connect to at most k nodes in each connected sample. While it may seem the more samples are connected, the better, connecting samples is a time-consuming process and so a balance between the number of connections and runtime needs to be achieved [154]. Figure 5.1, adopted from [25], shows an example of a roadmap for a point robot in a 2D Euclidean space. The gray areas represent the obstacles in the environment. The empty circles correspond to the nodes of the roadmap. The straight lines between circles correspond to edges. The number of k closest neighbors for the construction of the roadmap is three. The degree of a node can be greater than three since it may be included in the closest neighbor list of many nodes.

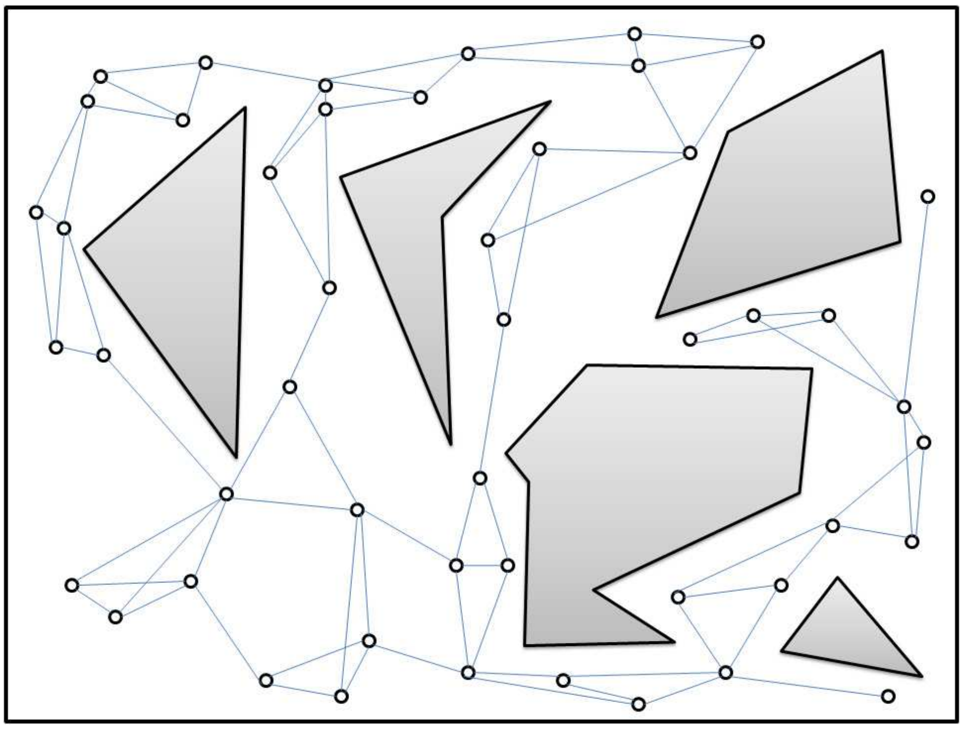


Figure 5.1: A roadmap graph for a point robot in a 2D Euclidean space [25].

Table 5.1: Sampling strategies for sampling-based motion planners.

Method	**Description**
	Uniform Sampling Strategies
Random	A sample is created by choosing random values for all DOFs.
Grid	It starts with a coarse unknown-resolution grid and refines this grid in the process to have the cell size. Grid points on the same level of the hierarchy are added in random order.
Halton	It uses Halton point sets as samples. These sets have been used in the discrepancy theory to obtain coverage of a region that is better than using a grid.
Cell-based	The first sample is generated randomly in the whole space, then the workspace is split into 23 equally sized cells. In a random order, a configuration is generated in each cell. Afterwards, it splits each cell into sub-cells and repeats this process for each sub-cell.
	Non-Uniform Sampling Strategies
Gaussian	It adds more samples near obstacles. The idea is to take two random samples, where the distance between the samples is chosen according to a Gaussian distribution. The free sample is added only if one of the samples lies in C_{free} and the other lies in C_{obst}.
Obstacle-based	A random sample is picked. If it lies in C_{free}, then it will be added to the graph. Otherwise, a random direction is picked and the sample moved in that direction with increasing steps until it becomes free and adds the resulting free sample.
Bridge Test	Two random samples are taken, where the distance between the samples is chosen according to a Gaussian distribution. The free sample is added only if both samples lie in C_{obst} and the point in the middle of them lies in C_{free}.
Medial Axis	It generates samples near the medial axis of the free space. All samples have 2-equidistant nearest points resulting in a large clearance from obstacles. The method increases the number of samples in small volume corridors but is relatively expensive to compute.
Nearest Contact	It generates samples on the boundary of the C-space and can be seen as the opposite to the medial axis technique.

To solve a particular query after finishing the preprocessing or learning phase, the start and goal configurations are added to the roadmap and a graph search algorithm is used to find a path. The efficiency of the algorithm depends on how well the roadmap can capture the connectivity of the configuration space. The path can be obtained by performing $A*$ algorithm (see Appendix C for more details) or a Dijkstra's shortest path query on the graph. Figure 5.2 shows an example of how to solve a query with the roadmap presented in figure 5.1. The configurations c_{init} and c_{goal} are first connected to the roadmap through $\dot{c}$ and $\ddot{c}$. Then a graph-search algorithm returns the shortest path denoted by the thick black lines.

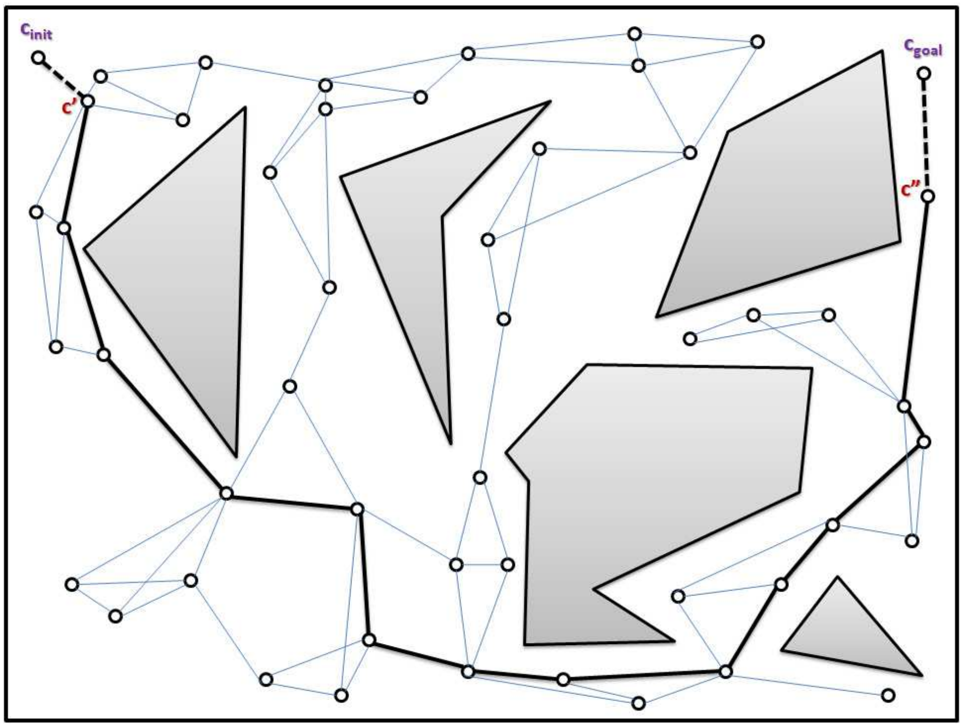


Figure 5.2: The query phase of the PRM planner [25].

From the performance perspective, the main drawback of PRM is that it heavily relies on collision checking. To mitigate this effect, algorithms like Lazy PRM [6] have been designed. Lazy PRM delays collision checks by assuming edges to be valid and actually checking them only if they are part of potential solutions. To reduce the number of collision checks even further, and achieve better coverage of the configuration space at the same time, the use of predictive models has been introduced in [16]. The idea behind predictive models is to compute an approximation of the configuration space using machine learning techniques. The approximation makes the inferring of the probability of a certain configuration collision free. Use of these probabilities is made instead of collision checking when connecting samples in the roadmap.

5.5.2 Tree-based Planners

In many cases, single query sampling-based motion planners can be used to quickly solve one particular planning problem instance. In these planners, the main data structure is typically a tree. The basic idea of these planners is that an initial sample (c_{init}) is chosen as the root of the tree. Then, newly produced samples are connected to samples already existing in the tree until it reaches to the destination configuration (c_{goal}).

Significant amounts of work have been dedicated to developing sampling and connection strategies of tree-based planners, biasing the direction in which the tree grows and achieving better coverage of the space [154, 88, 25]. These tree-based planners applied to motion planning explore the collision-free regions of the C-space trying to find a feasible path between two given configurations. The exploration is biased to solve this particular planning query and not to obtain information about the whole space. Most of the algorithms construct trees whose nodes are configurations computed during exploration. The search can be performed in unidirectional or bidirectional directions as shown in Figure 5.3. The unidirectional strategy constructs a single tree from one of the two given configurations until the other configuration is reached. The bidirectional strategy constructs one tree from c_{init} and another from c_{goal}. The solution is found when the two trees meet at a point. Choosing an unidirectional or a bidirectional search mainly depends on the characteristics of the problem to be solved [125]. The most popular representative of tree-based planners is the Rapidly-exploring Random Trees (RRTs). In the literature, there are many tree-based planners using an RRT-like algorithm as a base, such as Expensive-Spaces Tree (EST), Execution-extended RRT (ERRT), Reconfigurable Random Forest (RRF), and Dynamic RRT (DRRT).

RRTs have been shown to be effective for solving single-shot path planning problems in complex configuration spaces [38]. By combining random sampling of C-space with biased sampling around the goal configuration, RRTs efficiently provide solutions to problems involving vast, high-dimensional C-spaces that would be intractable using deterministic approaches.

The basic RRT algorithm is outlined in Algorithm 2. Beginning with the initial robot configuration (c_{init}) as the root node, it incrementally grows a tree until the tree reaches c_{goal}. The growth is performed one configuration at a time, by alternating the two steps that are common to most tree-based planners: selection and propagation. In the selection phase, a target configuration (c_{rand}) is uniformly selected at random from C-space. Then, among the samples already existing in the tree, the closest one to c_{rand} is selected (c_{near}). In the propagation phase, a new node is created by growing the tree

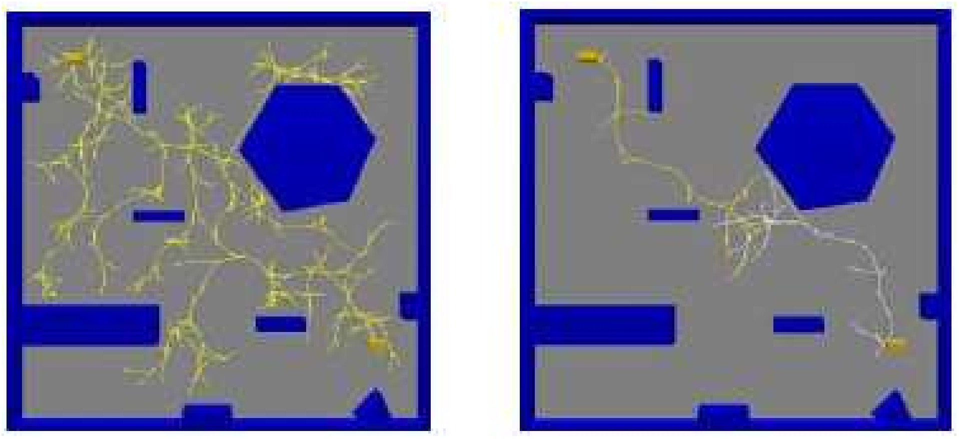


Figure 5.3: Bidirectional and unidirectional searches in tree-based planners [125].

some distance from c_{near} towards c_{rand}. If extending the tree towards c_{rand} requires growing through an obstacle, no extension occurs. This process is repeated until the tree grows to within some user-defined threshold of the goal. A very nice property that follows from this method of construction is that the tree growth is strongly biased towards unexplored areas of the configuration space. Consequently, exploration occurs very quickly.

One of the bottlenecks of RRTs is that in some environments most of the randomly selected samples will cause the expansion from the closest node in the RRT tree to fail. This produces a significant increase in the runtime of the algorithm. One way to mitigate this problem is to attach a radius to each sample in the built tree [154]. If the randomly selected sample is further away than the specified radius, another sample is picked until the distance to the nearest sample in the tree is less than the attached radius. This change reduces the likelihood of having a connection failure. Samples added to the tree are initially set to infinite radius; when a connection attempt fails from a sample; its radius is set to some workspace-dependent constant.

5.6 Summary

The significant progress in stable dynamic bipedal walking is leading to an increased research interest in developing autonomous navigation strategies tailored specifically to humanoid robots. As autonomous navigation becomes an increasingly important research topic for the humanoid robots, efficient approaches to perception, mapping, and motion planning, which are suited to their unique characteristics, will be required to integrate them easily in their typical operating environments.

To create motion plans for mobile or humanoid robots, the position of

Algorithm 2: Rapidly-exploring Random Tree (RRT)

```
input  : n← the number of attempts to expand the tree ($T$)
         $c_{init} \leftarrow$ the initial configuration where the tree is rooted
         $c_g \leftarrow$ a configuration toward which the tree is grown
output: A tree $T = (V, E)$

$V \leftarrow \{c_{init}\}$;
$E \leftarrow \{\}$;
for $i = 1$ to $n$ do
    $c_{rand} \leftarrow$ a randomly chosen from $C_{free}$;
    $c_{near} \leftarrow$ the closest neighbor of $c_g$ in $T$;
    $c_{new} \leftarrow$ progress $c_{near}$ by step-size along the straight line in
    C-space between $c_{near}$ and $c_{rand}$;
    if $c_{new} \in C_{free}$ then
        $V \leftarrow V \cup \{c_{new}\}$;
        $E \leftarrow E \cup \{(c_{near}, c_{new})\}$;
        if $c_{new} = c_g$ then
            return connected;
        else if $c_{new} = NIL$ then
            return failure;
        else
            return $NIL$;
return $T$;
```

the robot must be specified precisely. More specifically, a specification of the location of every point on the robot must be calculated to ensure that no point on the robot collides with the obstacle. Most of the current approaches for robot motion planning are based on the concept of configuration space. In addition to the robot's position specification, there are various constraints and difficulties that need to be addressed on top of the basic geometrical motion-planning problem to let the robot work in real-life scenarios, such as Robot dynamics, time-changing workspaces, real-time planning, dealing with uncertainty in motion and sensors, and consequently problems in localization and mapping.

For mobile and humanoid robots, a number of algorithms have been recently introduced to plan their motion. The grid-based search approaches, geometric methods, potential field algorithms, and sampling-based planners have had remarkable success in solving the motion planning problem. Sampling-based motion planners are considered as a breakthrough in motion planning and quickly became popular for various reasons. They avoid the problem of local minima, and solve many problems quite quickly. Their ultimate goal is to generate plans that can be executed with few modifications in real robotic platforms. Sampling-based algorithms are currently considered state-of-the-art for motion planning in high-dimensional spaces. Many previously considered difficult problems could be solved using sampling-based motion planners, while the fundamental ideas behind these planners were in general easy to describe and implement.

The sampling-based planner generates a roadmap graph for the robot's path. To compute footstep placements for biped humanoid robots from the resulting graph, a search method is needed to calculate the shortest feasible footstep sequence from the initial position to the target position. Heuristic search methods can be used to follow the roadmap graph and retrieve a shortest low-cost footstep sequence for the humanoid robots. They are fast enough to sense uncertainty, model errors, and handle obstacles for real-time re-planning in dynamic and unknown environments.

CHAPTER 6

Humanoid Robot Navigation System

A more natural interaction between humans and mobile robots can be achieved by bridging the gap between the format of spatial knowledge used by robots and the format of languages used by humans. This enables both sides to communicate by using shared knowledge. Spatial knowledge can be (re)presented in various ways to increase the interaction between humans and mobile robots. One effective way is to describe the route to the robot by using multimodal representation. This method can permit computer language-naive users to instruct mobile robots, which understand spatial descriptions, to naturally perform complex tasks using succinct and intuitive commands.

We implemented a complete navigation system for a humanoid robot to execute navigation tasks in indoor environments. The system is used by novice users to describe routes for the robot via a multimodal interface. The resulting route description is processed and represented in symbolic and topological map representations which are used as an initial path estimation for the robot. These representations are used with the output from the stereo vision to plan the path and footstep placements for the humanoid robot. In this chapter, we introduce the architecture of our Humanoid Robot Navigation System (HRNS). The main modules of HRNS are described briefly. Afterwards, the hardware and software characteristics of the experimental platform – HOAP-2 humanoid robot – are discussed. Finally, the experimental environment of HRNS is presented.

6.1 System Architecture

A more natural interaction between humans and mobile robots – with the least collective effort – can be achieved if there is a common ground of understanding [69, 10]. Most typical scenarios of interaction between humans and robots include the user who instructs a robot to perform certain actions in certain scenarios, such as moving to a location or manipulating an object. To instruct the robot to navigate in its surrounding environment, the robot navigation system should contain three basic components: planning process, navigation process, and environmental representation [116]. The planning process includes way-finding and locomotion levels of navigation [105]. It computes a mobile robot path or trajectory between the start and end points of the route. The navigation process provides the robot with the information required to move and follow the computed path or trajectory and also to plan the footstep locations of the bipedal humanoid robots. Finally, the environmental representation enables the robot to know its location and direction during navigation.

To describe a navigation task to a mobile robot, route instructions are used to specify the spatial information about the route environment and the temporal information about the move and turn actions which will be executed by the robot. Good route instructions should contain adequate information on these two aspects by considering the spatial environment of the robot and the relevant navigation and perception actions. To express the route in an effective way, the rules and sequence of commands should be expressed very concisely. Natural language uses symbols and syntactic rules to interact with the robots which dispose of represented knowledge at the symbolic level.

We have developed a navigation system for a humanoid robot to enable computer language-naive users to instruct their mobile robots by using a multimodal cognitive interface. Our main goal is to let the robot execute navigational tasks reasonably with some degree of autonomy to adapt to the user's route description. This requires both the adaptivity to dynamic or unknown environments and the ability to generate plans. Our approach is presented to qualify the humanoid robot to walk autonomously in miniature city or indoor environments.

The user describes the route to the robot verbally or graphically by using a Graphical User Interface (GUI) and the robot has no prior spatial knowledge of the environment's layout. The route description includes the start point, target point, actions, spatial relationships, and landmarks; whereas landmark locations are calculated in real-time from stereo vision. Our system combines vision-based sensing with a motion planner to allow the humanoid robot to navigate toward a desired goal position while avoiding obstacles. The motion planner computes an optimal sequence of footstep placements within a time-limited planning horizon. Footstep plans are reused and only partially recomputed as the humanoid robot discovers new findings in the route environment during navigation.

Figure 6.1 shows the main building blocks of our system and the output from each stage. The robot navigation is based on the route described by the user to generate an initial path estimation which is supplied to the

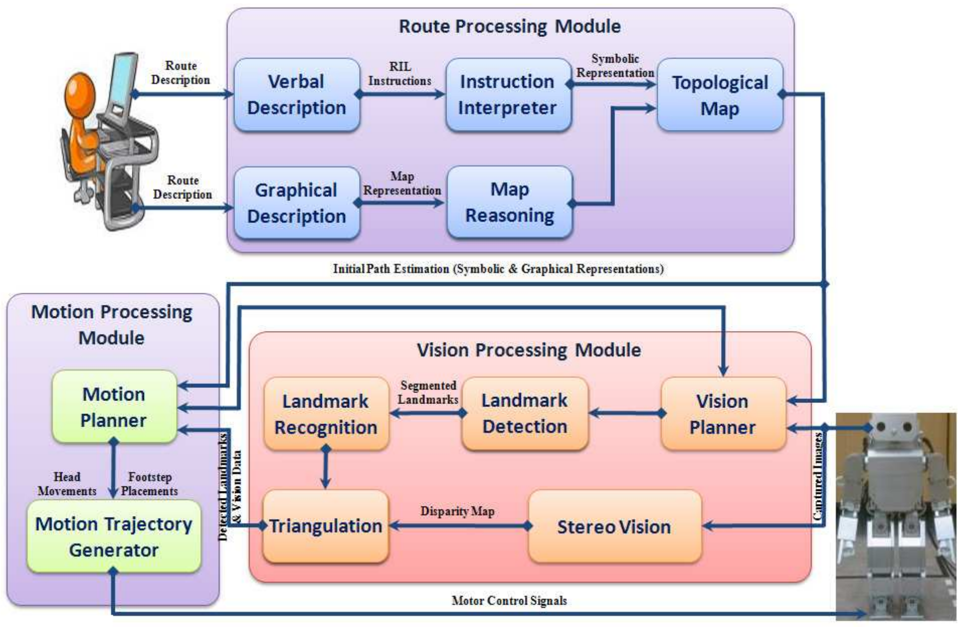


Figure 6.1: The architecture of our Humanoid Robot Navigation System (HRNS).

motion planner. The humanoid robot begins from the start point and moves along that path to collect information and recognize the landmarks by using its stereo vision. Based on the new findings and the processed route, the path is then re-planned to adjust the robot's position during navigation. The system is composed of three main modules: route processing, vision processing, and motion processing modules. These modules are discussed briefly in the next subsections.

6.1.1 Route Processing Module

The route processing module receives and processes the route description to generate symbolic and topological route representations. We present a spatial language to describe route-based navigation tasks for a mobile robot. In addition, we also present a graphical representation to provide the user with a simple interface to sketch the route for the robot. The instructions of this spatial language and graphical representation are implemented to provide an intuitive interface with which the computer language-naive user can easily and naturally describe a navigation task to a mobile robot in any indoor environment. In our system, the instructions of the processed route are analyzed to generate a symbolic representation of the navigation task via the instruction interpreter. The resulting symbolic representation is used to generate a topological map of the route to supply the robot with the information about the route's environment and the relationships between the landmarks. It is also supplied to the robot motion planning stage as the initial path estimation of the route description to ground the action and landmark symbols with their equivalent physical procedures and objects, respectively. The route processing module and the experimental results are discussed in detail in Chapter 7.

6.1.2 Vision Processing Module

The vision processing module processes the captured images from the humanoid robot's cameras to detect, recognize, and localize landmarks during navigation. It is based on a two-step classification stage which is robust and invariant towards scaling and translations. Also, it provides a good balance between fast processing time and high detection accuracy. An appearance-based classification method is initially used to provide the rough initial estimate of the landmark. It is followed by a refinement step using a model-based method to estimate an accurate classification of the landmark. The distance estimation between the robot and the processed landmark is calculated by triangulation. The vision processing module and the experimental results are discussed in detail in Chapter 8.

6.1.3 Motion Processing Module

The outputs of the last two modules are supplied to the motion processing module to calculate and execute the shortest humanoid robot footstep placements. The proposed motion planner is a combination of sampling-based planner and D* Lite search to generate dynamic footstep placements in an unknown environment. It generates the search space depending on non-uniform sampling of the free configuration space to direct the computational resources to troubled and difficult regions, such as turns and narrow passages. A modified cylinder model is used to approximate the trajectory for the robot's body-center during navigation. It calculates the actual distances required to execute different motion actions of the robot and compare them to the distances from the nearest obstacles. D* Lite search is then implemented to find dynamic and low-cost footstep placements within the resulting configuration space. The proposed hybrid algorithm reduces the searching time and produces a smoother path for the humanoid robot with low cost. The motion processing module and the experimental results are discussed in detail in Chapter 9.

6.2 Experimental Platform (HOAP-2)

We implemented our navigation system on the second generation of Fujitsu's Humanoid for Open Architecture Platform (HOAP-2) [41]. HOAP-2 is a 7

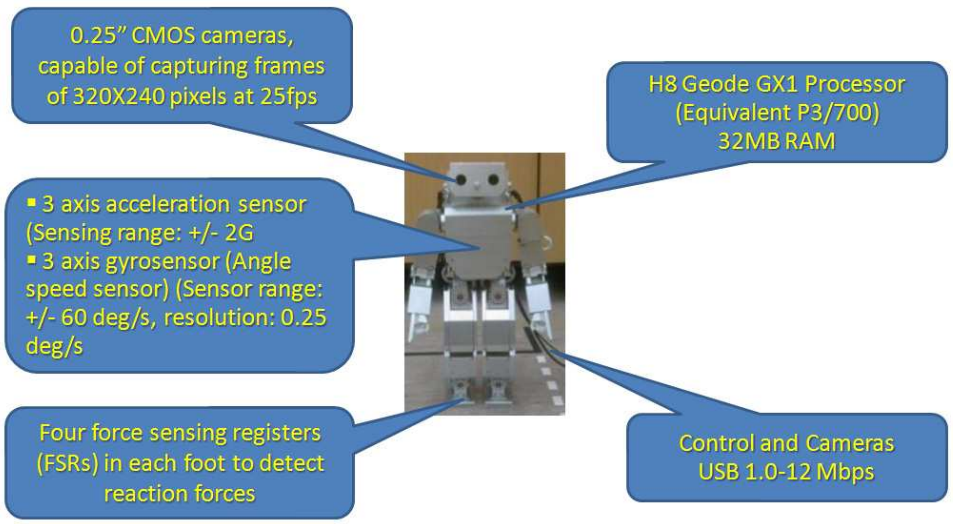

0.25" CMOS cameras, capable of capturing frames of 320X240 pixels at 25fps
H8 Geode GX1 Processor (Equivalent P3/700) 32MB RAM
▪ 3 axis acceleration sensor (Sensing range: +/- 2G
▪ 3 axis gyrosensor (Angle speed sensor) (Sensor range: +/- 60 deg/s, resolution: 0.25 deg/s
Four force sensing registers (FSRs) in each foot to detect reaction forces
Control and Cameras USB 1.0-12 Mbps

Figure 6.2: The experimental platform (HOAP-2).

kg humanoid robot with 50 cm height, 25 cm width, and 16 cm depth (see Figure 6.2). It is equipped with an accelerometer and gyroscope inside the torso. It also has four force sensing registers (FSRs) in each foot to detect reaction forces from the floor. The vision system consists of two 0.25" CMOS unsynchronized cameras (Logitech web quick-cameras) and is capable of capturing frames of 320 X 240 pixels at 25 fps. These cameras need an initial time between 10 to 15 seconds to focus and remove the blurring effect. The robot's cameras are connected via USB 1.0 connections to a laptop with 1.73 GHz Duo processor and 3 GB RAM to process the captured images.

HOAP-2 is equipped with 25 servo actuators (25 DOFs): six for each leg, four for each arm, one for each hand, two for the head, and one for its waist. Figure 6.3 illustrates the joints' names and positions on the HOAP-2 humanoid robot.

Concern the robot motion, if the ZMP walking pattern is used on the HOAP-2 humanoid robot, the robot does not fall over or trip itself up as long as it remains on a smooth, horizontal surface. On the other hand, there was a problem that caused the robot to wobble significantly and its feet to slip each time it put its right foot down. This was due to the fact that there are about 1 to 2 degrees of play in each servo. As the right foot was lowered to the ground after swinging forward, the play in the lateral and sagittal hip joints meant that the left back corner of the foot touched the ground before the rest of the foot and caused the robot to wobble significantly on its left foot, sometimes enough for the robot to fall over. The error in the swing foot is compensated by increasing the distance from the hip to the bottom of the support foot by 0.5 cm [63]. This meant that the swing foot did not

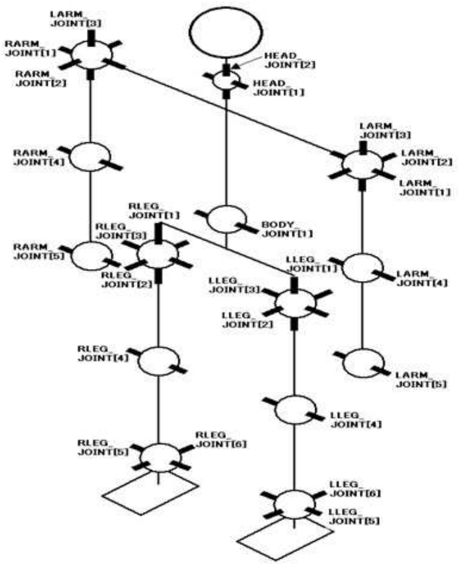


Figure 6.3: Names and positions of the joints on HOAP-2 [41].

touch the ground until it was supposed to. Furthermore, the adjustability of the walking cycle proved to be successful as well. The stride length can be adjusted to be anywhere from 0-6 cm without causing the robot to wobble or to show signs of instability.

On the other hand, HOAP-2 can operate in both wired and wireless modes. In the wired mode, high-speed, real-time communication is possible using an USB interface between the command PC and the robot body. An external 24 V power supply is used for the robot body. By using RT-Linux for the command PC, real-time feedback control is realized using the USB connection. In the wireless mode, a CPU unit inside the robot's body is used as a control host computer. Transmission of motion commands is done from the user's PC via wireless LAN. The internal battery is used as a power supply for the robot.

As shown in Figure 6.4, the software on the command PC of the HOAP-2 humanoid robot is mainly divided into two parts. The first is the real-time robot communication module which is carried out in real time kernel space. The second is the program which loads indicated data on the robot and is carried out in the user space. A data loading indication program and a real-time robot communication module transmit and receive data and commands by using the common memory. The data loading program writes data and commands in the common memory to acquire data from the file and the standard input. A real-time robot communication module, which is written in the common memory, transmits and writes the result received from the robot in the common memory. A data display program indicates the result

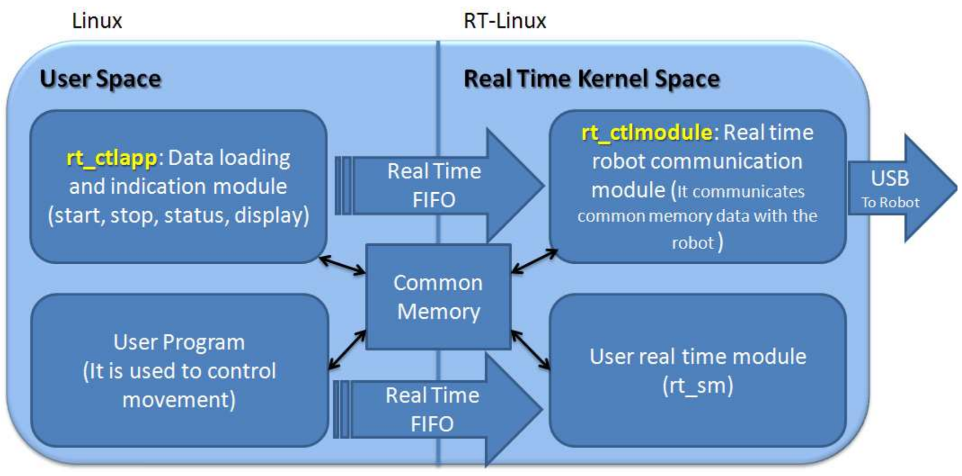

Linux
RT-Linux
User Space
Real Time Kernel Space
rt_ctlapp: Data loading and indication module (start, stop, status, display)
Real Time FIFO
rt_ctlmodule: Real time robot communication module (It communicates common memory data with the robot)
USB To Robot
Common Memory
User Program (It is used to control movement)
Real_Time FIFO
User real time module (rt_sm)

Figure 6.4: The software composition on the command PC of HOAP-2.

written in the common memory. Therefore, the user's program can operate and control the robot's actuators if the user creates a program that is reading and writing from the common memory.

6.3 Experimental Environment

We built a miniature city to be used as an experimental environment for our humanoid robot navigation system. The advantage of using a miniature city as a testing environment is the ability to build a complex route structure in the limited space of a laboratory. We implemented a miniature city to test our system and evaluate its performance. This city is a simulation of the downtown of Hamburg. The design is as realistic as possible to enable the users to apply natural routes for the outdoor real-size environment. The miniature city is built on a 5.0 m x 3.5 m area which is suitable to the dimensions of the HOAP-2 humanoid robot. Figure 6.5 shows the physical realization of our experimental environment.

Some buildings in the miniature city have unique signs taken from real life to indicate supermarkets, stores, or restaurants; such as the Lidl supermarket, the Karstadt store, and the Burger King Restaurant, respectively. These signs are used to recognize these landmarks during the robot's navigation in the city. The other buildings have unique features which can be easily noticed from the other surroundings. These features include the style and color of the building. The railway stations, the town hall, and the church are examples of this type. The boundaries of streets are presented as black straight lines on the ground, whereas the crossroads are indicated as white lines.

We used the Google SketchUp program [106] to created a 3D model for

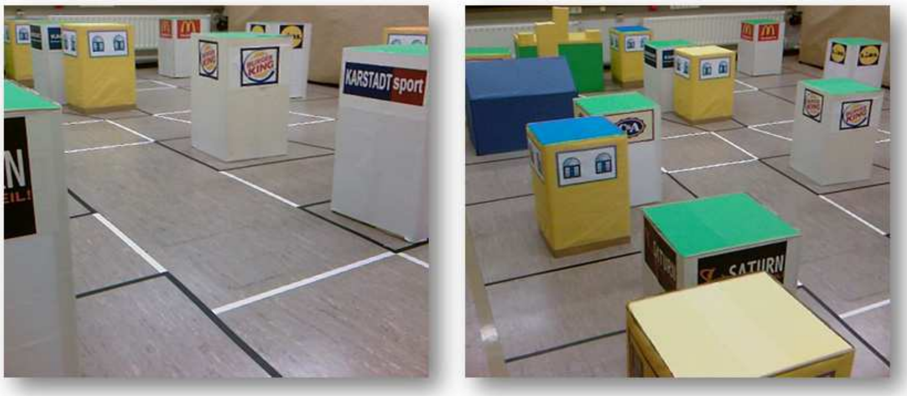


Figure 6.5: The experimental environment.

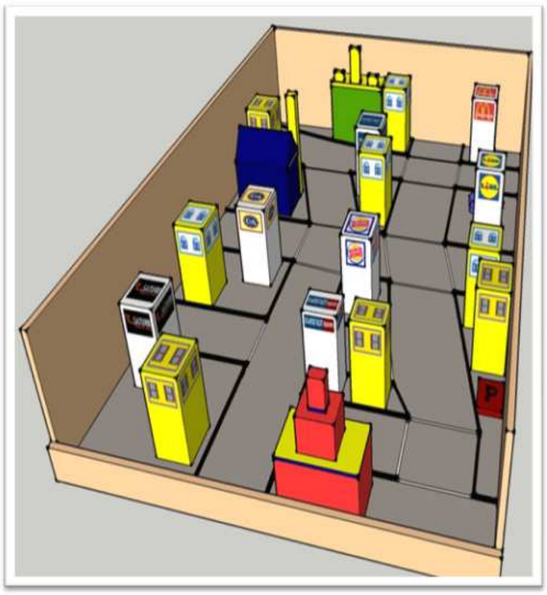


Figure 6.6: The 3D model of the miniature city.

the miniature city. This model is used to test our proposed motion planning algorithm and compare it with some state-of-the-art sampling-based algorithms. Figure 6.6 shows the 3D model of the experimental environment.

6.4 Discussion

In this chapter, the main building blocks of our humanoid robot navigation system are discussed. The system is based on a multimodal interface which is used to describe the route verbally or graphically. The hardware and software characteristics of the experimental platform are discussed. Finally, the experimental environment of HRNS is presented.

CHAPTER 7
A Cognitively Motivated Route–Interface

We present both spatial language and graphical representation to describe route-based navigation tasks for a mobile robot. The instructions of this spatial language are implemented to provide an intuitive interface with which novice users can easily and naturally describe a navigation task to a mobile robot in a miniature city or in any other indoor environment. The instructions of the processed route are analyzed to generate a symbolic representation via the instruction interpreter. The resulting symbolic representation is supplied to the robot motion planning stage as an initial path estimation of route description and it is also used to generate a topological map of the route's environment.

In this chapter, the route processing module is discussed in detail. First, the multimodal route description interface is presented. The structures of the proposed spatial language and the graphical route representation are introduced. The instruction interpreter and the lexicon structure are illustrated in Section 7.2. Afterwards, in Section 7.3, the generation of the topological map of the processed route is described. Finally, the results of the conducted route experiments are discussed.

7.1 Multimodal Route Instructions

At the beginning, the user, who is familiar with the environment, produces a route description for the humanoid robot to execute a navigation task. The route description should contain two prominent types of information. On the one hand, it should contain information about landmarks and decision points. These decision points represent positions on the route on which the robot can choose between different tracks and they are mostly characterized with their relation to landmarks. On the other hand, it should include information about actions the robot has to perform. Therefore, route instructions specify actions, paths, tracks, positions and landmarks in relation to each other [153]. Different groups of words characterize these components. For example, the actions mentioned in route instructions are specifically described with verbs of position, verbs of locomotion, and verbs of change of orientation.

In our system, the route description can be presented in two different ways. It can be represented in written form by using our proposed semi-formal language or by using a simple graphical interface to indicate the path between the start and end points for the robot. These two methods of route descriptions will be discussed in the ensuing subsections.

7.1.1 Verbal Route Description

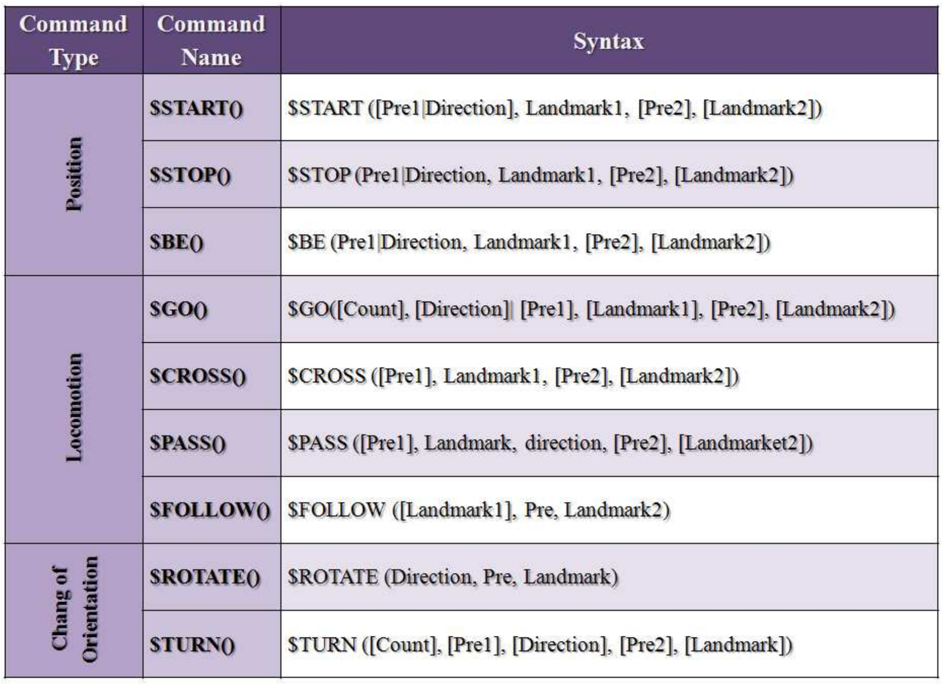
In our system, we present a spatial language – called Route Instruction Language (RIL) – to describe route-based navigation tasks for a mobile robot. This language is implemented to present an intuitive interface that will enable novice users to easily and naturally describe a route to a mobile robot in indoor and miniature city environments. We proposed this language to avoid ambiguity and misunderstanding during route description. Therefore, a non-expert user can describe the route for the mobile robot by using simple and easy to understand instructions.

The RIL is developed to describe the route between the start and end points to a mobile robot. It is intended as a semi-formal language for instructing robots, to be used via a structured graphical user interface. RIL provides elementary instruction statements which are processed to supply the robot with a sequence of motion actions. During navigation, this sequence of actions is processed by the motion planner to determine the footstep placements which will be effected by the humanoid robot to execute the route. Each statement in the RIL constitutes a spatial instruction which relates verbally coded motion concepts to one or more landmarks by use of a suitable spatial relationship.

The commands of the RIL and their syntaxes are shown in Table 7.1. Each instruction of the RIL specifies motion verbs, directions, destinations, and landmarks. The RIL commands are divided into three basic types: position, locomotion, and change of orientation instructions.

The position commands are used to indicate the current position of the robot during navigation. These instructions are primarily used to identify the start and end positions of the robot. They can also be used during the robot route description to describe relevant confirmations of the robot's current position with respect to one or more landmarks. These instructions are represented in RIL by using three different commands: $START(), $STOP(), and $BE().

The Locomotion commands are used to instruct the robot to move in the spatial environment in a specific direction or to follow a certain path.



Command Type	Command Name	Syntax
Position	$START()	$START ([Pre1|Direction], Landmark1, [Pre2], [Landmark2])
	$STOP()	$STOP (Pre1|Direction, Landmark1, [Pre2], [Landmark2])
	$BE()	$BE (Pre1|Direction, Landmark1, [Pre2], [Landmark2])
Locomotion	$GO()	$GO([Count], [Direction]| [Pre1], [Landmark1], [Pre2], [Landmark2])
	$CROSS()	$CROSS ([Pre1], Landmark1, [Pre2], [Landmark2])
	$PASS()	$PASS ([Pre1], Landmark, direction, [Pre2], [Landmarket2])
	$FOLLOW()	$FOLLOW ([Landmark1], Pre, Landmark2)
Chang of Orientation	$ROTATE()	$ROTATE (Direction, Pre, Landmark)
	$TURN()	$TURN ([Count], [Pre1], [Direction], [Pre2], [Landmark])

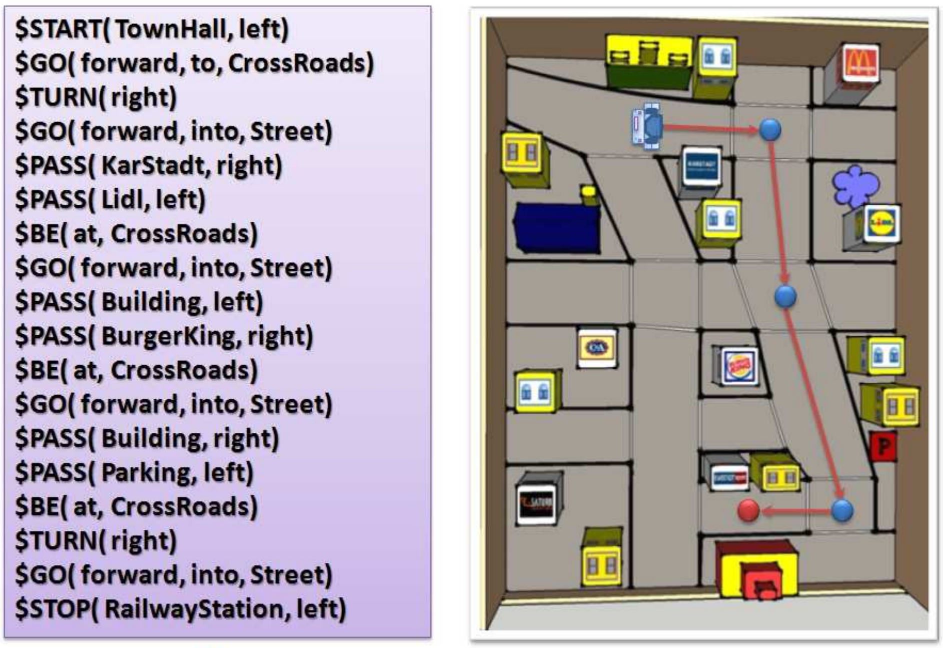
Table 7.1: The command set of the Route Instruction Language (RIL).

In other words, these instructions give the robot the order to move to a particular region or to go in a particular direction with respect to one or more landmarks. In RIL, the locomotion commands are introduced by using four basic instructions: $GO(), $CROSS(), $PASS(), and $FOLLOW().

The last category is the change of orientation commands, which are used to rotate around a landmark or turn in a certain direction. These commands are used to change the direction of the robot by turning or rotating to a specific direction. $TURN() and $ROTATE() commands are used in RIL to represent the changes in orientation of the robot's current position during navigation.

Table 7.1 also shows the syntaxes of the RIL instructions. The instruction's syntax consists of a command word and an arbitrary number of arguments. The command word indicates the action which will be taken by the mobile robot and is represented in the imperative form of the verb, e.g., GO, TURN, BE, etc. Each argument is a place holder for a specific group of words such as prepositions, directions, the number of turns, and landmarks. To add more flexibility to the command syntax, multiple kinds of command syntaxes have been defined. Mandatory arguments are typed without any brackets, whereas optional arguments are placed between rectangular brackets "[ ]". The pipe symbol "|" indicates an OR operator.

For example, the $GO() command can be represented by the following



$START(TownHall, left)
$GO(forward, to, CrossRoads)
$TURN(right)
$GO(forward, into, Street)
$PASS(KarStadt, right)
$PASS(Lidl, left)
$BE(at, CrossRoads)
$GO(forward, into, Street)
$PASS(Building, left)
$PASS(BurgerKing, right)
$BE(at, CrossRoads)
$GO(forward, into, Street)
$PASS(Building, right)
$PASS(Parking, left)
$BE(at, CrossRoads)
$TURN(right)
$GO(forward, into, Street)
$STOP(RailwayStation, left)

(a) Route description (b) Miniature city map

Figure 7.1: "Town hall – railway station" route description by using RIL.

syntax:

$GO([Count],[Pre1|Direction],[Landmark1],[Pre2],[Landmark2])

where "*Count*" presents the number of turns in a specific direction. For example, ***$GO(2, RIGHT)*** means that the robot will take the second right turn. "*Pre*" represents the formal counterpart to a preposition or an adverb which will be used in the spatial statement, such as to, between, and at. "*Direction*" specifies the direction of the turn or a landmark, i.e. left, right, forward, and backward. Finally, "*Landmark*" represents the name of a pre-defined landmark in the knowledge database.

To prevent ambiguity and misunderstanding, RIL uses an extrinsic reference frame to refer to all landmarks and actions in route description which is based on the robot's viewing perspective. It also prevents transformation between different reference types while the route is processed. Figure 7.1 shows an example of a route description from the town hall to the railway station in our miniature city using RIL.

In this route, the robot is instructed to begin at a starting point with the town hall to its left. First, the robot has to move to the crossroads, and then it should turn right. Afterwards the robot is instructed to walk straight on, to pass the Karstadt store on its right, to pass the Lidl supermarket on its left, and to move to the next crossroads. Then it is to walk straight on, pass a building to the left, and the Burger King restaurant to the right, and go on until reaching the next crossroads. The next instructions include the robot's crossing of the street, going straight on, passing a building to the right, passing a parking place to the left, and then turning right at the next crossroads. Finally, the robot has to keep walking down the street until it is standing to the left of the railway station, which is determined as its destination.

7.1.2 Graphical Route Description

We proposed another way to describe the route for the robot by using a simple graphical representation. The main aim of this interface is to provide the user with a simple tool that attempts to discretize the route environment as a graph-like representation of places connected by paths. It represents the environment as a graph where nodes represent places and edges represent connections between places.

On the other hand, the graphical route representation helps to facilitate reasoning of the route, mainly due to the compactness of the representation. It represents the route in a symbolic nature that allows higher-level reasoning (such as order, connectivity, and regions) which can enhance the spatial knowledge of the route environment.

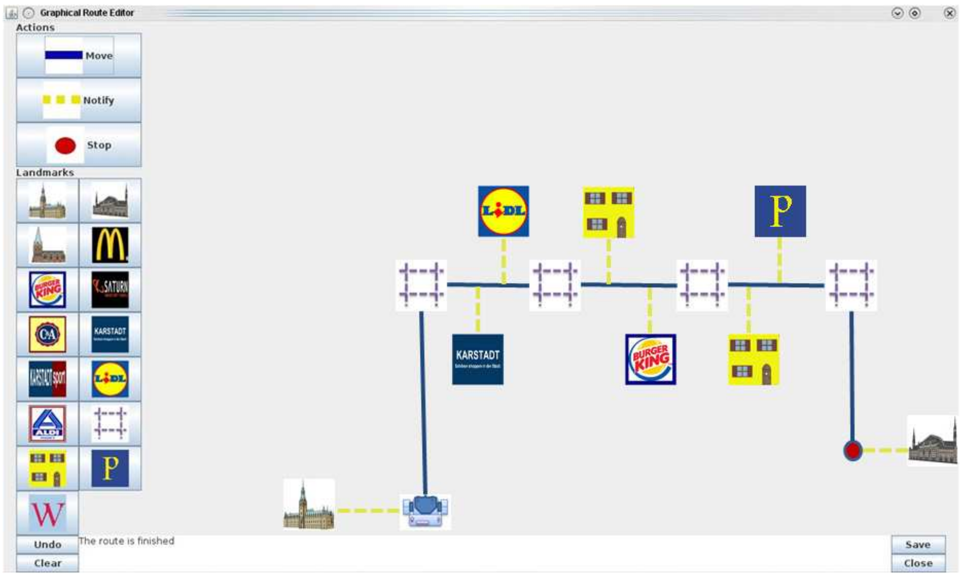
In our implementation, we proposed three main actions of the robot: move, notify, and stop. The move action is used to draw the estimated path for the mobile robot. It is presented as solid straight lines. Robot turns are indicated as straight lines in different directions. The notify action presents the current robot's position with respect to one or more landmarks. It also indicates the position of landmarks during robot motion in a specific direction. It is represented as dashed line which connects landmarks with the robot's path. Finally, the stop action is used to indicate the target point of the robot with respect to one or more landmarks. Table 7.2 lists the actions and landmarks which are used in the graphical representation.

In the graphical representation, the robot's position is shown at the bottom of the map as an origin of the route. The user begins to draw the map by starting from the robot's position until he reaches the target point. At the beginning, the user localizes the current robot's position by drawing one or more landmarks connected to the robot's image in a specific direction. Then, he starts to draw the robot's path by using motion action and identifies the path by choosing some landmarks. Finally, he uses the stop action to finish the robot's route and indicates the target point. Figure 7.2 shows a graphical representation of the same route presented in the previous section ("Town hall – railway station" route) by using the route map editor.

For map reasoning, during the map drawing, the user's actions are analyzed and converted to a symbolic representation. The initial robot's posi-

Name	Symbol	Name	Symbol
Move Action	▬	Notify Action	■ ■ ■
Stop Action	●	Town Hall	
Railway Station		Church	
McDonald's	M	Burger King	BURGER KING
Saturn Store	SATURN GEIZ IST GEIL	C&A Store	C&A
KarStadt store	KARSTADT Schöner shoppen in der Stadt	Sport KarStadt store	KARSTADT sport
Lidl Supermarket	LIDL	Aldi Supermarket	ALDI MARKT
Crossroads		Building	
Parking Place	P	Water pool	W

Table 7.2: The used symbols and landmarks in the graphical representation.



The route is finished

Figure 7.2: The graphical representation of the "town hall – railway station" route.

tion is considered as the start point of the route with respect to one or more landmarks. The stop position is considered as the robot's target. The move actions are translated into motion and turn actions. For more details about the symbolic representation, see Section 7.2.4.

7.2 Instruction Interpreter

The instruction interpreter is used to discriminate, identify, and categorize the motion actions of the processed route description. It converts the verbal description of the route produced by using RIL into a sequence of actions that the robot must take in order to successfully follow paths anticipated by the instruction giver. It combines definitions from the lexicon according to the parse structure of the instruction, creating a symbolic script that describes the navigation process. The generated symbolic representation is used to create a topological map for the route environment. It is also supplied to the motion planner as an initial path estimation of the navigation task to help in generating the footstep placements for the humanoid robot.

The instruction interpreter contains a simple parser, a lexicon, a syntactic analysis, and a symbolic generator (see Figure 7.3). The parser is supplied by the route description text to split it into a sequence of words. The resulting list is entered at the syntactical analysis stage to identify the structure of instructions by consulting in the lexicon to obtain the type and features of the resulting words. Finally, the result will be supplied to the

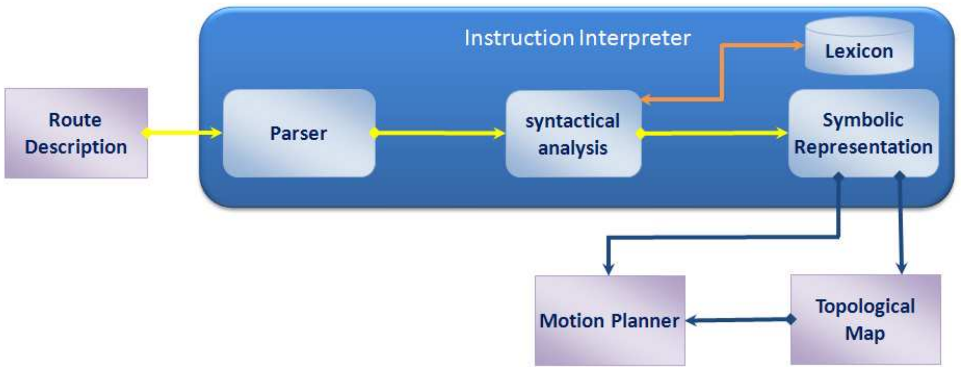


Figure 7.3: The structure of the instruction interpreter.

symbolic representation stage to generate an equivalent symbolic script of the route. The components of the instruction interpreter will be elucidated in the next subsections.

7.2.1 The parser

The parser analyzes the raw route description text supplied by the user and prepares it for the syntactic analysis stage. The parser separates the text into individual instructions. Each instruction is split into a sequence of words using space and punctuation characters as delimiters. The resulting list is entered at the syntactical analysis stage to identify the structure of instructions.

7.2.2 The Syntactical Analysis

The syntactical analysis stage is provided by a list of the resulting words from the parser to identify the structure of the instructions. It compares the structure of the processed instruction with a list of all kinds of instruction syntaxes which are understandable by the robot. Each word is looked up in the lexicon to obtain its type and features.

The syntactical analysis is robust to unexpected input. If it encounters a constituent that cannot be modeled, it will ignore it while modeling the remainder of the instruction. Likewise, if it cannot analyze one instruction from a set of route descriptions, it will analyze the others. Additionally, the syntactical analysis is used to divide the route description into segments. Each segment begins with a motion action (i.e. $GO() instruction) and ends before the next one – except for the starting and ending statements. All statements in the segment are processed in series and executed as a single sub-route in the robot navigation task.

Type of expression	Symbolic representation
Verbs of position	**BE_AT(x, p)**
Verbs of motion	**GO(x, w)**
Verbs of change of orientation	**CH_ORIENT(x, d)**
Landmark notification	**VIEW(x, p)**
Local preposition or adverb	**LOC(p, Pre(LM))**
Directional preposition or adverb	**TO(w, Pre(LM))** **FROM(w, Pre(LM))** **VIA(w, Pre(LM))** **LOC(w, Pre(LM))**
Projective terms	**Pre(LM, rsys)**

Table 7.3: Descriptive operators used in symbolic representation.

7.2.3 The Lexicon

The lexicon is a module of linguistic knowledge that maps words onto structures representing their meaning. It combines syntactic and semantic information about the words such that the syntactic structure can support the derivation of the meaning of the instructions. Thus, the task to construct the meaning of a route instruction presupposes a coherent and consistent system of entries in the spatial lexicon.

We have defined a lexicon of words in terms of RIL, and used that lexicon to analyze the route description. The lexicon is used to find the type of the parsed words and also combines these words with their definitions. The available types of words in the lexicon are command verbs, directions, prepositions, numbers of turns, and landmarks. Each verb entry in the lexicon consists of an action verb and an associated script composed from the set of its primitives and depends on the specified arguments passed to its instruction. It is defined as a script of primitive operations that run on data extracted from the analyzed instruction. In other words, each entry in the lexicon consists of a symbol and an associated script composed from the set of primitives. Depending on the processed instruction, the lexicon picks the suitable subscript for the processed word. Some spatial routines in the lexicon return a subscript as their result.

7.2.4 The Symbolic Representation

After analyzing the route instructions syntactically and connecting each resulting verb with its motion procedure, the symbolic representation of the route is generated. This symbolic script is based on CRIL representation which was developed by Tschander et al. [153]. Table 7.3 lists some descriptive operators used in the symbolic representation.

The resulting symbolic script consists of three basic components: motion actions, spatial relationships, and landmarks. The motion actions are classified into the following four essential actions:

!BE_AT() Action: It presents the position of the robot during navigation. It identifies the start, current, and end positions of the robot during navigation.

!GO() Action: It indicates the motion actions which should be taken by the mobile robot.

!VIEW() Action: It is used to notice a landmark in a certain direction or region during navigation.

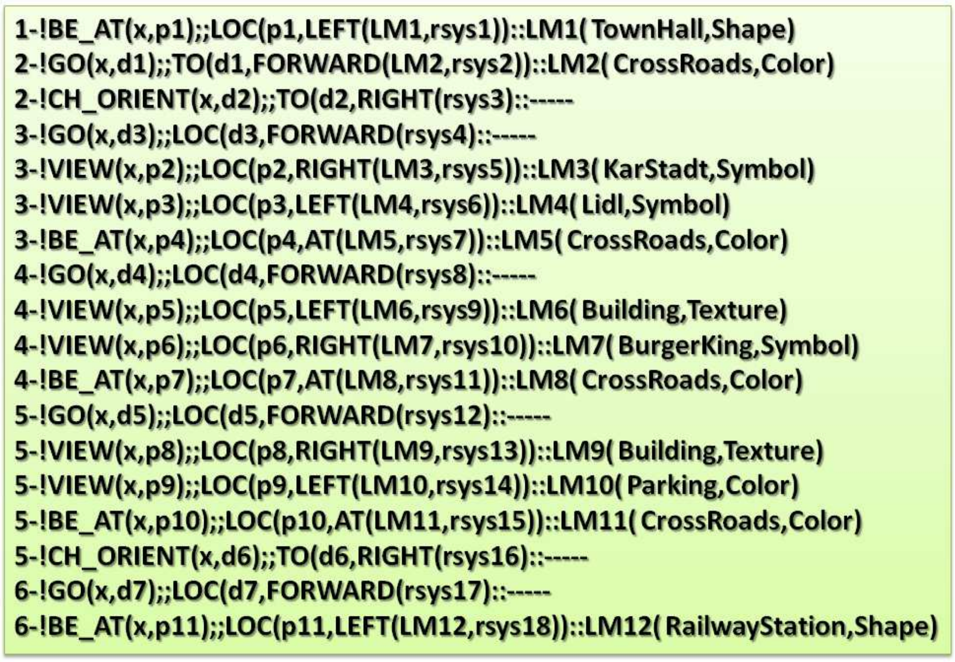
!CH_ORIENT() Action: It is used to indicate a change in the current orientation of the mobile robot motion during navigation based on a specific direction or landmark.

The spatial relationships are classified into two types. First, relations represent a location with respect to a landmark. Second, relations specify a direction with respect to one or two landmarks. The spatial or directional relationships can be divided into four classes. The first class is the goal relationships which specify the end of the path. The second one is the source relationships which give the start of the path, the third class is the course relationships which characterize the intermediate course of the path, and the final class is the shape relationships which identify the shape of the path.

Landmarks in our miniature city are classified into definite and indefinite landmarks depending on their features. Definite landmarks have unique characteristics which single them out from among the other landmarks in the miniature city, such as the Burger king restaurant, the Saturn store, and the town hall. On the other hand, indefinite landmarks have a number of properties that are not unique such as buildings, crossroads, and streets. The landmarks are represented in the symbolic representation by using the following syntax:

$$LM_i(\textbf{Name, type})$$

where i presents the landmark number in the route. The "**Name**" argument represents the landmark name. The "**type**" argument presents the way that is used to recognize the landmark. The landmark features are retrieved from the database which contains data about their shape, color or color histogram, and recognition method values. In addition to the retrieved features, the relationship feature is extracted from the processed route to describe the relation between the current processed landmark and other landmarks in the same path segment. It is used to handle uncertainty and missing information during the robot navigation.



```
1-!BE_AT(x,p1);;LOC(p1,LEFT(LM1,rsys1))::LM1( TownHall,Shape)
2-!GO(x,d1);;TO(d1,FORWARD(LM2,rsys2))::LM2( CrossRoads,Color)
2-!CH_ORIENT(x,d2);;TO(d2,RIGHT(rsys3)::-----
3-!GO(x,d3);;LOC(d3,FORWARD(rsys4)::-----
3-!VIEW(x,p2);;LOC(p2,RIGHT(LM3,rsys5))::LM3( KarStadt,Symbol)
3-!VIEW(x,p3);;LOC(p3,LEFT(LM4,rsys6))::LM4( Lidl,Symbol)
3-!BE_AT(x,p4);;LOC(p4,AT(LM5,rsys7))::LM5( CrossRoads,Color)
4-!GO(x,d4);;LOC(d4,FORWARD(rsys8)::-----
4-!VIEW(x,p5);;LOC(p5,LEFT(LM6,rsys9))::LM6( Building,Texture)
4-!VIEW(x,p6);;LOC(p6,RIGHT(LM7,rsys10))::LM7( BurgerKing,Symbol)
4-!BE_AT(x,p7);;LOC(p7,AT(LM8,rsys11))::LM8( CrossRoads,Color)
5-!GO(x,d5);;LOC(d5,FORWARD(rsys12)::-----
5-!VIEW(x,p8);;LOC(p8,RIGHT(LM9,rsys13))::LM9( Building,Texture)
5-!VIEW(x,p9);;LOC(p9,LEFT(LM10,rsys14))::LM10( Parking,Color)
5-!BE_AT(x,p10);;LOC(p10,AT(LM11,rsys15))::LM11( CrossRoads,Color)
5-!CH_ORIENT(x,d6);;TO(d6,RIGHT(rsys16)::-----
6-!GO(x,d7);;LOC(d7,FORWARD(rsys17)::-----
6-!BE_AT(x,p11);;LOC(p11,LEFT(LM12,rsys18))::LM12( RailwayStation,Shape)
```

Figure 7.4: The resulting symbolic representation of the "town hall – railway station" route description.

Figure 7.4 shows the resulting symbolic representation of the route described in Figure 7.1. It displays the three types of information extracted from the route description: actions, spatial relationships, and landmarks. Where "**rsys**" refers to a spatial reference system that has to be anchored relative to the conceptual representation of the preceding segments of the route instruction. The "**x**", "**LM**", "**w**", "**p**", "**t**", and "**r**" symbols refer to the robot, landmark, path, position, track, and region, respectively. The grounding of the symbolic representation with the perceptual data in the physical environment will be discussed in Chapter 9.

7.3 Topological Map

After creating the symbolic representation of the route, the robot requires an adequate representation of the route environment. This representation should be abstract enough to facilitate higher-level reasoning tasks like strategic planning or situation assessment, and still be detailed enough to allow the robot to perform lower-level tasks like path planning/navigation or self-localization. The topological map is used to describe relationships among features of the environment in a more abstract representation without any absolute reference system. Our implementation of the topological map represents the robot's workspace in a qualitative description. It presents a graph-like description of the route where nodes correspond to significant, easy-to-distinguish landmarks, and arrows correspond to actions or action sequences which will be executed by the mobile robot.

The generated topological representation of the route is mainly used to treat the ambiguity which can be occur in situations where a landmark can be described not only by its properties, like color and shape, but also by its relations to other landmarks. By considering relations, we may be able to resolve cases where the known properties of the landmark are not sufficient to distinguish it from other similar landmarks, or the robot cannot recognize the current landmark. Therefore, the topological map is used to comprehend the route instructions by building up a mental representation of the processed route. This representation contains spatial information about the route, the sequence of actions to be performed, and the relations between landmarks. In the navigation phase, the robot has to match the internal representation (i.e. topological map) against the perceived scenes. Figure 7.5 shows the generated topological map of the "town hall – railway station" route which is represented in Figure 7.1.

In the topological map representation, the blue arrows represent the estimated robot path, whereas the yellow dashed lines represent the positions of the landmarks. The rounded rectangles represent the processed landmarks and their colors indicate the type of landmarks. Finally, orange circles indicate the start and destination points, whereas green circles represent intermediate nodes in the robot's path.

7.4 Experimental Results and Evaluation

To evaluate the routes which are written by using the RIL instructions, we conducted two different experiments. The first experiment is carried out to test the suitability of the RIL for the novice users. The second experiment is conducted to analyze the resulting RIL routes and compare them with their equivalent verbal routes. It also tests the usability of both RIL instructions and the graphical user interface of our system.

7.4.1 RIL for Novice Users

We carried out an experiment to test the suitability of the proposed route instruction language as an interaction tool between a computer-language naive user and a mobile robot. 18 participants took part in the experiment (mean age = 29.5, SD = 3.35). None of the participants had any background knowledge on route instructions and robotics.

The experiment was conducted in single sessions. The participants were first supplied with instructions about the purpose of the experiment, a general idea about RIL, and the estimated time of the experiment. Then, we gave them a description of the RIL syntax, a map of the miniature city, and

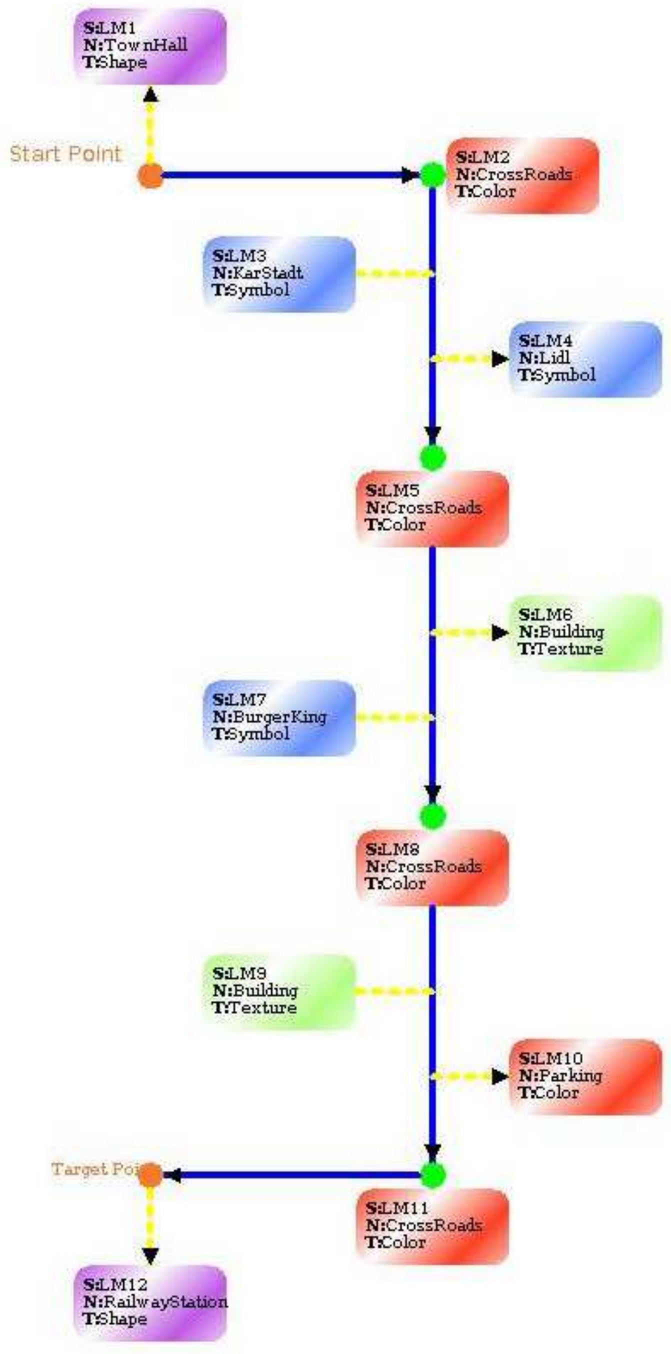
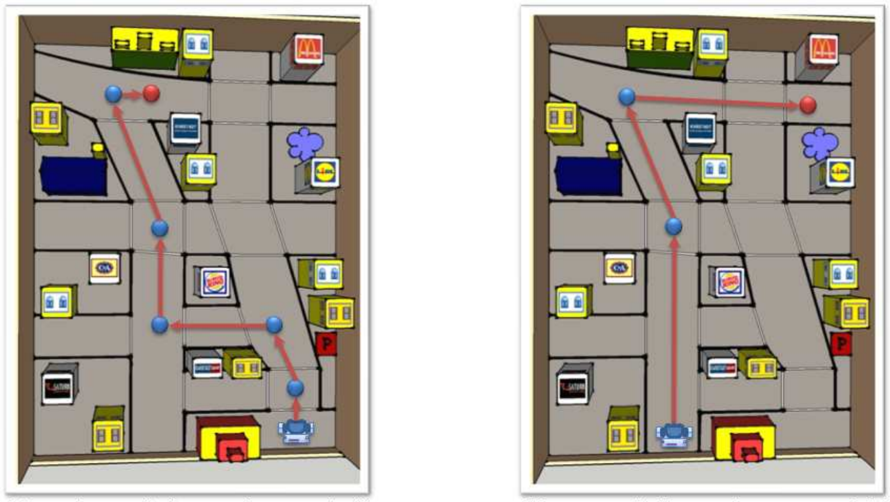


Figure 7.5: The topological map representation of the "town hall – railway station" route.



(a) Sample route "railway station – town hall" (b) Test route "railway station – MacDonald's"

Figure 7.6: The sample and tested routes which were used in the experiment.

an example of a suitable route description. Figure 7.6 shows the example route which represents the route from the railway station to the town hall in the miniature city. Afterwards, we asked them to describe a route between the railway station and the McDonald's restaurant as depicted in the last Figure. Finally, we gave them a questionnaire to know their impression of the usage of the RIL. There were no time limitations in the experiment. The entire session took approximately 15-20 minutes.

After analyzing the tested route and the answered questions for each participant, we found that:

- 89% of the participants described the route correctly, but the rest were confused about how to use some commands and parameters.
- 83% of the participants stated that the RIL is simple and easy to learn, but the rest of them preferred to use a controlled natural language without any specific syntax for the instructions.
- 78% of the participants agreed that it is better to provide the commands of RIL with many optional parameters than to restrict them to a single syntax.

7.4.2 Route Description Analysis

We conducted a second experiment to analyze the route descriptions written by RIL using a GUI. This analysis had two main purposes. The first aim is to dissect the resulting verbal and RIL routes produced by the participants.

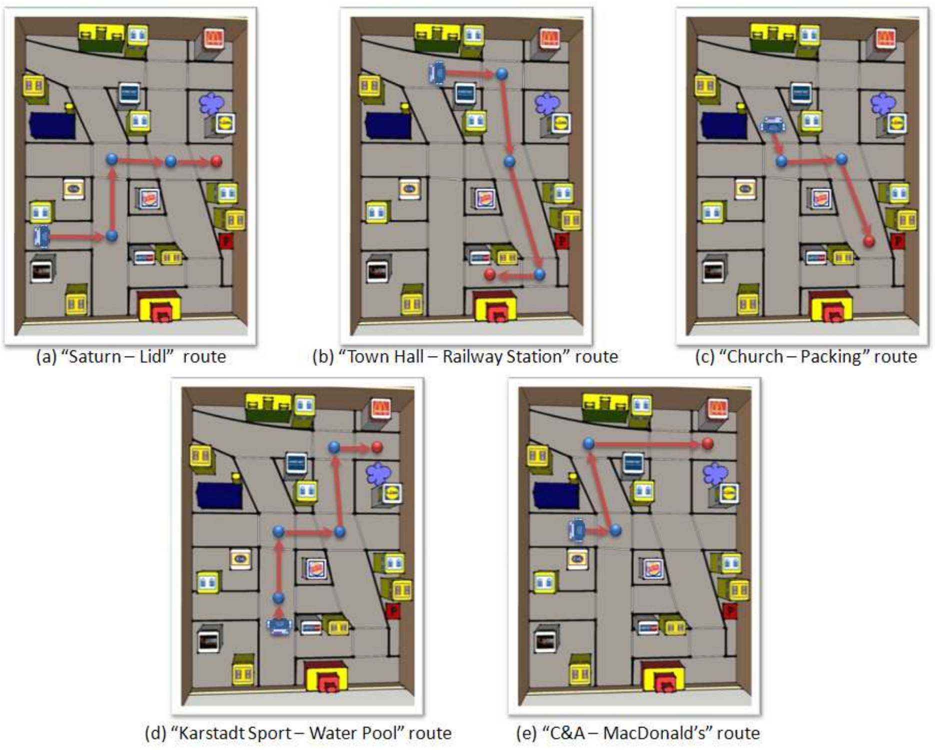

(a) "Saturn – Lidl" route
(b) "Town Hall – Railway Station" route
(c) "Church – Packing" route
(d) "Karstadt Sport – Water Pool" route
(e) "C&A – MacDonald's" route

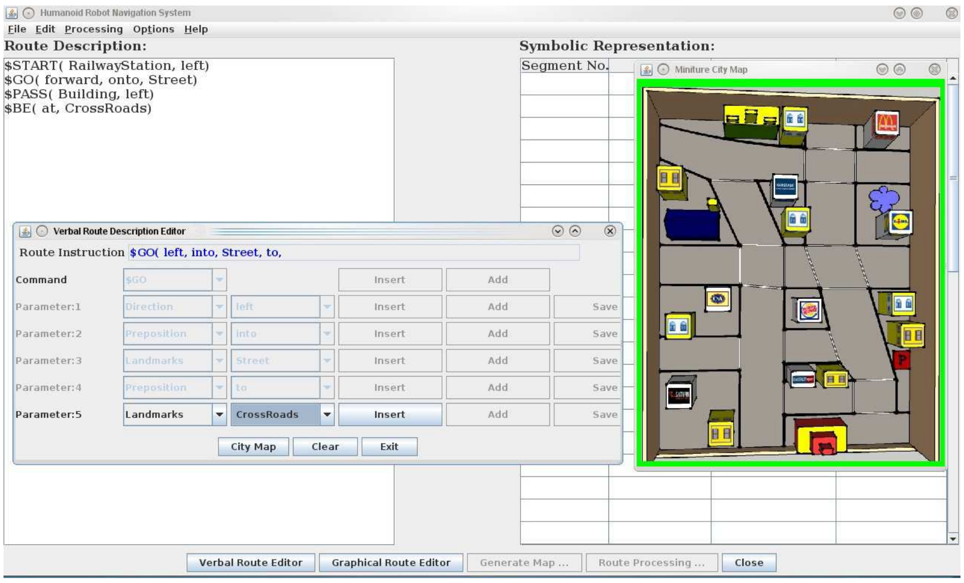
Figure 7.7: The five routes used in the experiment.

The second purpose is to analyze the RIL instructions to know the common instructions used by the users.

15 participants (7 male, 8 female) took part in this study, ranging in age from 23 to 35 (mean age = 31.07, SD = 3.49). The participants had no previous experience with natural language processing and robotics. They were tested individually. Each participant was asked to describe 5 different routes in the miniature city. Figure 7.7 illustrates the routes used in the study.

At the beginning, the participants were asked to describe these routes verbally. After finishing the verbal description of the five routes, we explained the command syntax of the RIL to the participants and how to instruct the robot to travel from one place to another in the miniature city. Then, they were asked to describe these routes again by using the RIL via a GUI (as shown in Figure 7.8). After finishing all route descriptions, the users were asked to answer a simple questionnaire to get their impression of describing the routes by using the GUI.

We analyzed the resulting 75 verbal route descriptions (5 routes/user). We found that 98% of the resulting descriptions can be presented by using RIL. For the RIL routes, we had 830 instructions resulting from the par-



Route Description:

$START(RailwayStation, left)
$GO(forward, onto, Street)
$PASS(Building, left)
$BE(at, CrossRoads)

Route Instruction $GO(left, into, Street, to,

Command	$GO	
Parameter:1	Direction	left
Parameter:2	Preposition	into
Parameter:3	Landmarks	Street
Parameter:4	Preposition	to
Parameter:5	Landmarks	CrossRoads

Figure 7.8: The graphical user interface used to write RIL routes.

Route	Time (min) (SD)	No. of Instructions (SD)	Correct Instructions (SD)
1	7.53 (2.95)	11.07 (2.25)	92% (0.13)
2	6.73 (2.76)	11.93 (1.79)	92% (0.14)
3	4.00 (1.13)	08.87 (1.77)	95% (0.09)
4	5.40 (1.24)	13.20 (2.21)	96% (0.07)
5	4.07 (1.10)	10.27 (1.87)	94% (0.10)

Table 7.4: The statistical analysis of the resulting route descriptions.

ticipants' routes. Table 7.4 shows the average and standard deviation of the time, number of instructions per route, and correct percentage of instructions for the resulting route descriptions. Only 47 routes succeeded in reaching the goal, whereas the other 28 routes contain one or more wrong instructions which cannot be analyzed correctly to produce the equivalent symbolic and topological map representations of the route.

Figure 7.9 shows the frequency of the three instruction categories of RIL in the resulting route descriptions. It is obvious that the locomotion instructions are the most used RIL category. On the other hand, Figure 7.10 illustrates the frequency of the RIL instructions in the resulting participants' routes. It is observed that the "Go" instruction is the most used command and the "FOLLOW" and "ROTATE" instructions are the less used commands in the resulting route descriptions. Consequently, the "GO" instruction clearly dominates among all analyzed instructions and the "for-

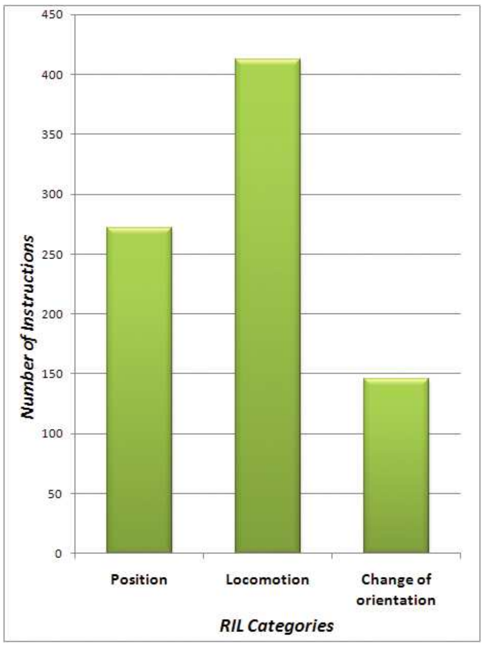


Figure 7.9: Occurrence statistics of RIL categories in the resulting routes.

ward" type that includes instructions such as "GO ahead", "GO down", "GO along" etc., is the most frequent type among other "GO" instructions as shown in Figure 7.11.

After analyzing the answered questions of the given questionnaire, we found that:

- 88% (SD = 0.13) of the participants stated that the GUI is simple and easy to use.
- 93% (SD = 0.11) of the participants declared that the RIL is simple and easy to learn.
- 90% (SD = 0.21) of the participants did not face any ambiguity or misunderstanding during route description.
- 78% (SD = 0.39) of the participants agreed that it is better to provide the commands of RIL with many optional parameters than to restrict them to a single syntax.

At the end, the results of the experiments confirmed that RIL is simple to learn and it is well suited to describe the route in indoor environments. The GUI facilitates the route description and lets the novice user describe the routes easily without ambiguity and misunderstanding. On the other hand, we have found that most of the commands that participants chose can

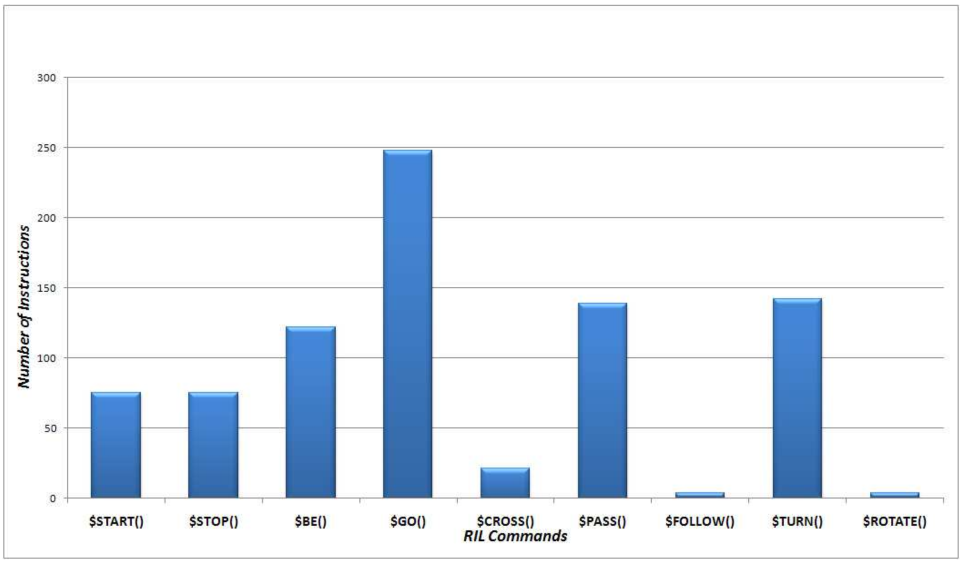


Figure 7.10: Occurrence statistics of RIL instructions in the resulting routes.

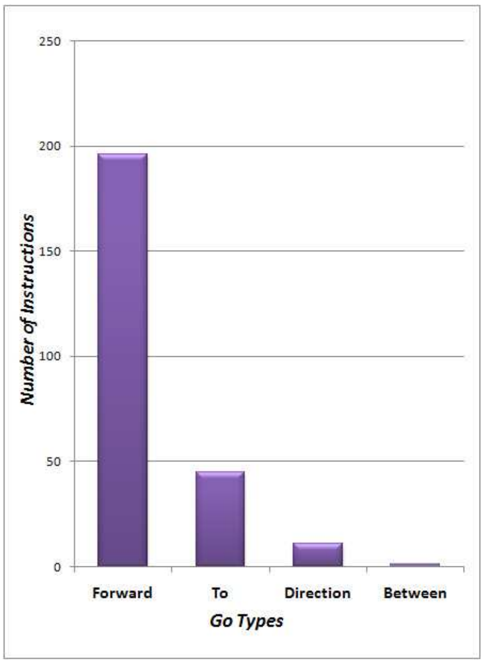


Figure 7.11: Occurrence statistics of "GO" instruction types in the resulting routes.

be classified or decomposed into these categories, and by considering only such commands, we can replicate the paths with reasonable accuracy.

7.5 Discussion

In this chapter, the route processing module of our humanoid robot navigation system is discussed. The system is based on a multimodal interface which is used to describe the route verbally or graphically. The route processing model of our system is presented in detail. We have presented the Route Instruction Language (RIL) – a semi-formal language – to be used by inexpert users to instruct humanoid robots in a miniature city environment. Based on the RIL and graphical representation, we designed and realized an intuitive interface to mobile robots preventing misunderstanding and ambiguities in route descriptions. Starting from a set of commands, the instruction interpreter stage performs the analysis of route instructions and its lexicon relates the internal procedures to perceptual objects and specifies actions that can be carried out by the humanoid robot. The instruction interpreter analyzes the route to generate its equivalent symbolic representation which is supplied to the motion planner as initial path estimation.

The resulting symbolic representation of the route is used to generate a topological representation of the route to supply the robot with global information about the route and to prevent it from getting trapped in local loops or dead-ends in unknown environments. The symbolic representation is supplied to the motion planner to ground the landmark symbols in their equivalent physical objects by using perceptual anchoring.

We conducted some experiments to test and evaluate the performance of the interface of HRNS. We ran two different experiments to test the validity of the proposed route instruction language (RIL). The first experiment was conducted to test the suitability of the RIL for the naive user. The second experiment was run to compare the verbal and RIL route instructions. The results were analyzed to evaluate the performance the RIL and the GUI. The results of the route experiments confirmed that RIL is simple to learn and is well suited to describe the routes in indoor environments. They also illustrated that the GUI provides the novice users with an easy interface for the route description.

CHAPTER 8

Robot Landmark Processing System

In mobile robot scenarios, it is expected that the robot autonomously navigates through home or office environments and processes objects/landmarks during navigation. Landmark processing is identified as one important research area in robot navigation systems. It is a key feature for building robots capable of navigating and performing tasks in human environments. For autonomous navigation, the mobile robot requires a high-speed reliable vision approach that does not introduce significant delays in the control loop and can perform in real time. Therefore, the object detection and recognition approaches, which are suitable to be adapted to mobile robots, should be fast, reliable, and flexible techniques.

In this chapter, we present our Robot Landmark Processing System (RLPS) which is implemented to detect, identify, and localize different types of landmarks during robot navigation in indoor or miniature city environments. The aim of our work is to develop a robust lightweight object processing system with a high detection rate that can actually be used by mobile robots and meet their hard constraints to recognize landmarks during navigation. The system is based on a two-step classification stage which is robust and invariant towards scaling and translations. Also, it provides a good balance between fast processing time and high detection accuracy. An appearance-based classification method is initially used to provide the rough initial estimate of the landmark. It is followed by a refinement step using a model-based method to estimate an accurate classification of the processed landmark. On the other hand, stereo triangulation is calculated to determine the landmark's position in the environment by using the robot's cameras.

8.1 Vision System Architecture

We have developed an online robot landmark processing system (RLPS) running on the HOAP-2 humanoid robot. RLPS is used to detect, classify, and localize different types of landmarks during humanoid robot navigation. The robot autonomously navigates in an indoor environment, moves according to the processed route description, and localizes its position in the environment with respect to the detected landmarks. The robot recognizes predefined landmarks, estimates their position in the environment, and integrates the result with the localization module to automatically process the landmarks and use them in motion planning. Our main goal is to implement a robust, accurate, and real-time landmark processing system for mobile robot navigation which can handle different types of landmarks.

The robot uses the symbolic representation and the generated topological map of the route description to decide which landmark will be processed during navigation (for more details, see Chapter 7). The topological map represents the route description in a graphical representation and it also retrieves the relationships between the landmarks to handle uncertainties during robot navigation. The symbolic representation of the route is grounded to the output of RLPS by using perceptual anchoring. The result is supplied to the path and footstep planners to generate the shortest feasible footstep placements of the humanoid robot (see Chapter 9).

As shown in Figure 8.1, the architecture of RLPS can be divided into three basic stages: stereo calibration, stereo vision and triangulation, and landmark classification. The stereo calibration is used to calculate the internal and external parameters of the left and right cameras of the robot. The stereo vision stage is responsible for creating disparity and depth maps of the captured images. The outputs from the stereo vision combined with the external parameters of the stereo pairs are used to calculate the 3D position of the retrieved landmarks in the real world. Last but not least, the landmark classification stage is responsible for detecting and recognizing the landmarks from the captured frames.

RLPS is effectively based on a two-step classification process. An appearance-based classification method is first used to get a fast and rough estimation of the landmarks. The resulting hypotheses are refined by a model-based classification method to get accurate landmark recognition. The combination between these two methods provides a computational efficiency procedure. The Hough transform, color detection, or color histogram of the detected landmark provides a rough initial estimation of the landmark. It is followed by a refinement step using Scale Invariant Features Transform (SIFT) to get an accurate estimation for the landmark. On the other hand, we used the disparity map combined with recognized landmarks to calculate their position with respect to the robot's position during navigation. The position of the processed landmark is determined by calculating the stereo

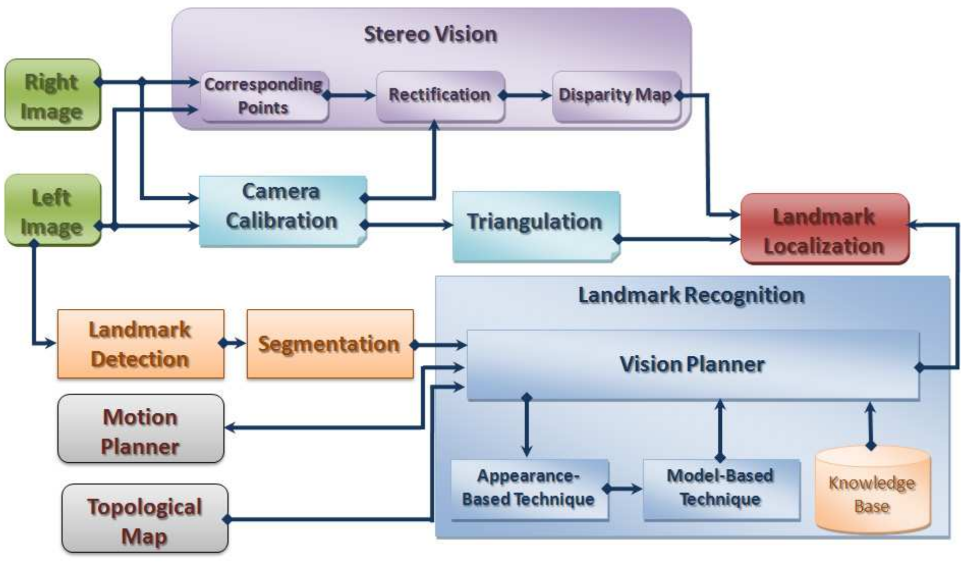


Figure 8.1: The architecture of the robot landmark processing system (RLPS).

triangulation. In the following sections, the main building blocks of RLPS will be discussed in detail.

8.2 Stereo Camera Calibration

Camera calibration is the task of relating the ideal pinhole model of the camera to an actual imaging device and it is also the task of retrieving the relative position and orientation of the cameras. Therefore, it is used to determine the geometry of the stereo setting which is needed for triangulation and also for removing radial and tangential distortions provided by camera lenses. Consequently, the process of the stereo camera calibration can be divided into two main tasks: calculating the intrinsic parameters of each camera to handle the distortions of the lenses and calculating the extrinsic parameters of the stereo pair to calculate the landmarks' positions in the real environment.

Our stereo vision system receives the left and right frames from two CMOS cameras located in the robot's head. By calibrating the cameras, image distortion is removed and cameras' parameters such as the focal length and the principal point are determined. These parameters are useful later for computing 3D range data. We first recover the intrinsic parameters and distortion coefficients offline by using Zhang's camera model [166] during a standard checkerboard-based calibration stage. This method works very well and allows subsequent distance images to be rectified even if the sensors have

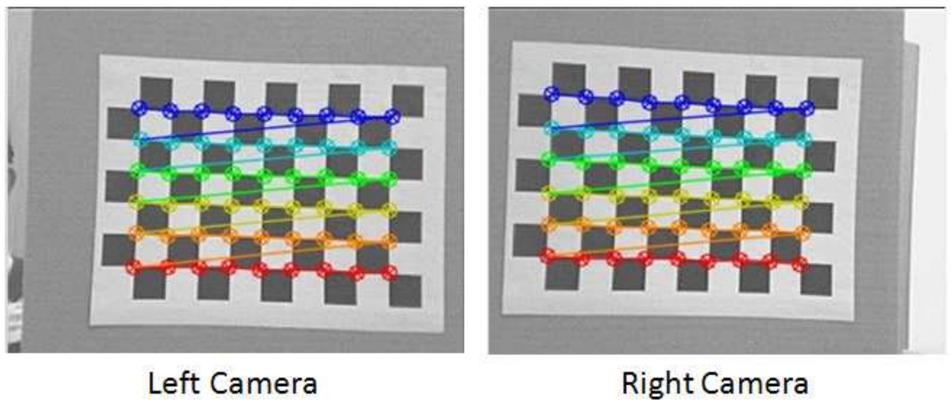


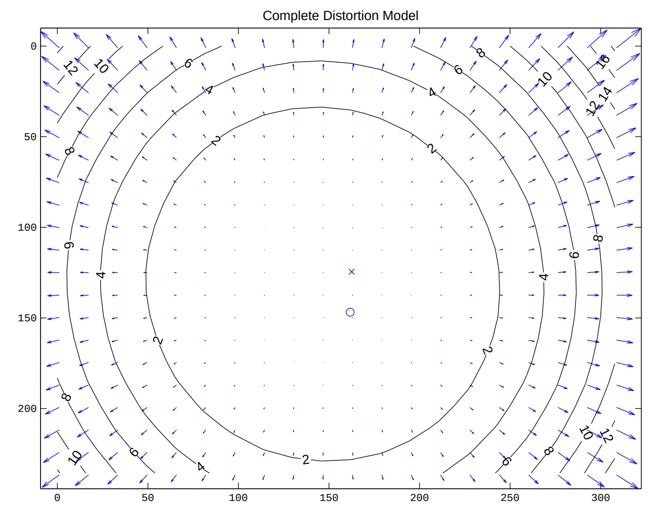
Figure 8.2: The calibration of the left and right robot's cameras by using a chessboard pattern.

comparatively low resolutions. The resulting matrix of the internal camera parameters has the following shape:

$$\mathbf{M} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

f_x and f_y represent the product of the physical focal length of the lens and the size of the individual imager elements in the x and y directions, respectively. c_x and c_y introduce the coordinates of center of radial lens distortion. A regular chessboard pattern is used as a calibration object which is much easier to deal with. Ten different views of the chessboard are used to calculate the cameras' parameters where the corners of the black squares are detected and used as calibration points. Figure 8.2 shows the calibration of the left and right cameras by using a chessboard pattern.

We used Brown's technique [11] to calculate the distortion parameters of the camera's lens. We have two main lens distortions: radial and tangential distortions. The radial distortion arises as a result of the shape of the lens, where the lenses of real cameras often noticeably distort the location of pixels near the edges of the imager. This bulging phenomenon is the source of the "barrel" or "fish-eye" effect. This distortion is small and can be characterized by the first three terms of a Taylor series expansion around $r = 0$ (k_1, k_2, and k_3). On the other hand, the tangential distortion arises from the assembly process of the camera as a whole. It is due to manufacturing defects resulting from the lens not being exactly parallel to the imaging plane. It is minimally characterized by two additional parameters, p_1 and p_2. Therefore, the distortion matrix of the camera can be represented as:



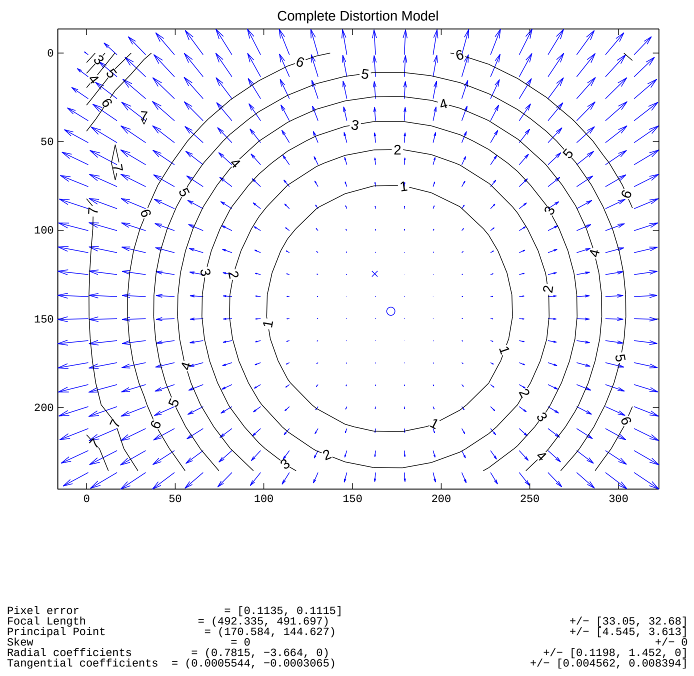
```
Pixel error                      = [0.1027, 0.09916]
Focal Length                = (417.715, 417.898)                                         +/- [27.46, 26.92]
Principal Point              = (160.678, 145.854)                                        +/- [2.783, 4.445]
Skew                             = 0                                                              +/- 0
Radial coefficients         = (0.4178, −0.2363, 0)                                   +/- [0.06689, 0.456, 0]
Tangential coefficients  = (0.01332, 0.01296)                                     +/- [0.004796, 0.005334]
```

Figure 8.3: The complete distortion model of the left camera.

$$\mathbf{D} = \begin{pmatrix} k_1 \\ k_2 \\ p_1 \\ p_2 \\ k_3 \end{pmatrix}$$

Figures 8.3 and 8.4 show the impact of the complete distortion models on each pixel of the images captured by the left and right robot's cameras, respectively. These models contain the radial and tangential distortion components of the cameras. Each arrow represents the effective displacement of a pixel induced by the lens distortion. Observe that points at the corners of the image are displaced by as much as 25 pixels. On these figures, the cross indicates the center of the image, and the circle represents the location of the principal point.

We implemented the stereo calibration of the robot's cameras by calibrating each camera independently and then applying geometric transformation of the external parameters to find out the geometry of the stereo setting. The external camera parameters are needed for both the correspondence problem, which determines the epipolar lines for determining point correspondences, and for triangulation, which is used for determining positions in the real world. Figure 8.5 illustrates the extrinsic model of the stereo



```
Pixel error                  = [0.1135, 0.1115]
Focal Length             = (492.335, 491.697)                            +/- [33.05, 32.68]
Principal Point           = (170.584, 144.627)                            +/- [4.545, 3.613]
Skew                          = 0                                                +/- 0
Radial coefficients        = (0.7815, -3.664, 0)                        +/- [0.1198, 1.452, 0]
Tangential coefficients  = (0.0005544, -0.0003065)                 +/- [0.004562, 0.008394]
```

Figure 8.4: The complete distortion model of the right camera.

camera calibration by using ten different views of the chessboard pattern. We used the left camera as world reference system, so the parameters to be found are the translation vector and rotation matrix of the right camera with respect to the left one.

The extrinsic parameters of the stereo pair describe the pose of the object relative to the cameras' coordinate system in terms of a rotation and a translation. In general, a rotation in any number of dimensions can be described in terms of multiplication of a coordinate vector by a square matrix of the appropriate size. Ultimately, a rotation is equivalent to introducing a new description of a point's location in a different coordinate system. Rotation in three dimensions can be decomposed into a two-dimensional rotation around each axis in which the pivot axis measurements remain constant. The translation represents a shift from one coordinate system to another system whose origin is displaced to another location; in other words, the translation vector is just the offset from the origin of the first coordinate system to the origin of the second coordinate system. Table 8.1 lists all the calculated internal and external parameters of the stereo camera calibration of the robot.

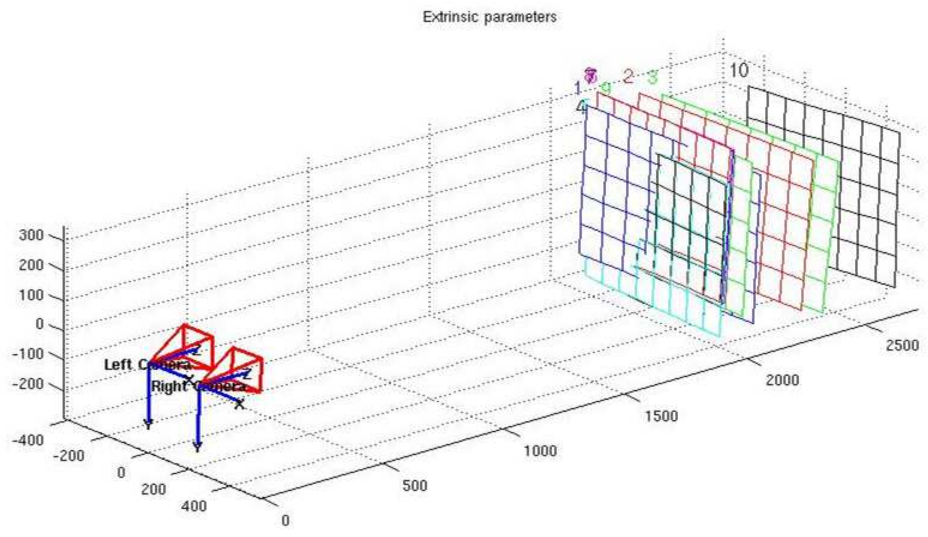


Figure 8.5: The extrinsic model of the stereo camera calibration.

Parameters	**Values**

Left intrinsic matrix	457.584275 0.000000 153.147984
	0.000000 457.584275 135.743784
	0.000000 0.000000 1.000000

Left distortion matrix	0.429594 3.272839 0.00 0.00 -35.568513

Right intrinsic matrix	457.584275 0.000000 172.711794
	0.000000 457.584275 148.465763
	0.000000 0.000000 1.000000

Right distortion matrix	0.476268 1.252977 0.00 0.00 -21.456410

Stereo rotation matrix	0.999794 -0.006803 -0.019144
	0.006036 0.999188 -0.039830
	0.019400 0.039706 0.999023

Stereo translation matrix	-2.532479 -0.063407 0.018712

Essential matrix	-0.001343 -0.021214 -0.062599
	0.067838 0.100428 2.529647
	0.048107 -2.530855 0.099654

Fundamental matrix	-0.000000 -0.000002 -0.002361
	0.000006 0.000009 0.104717
	0.001125 -0.108026 1.000000

Table 8.1: The resulting internal and external calibration parameters of the robot's cameras.

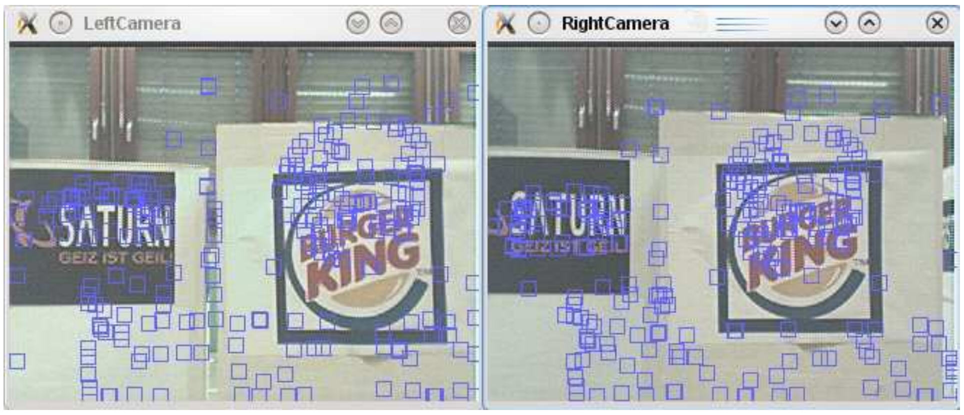


Figure 8.6: The corresponding features in the left and right images by using Lucas-Kanade optical flow in pyramids.

8.3 Stereo Vision and Landmark Localization

Stereo vision has the advantage that it is able to obtain an accurate and detailed 3D representation of the environment around the robot by passive sensing (i.e. cameras) and at a relatively low sensor cost. It is an important mechanism in robots, allowing judgments to be made based on the disparity between the images captured by each camera. We use stereo vision as a reliable and effective way to extract range information from the environment. The disparity map resulting from the stereo vision process is integrated with the landmark classification stage to obtain the position of the nearest landmarks to the robot. The stereo vision process is divided into four main steps: corresponding calculation, rectification, disparity map generation, and triangulation.

In the corresponding point's calculation stage, the features points in the left image are retrieved and their equivalent features in the right image are determined. The left image is scanned to find corners with big eigenvalues and then the features that are too close to stronger features are removed. Therefore, we use the Lucas-Kanade optical flow in pyramids [9] to calculate the coordinates of the feature points in the left captured frame and their corresponding points in the right captured frame. Figure 8.6 shows the feature points in the left image and their equivalent points in the right image.

The rectification stage is used to determine a transformation of each image plane so that pairs of conjugate epipolar lines become collinear and parallel to one of the image axes. The rectified images can be thought of as acquired by a new stereo rig, obtained by rotating the original cameras around the optical center. The important advantage of rectification is the computation of correspondences, by which a 2-D search problem is generally

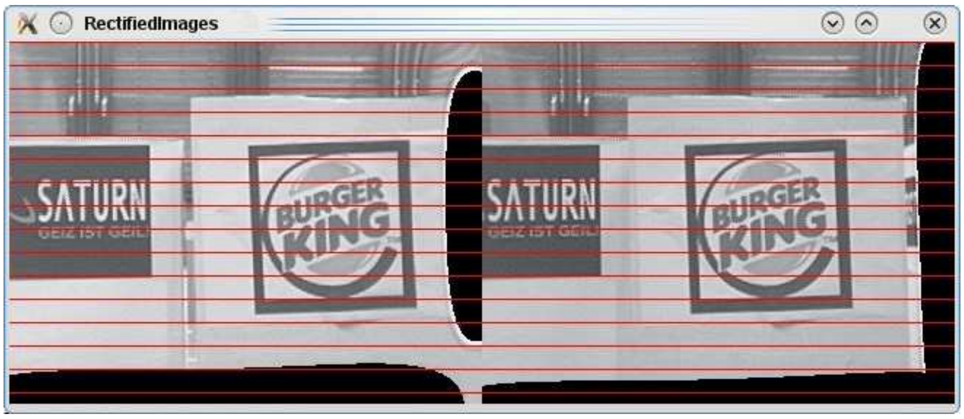


Figure 8.7: The resulting rectified image of the stereo pair.

reduced to a 1-D search problem, typically along the horizontal raster lines of the rectified images. Figure 8.7 shows the resulting rectified image for images illustrated in Figure 8.6. We used Bouguet's algorithm [9] which uses the rotation and translation parameters from two calibrated cameras. It simply attempts to minimize the amount of change reprojection produces for each of the two images (and thereby minimize the resulting reprojection distortions) while maximizing the common viewing area. The average error of epipolar geometry, which is computed by all good matches, is 0.615498 pixel.

Afterwards, the distance between the corresponding points in the left and right images in pixels is computed. The output of this step is a disparity map, where the disparities are the differences in x-coordinates on the image planes of the same feature viewed in the left and right cameras:$x^l - x^r$. We use the feature-based method by Birchfield et al. [4] to construct the disparity map. This method looks at features in one image and tries to find the corresponding feature in the other image. The features can be edges, lines, circles and curves. The main advantage of the feature-based algorithm is its speed. The process of finding features in both images and then calculating the disparity is carried out easily in real time. Figure 8.8 shows the disparity and depth map of the stereo images illustrated in Figure 8.6.

Finally, to calculate the object's position in the 3D environment, the disparity map should be turned into distances by triangulation. This step is called reprojection, and the output is a depth map. We use the triangulation method to compute the world coordinate of all points in the images by using the disparity map, the focal distance of the two cameras and the geometry of the stereo setting (relative position and orientation of the cameras). The triangulation calculates the approximate position of the landmarks in the real world, which will be used during the path planning stage and also in

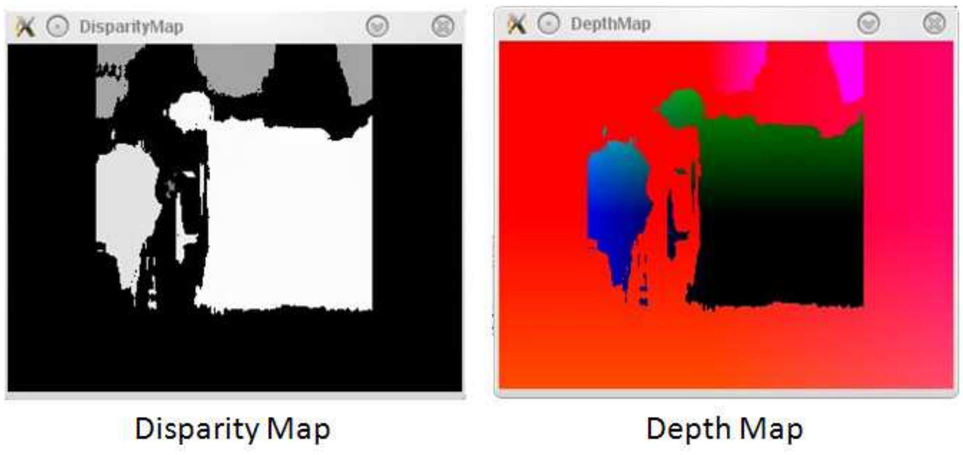


Figure 8.8: The disparity and depth maps of the stereo images.

executing the navigation task.

8.4 Landmark Detection and Segmentation

In robotic scenarios, the landmark detection stage should be capable of processing images extremely rapidly and of achieving high detection rates. The landmark detection stage is responsible for detecting and segmenting different types of landmarks from the captured image during robot navigation. Its main goal is to retrieve the landmarks from the captured image based on their shape and rejects false positives. It then segments the detected landmarks from the image background and stores them into individual images. These images will be fed to the recognition stage to classify the landmarks depending on both their features and the processed route.

We used sixteen different landmarks to examine our system as shown in Figure 8.9. All data of these landmarks are stored in the knowledge base. These landmarks are classified into three groups as follows:

- Logo landmarks: They have a unique symbol or trademark which is used to identify them from other landmarks. These symbols are used to recognize the landmarks in our miniature city (for more details, see chapter 6) during robot navigation. These landmarks have logos of supermarkets, department stores, and restaurant; such as the Lidl supermarket, the C&A store, and the Burger King Restaurant.

- Color landmarks: They have unique shapes and characteristics which set them apart from the other surroundings, such as their shape and color features. The railway station and town hall are examples of this type of landmarks.

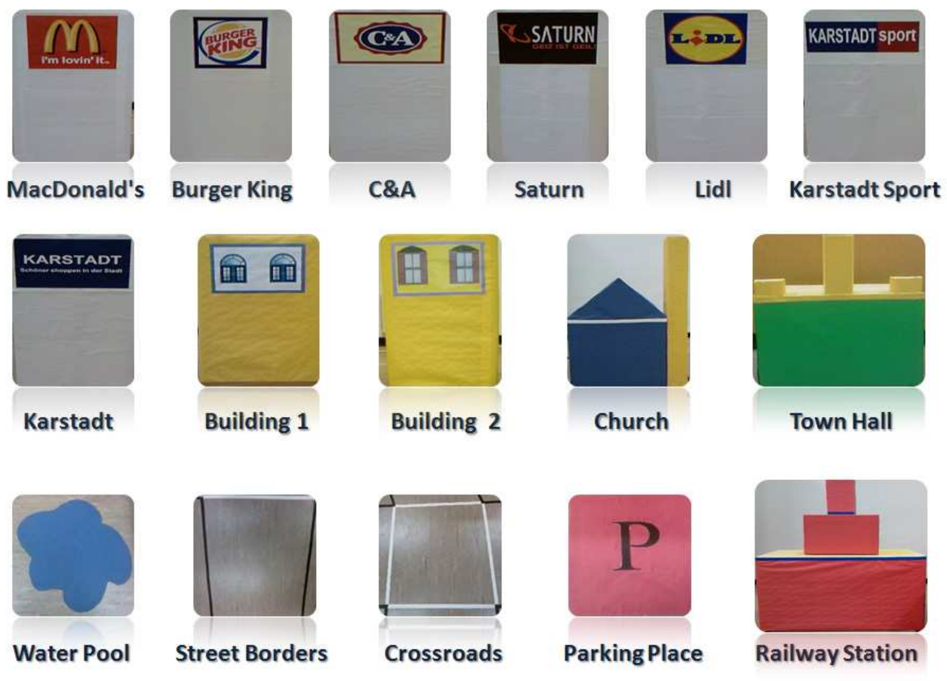


Figure 8.9: A set of landmarks used in our system.

- Street landmarks: They are simply recognized by their color and position in the miniature city. We recognize the crossroads in our miniature city by using white lines, whereas the street boundaries are identified as black straight lines.

The landmark detection stage is processed in four basic steps as shown in Figure 8.10. The first step is the down- and up-scaling process which is used to filter out the noise. It applies down and up sampling to the captured image by using Gaussian pyramid decomposition. In the second step, the canny filter is applied to find the edges in the input image. The canny edge detector gives a good approximation of the optimal operator. It maximizes the product of signal-to-noise ratio and localization [137]. The third step is to dilate the canny filter output to remove potential holes between edge segments. The last step is to find and approximate the contours. We find all contours in the image and restrict it to the extreme outer contours, then approximate the contours by using the Douglas Peucker algorithm [9].

The average time of the landmark detection stage is 105 ms. If the robot fails to detect any landmark in the captured image, vision planner sends a signal to the humanoid robot motion planner to pan-tilt the robot's head with a predefined angle or to instruct the robot to move closer toward the landmark.

As an alternative to the computationally expensive windowing strategies,

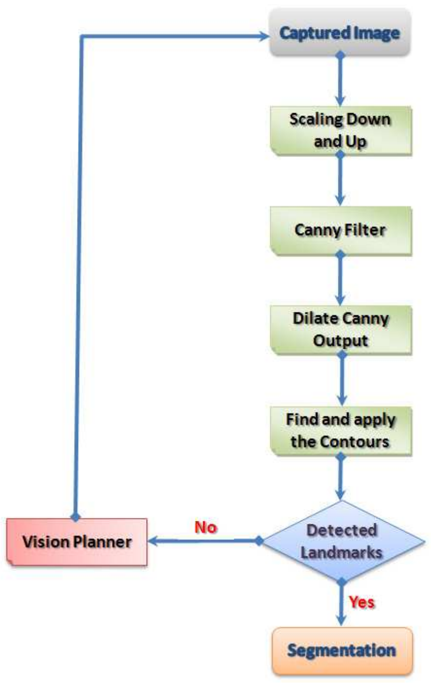


Figure 8.10: The flow diagram of the landmark detection stage.

an image segmentation strategy is proposed. This method could improve results by reducing background clutter. We crop the detected landmarks from the image background to reduce the processing time during the recognition stage. This allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. The segmentation technique is based on the outputs of the detection clutter. The detection stage provides it with the proposed landmark regions, whereas disparity provides it with the position of the landmarks with respect to the robot's position. Once the object is segmented from the background, it has to be represented in a compact way for future indexing.

8.5 Vision Planner

The vision planner decides which algorithm should be used to recognize landmarks seen by the robot during navigation. This learning is based on both simple attributes extracted on-line from the images and data retrieved from the symbolic and graphical representations of the processed route. Therefore, the vision planner chooses the suitable appearance-based technique for the detected landmarks depending on their features. For the appearance-based approaches, the vision planner can choose between Hough transform, color detection, or color histogram techniques. For the model-based approaches, it can choose whether or not to use the SIFT approach.

The vision planner concerns the problem of learning from interaction to achieve a goal. On each interaction step the robot senses the current state of the environment by using its cameras, and chooses a recognition technique to process this. The policy of choosing is some function that tells the robot which recognition approach should be chosen, and is learned based on data provided by the route description.

Finally, the vision planner stores the recognized landmarks and their positions. It can use this information to get an idea of where the position of the new landmarks is with respect to the detected landmarks. The vision planner provides the vision output to the symbol grounding stage in the motion planner to connect the symbolic representation of the landmarks with their equivalent physical data.

8.6 Landmark Classification

Landmark classification is the core stage of RLPS. It is implemented by using a two-step classification. The major advantages of the proposed two-step classification based method are its robustness and invariance towards scaling and translations. Also, it provides a good balance between fast processing time and high detection accuracy. First, an appearance-based method is used to classify the landmarks to get an initial estimation of the processed landmark. Then, a model-based classification is used to refine the recognition stage and obtain an accurate estimation of the landmark. This combination of appearance-based and model-based methods leads to a robust classification of the landmark and also speeds up the recognition process.

In addition to the proposed two-step classification, we proposed multiple methods integration for landmark recognition to handle many types of landmarks in the environment. The robot selects an appropriate method to detect landmarks according to the situation. All selected techniques are suitable to be adapted to mobile robots, and we evaluate them on a challenging dataset of landmarks (see Figure 8.9).This yields a flexible higher performance landmark classifier.

8.6.1 Appearance–Based Stage

We use the color histogram, color detection, and Hough Transform (HT) as appearance-based methods to get a fast rough classification of the landmarks. Selecting which algorithm should be used by a mobile robot computer vision system is a decision that is usually made a priori to the processing stage of the captured image. In order to decide which technique should be used by RLPS, the vision planner should be checked first.

For logo landmarks, a color histogram is calculated first to produce initial hypotheses before supplying the processed landmark to the model-based stage to get an accurate estimation of this landmark. As the RGB color space is not very stable with regard to alterations in the illumination, the representation of a color with the RGB color space contains no separation between the illumination and the color parts. Therefore, we used the HSV color space, which is robust against alterations in illumination, because the color parts and the illumination are represented separately. The color histogram returns the hue distribution of the detected landmarks and does not preserve the geometric structures of these landmarks. The hue color component is determined by the dominant wavelength in the spectral distribution of light wavelengths. This component is ideally independent of the lighting conditions and the distance between object and observer. Therefore, they are reliable parameters for object recognition.

On the other hand, color histograms of training images, which are stored in the database, are computed offline to reduce the consumed time. Only the histogram of the tested landmark needs to be calculated. Comparison of these two distributions (detected and stored landmarks) which are represented in the form of histograms is made on the basis of the correlation coefficient k_c for these distributions, which has the form:

$$k_c = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\left[\sum_{i}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2\right]^{1/2}}$$

where x_i and y_i are histograms for the detected and stored color distributions, respectively. Whereas, $\overline{x}$ and $\overline{y}$ are the average values of these distributions. If the two histograms are identical, the correlation coefficient is equal to unity. The stronger the correlation coefficient differs from unity, the stronger is the diversity between the considered distributions. Thus the correlation method of comparison of histograms is the simplest and strongest method for the analysis of observed data for a certain meteorological phenomenon. The resulting correlation coefficients of the color histograms with the information retrieved from the topological map and the route symbolic representation are used as an initial estimation of the processed landmark. Figure 8.11 shows the hue color histograms of some landmarks which are detected by using their logo. The average time of the color histogram in our system is 3.4 ms.

Shape landmarks are detected by using their colors and shapes. First, we convert the color space of the processed image from the RGB to the HSV system which is good for illumination variations. Then, the strength of the captured colors is increased by increasing the saturation component of the image by a certain threshold which would make it easier for the robot

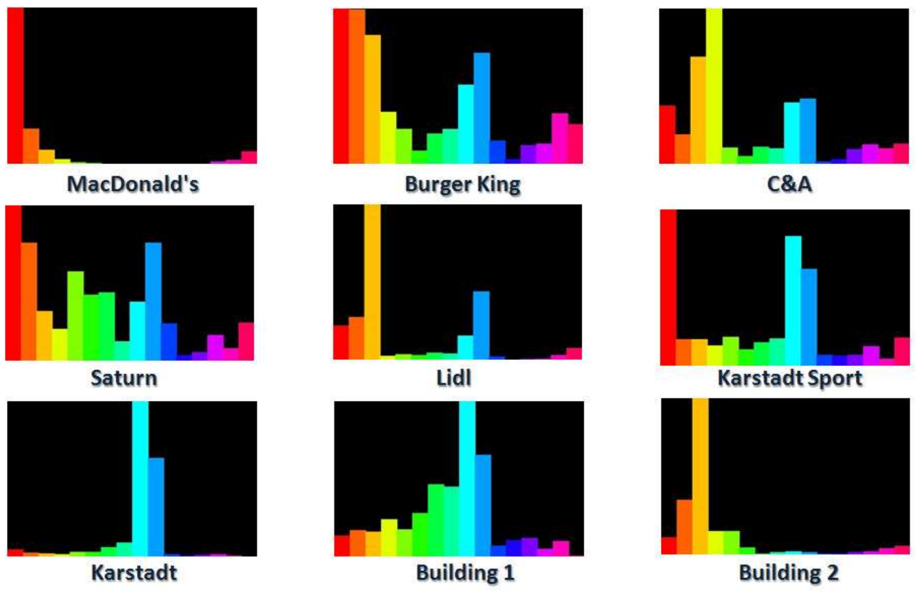


Figure 8.11: The hue color histograms of logo landmarks.

to realize images nicely. Afterwards, the color space is converted again to RGB color space to check the landmark color. Finally, we treat the resulting image by using erosion followed by a dilation operation. These two operations arise in a wide variety of contexts such as removing noise, isolating individual elements, and joining disparate elements in an image [137]. In general, whereas dilation expands the processed region, erosion reduces this region. Moreover, dilation will tend to smooth concavities and erosion will tend to smooth away protrusions. Figure 8.12 illustrates the basic four stages of the color detection stage for a railway station landmark.

Finally, to recognize crossroads and street boundaries, we used the Progressive Probabilistic Hough Transform (PPHT) [101] to find the straight lines on the ground. PPHT minimizes the amount of computation needed to detect lines by exploiting the difference in the fraction of votes needed to reliably detect lines with different numbers of supporting points. This algorithm is ideally suited for real-time applications with a fixed amount of available processing time, since voting and line detection is interleaved. The most salient features are likely to be detected first. PPHT is robust to noise and occlusions and it is not computationally expensive as compared to the normal Hough transform. The average detection time of PPHT in our system is 1.66951ms.

At the beginning, we applied the Canny edge detector to find edges in the processed image, then PPHT randomly selects sets of data points from which the surface parameters can be directly computed and records. If many data sets yield the same parameters, a high score for these parameters

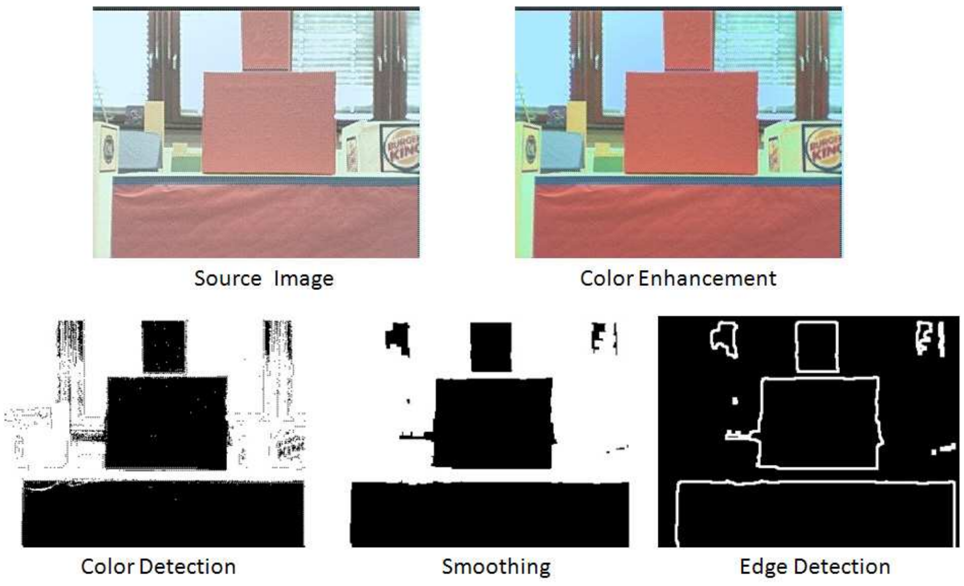


Figure 8.12: The color detection stages of the "railway station" landmark.

is obtained. Figure 8.13 shows the detected lines by using PPHT during recognized crossroads and street boundaries.

8.6.2 Model–Based Stage

This stage classifies the detected landmarks according to their geometrical properties. After applying the appearance-based technique and having some hypotheses, we used the model-based stage to refine the recognition stage and obtain an accurate estimation of the landmark.

The SIFT technique is used to classify the landmarks which have unique logos in the environment, such as the Lidl supermarket, the Saturn store, and the Burger King Restaurant. Figure 8.14 illustrates the main components of the recognition process by using the SIFT approach. Initially, SIFT descriptors of the detected features in the processed landmark are determined. Then the resulting descriptors are matched to the ones with the highest hypotheses which are stored in the landmark knowledge base by using the Euclidean distance. False matches are rejected if the distance of the first nearest neighbor is not distinctive enough when compared with that of the second. Figure 8.15 shows SIFT descriptors of the detected features for some landmarks, whereas the matching process between the examined images and the stored landmarks is shown in Figure 8.16.

Once a set of matches is found, the Hough Transform is used to cluster each match of every knowledge base image depending on its particular transformation. Although imprecise, this step generates a number of initial

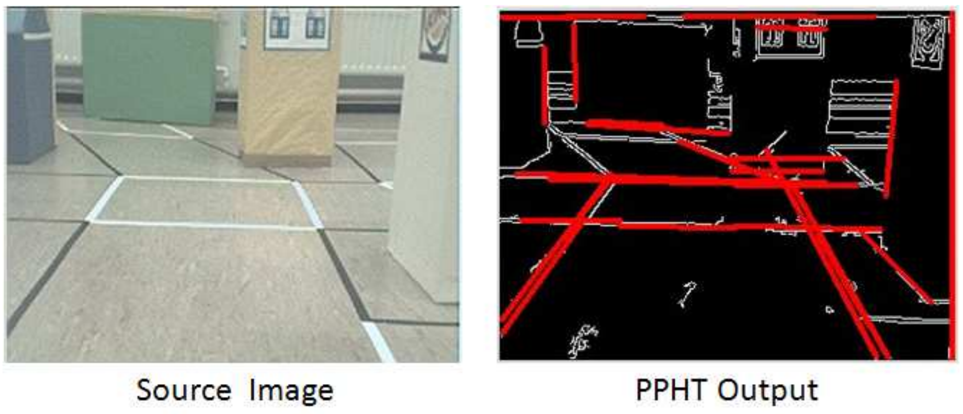


Figure 8.13: The detection of the crossroads and street boundaries by using PPHT.

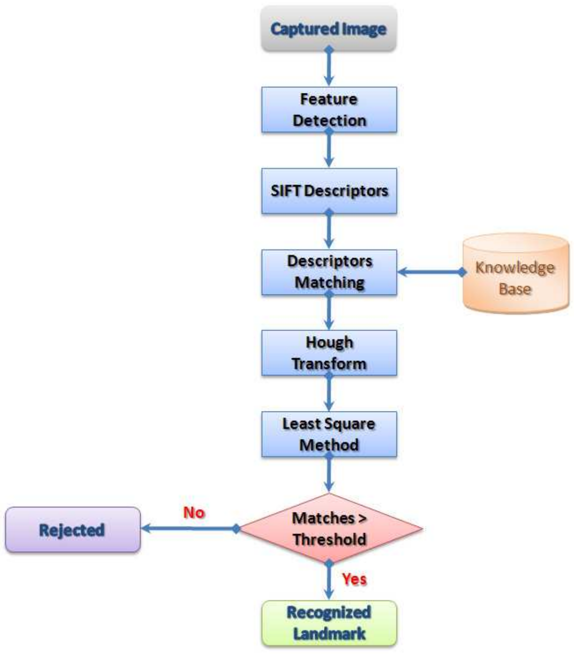


Figure 8.14: The block diagram of the SIFT approach.

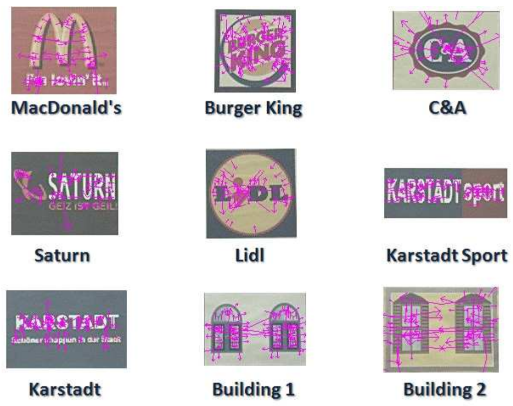


Figure 8.15: The SIFT descriptors of detected features for logo landmarks.

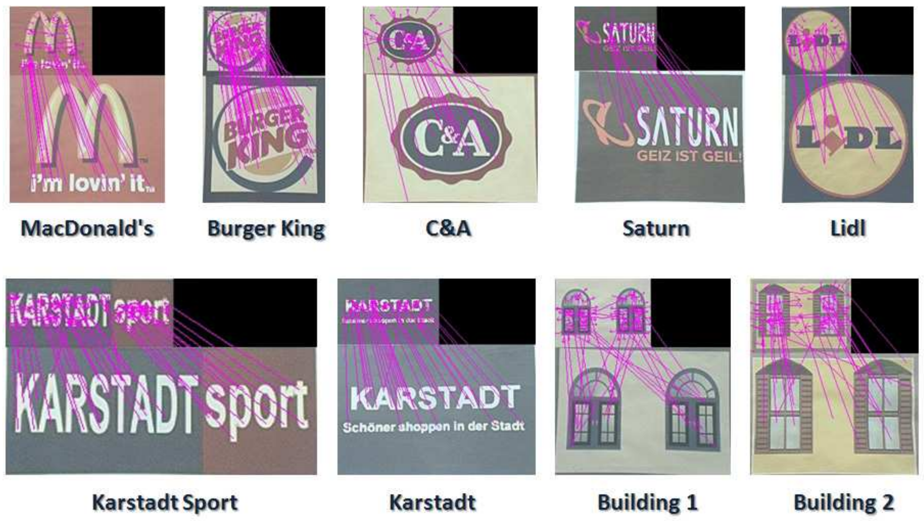


Figure 8.16: The matching between the calculated SIFT descriptors of the processed images and those of the stored landmarks in the knowledge base.

coherent object hypotheses and removes a notable portion of the outliers that could potentially confuse more precise but also more sensitive methods. All clusters with at least three matches for a particular training object are accepted, and fed to the next stage: the Least Squares method, used to improve the estimation of the affine transformation between the model and the test images.

Finally, the landmark with the highest matching points is chosen as the recognized landmark (i.e. the more SIFT-matches found in an image, the more likely it is that that image contains the object). If the number of matches exceeds an object-dependent threshold, the object is considered recognized. Some objects have more features than others and are thus easier to recognize. To minimize the number of false positives, the threshold depends on the number of features found during training. The average SIFT recognition time in our system is approximately 550 ms.

After recognizing the landmarks, the topological map is used to specify which landmarks will be processed by the robot. The robot focuses only on the landmarks which are already mentioned in the route description and presented in the topological map. This leads to decreasing the processing time be ignoring unwanted landmarks. The nearest landmark in the route description to the robot is chosen by using the disparity map, and then triangulation is used to calculate the approximate world position of this landmark. After recognizing landmarks and calculating their locations in the real world, the robot navigates to the landmark by using the information supplied by the topological map.

8.7 Experimental Results

In this section, we demonstrate that RLPS is more powerful as it learned to recognize a wide range of landmarks and to find them in the environment even when there are landmarks of similar colors in the field of view. It also handles landmarks with viewpoint variances, occlusions, and illumination changes. The main purpose of these experiments is to evaluate our proposed techniques to be used in an indoor mobile robot. As mentioned previously, we implemented our system on a HOAP-2 humanoid robot. The robot is equipped with two 0.25" CMOS unsynchronized cameras. The two cameras are linked via USB connections to an Intel 1.73 GHZ Duo laptop with 3GB RAM. The operating system is a standard Linux distribution and kernel with Video for Linux drivers for video capture. In its current form the system can process 320 x 240 images at 25 fps.

To evaluate the performance of our robot landmark processing system, the following procedure was carried out: first, we approximately captured 1000 different images from our miniature city with different viewing angles, distances from the robot, occlusions, and illumination changes. Then, we

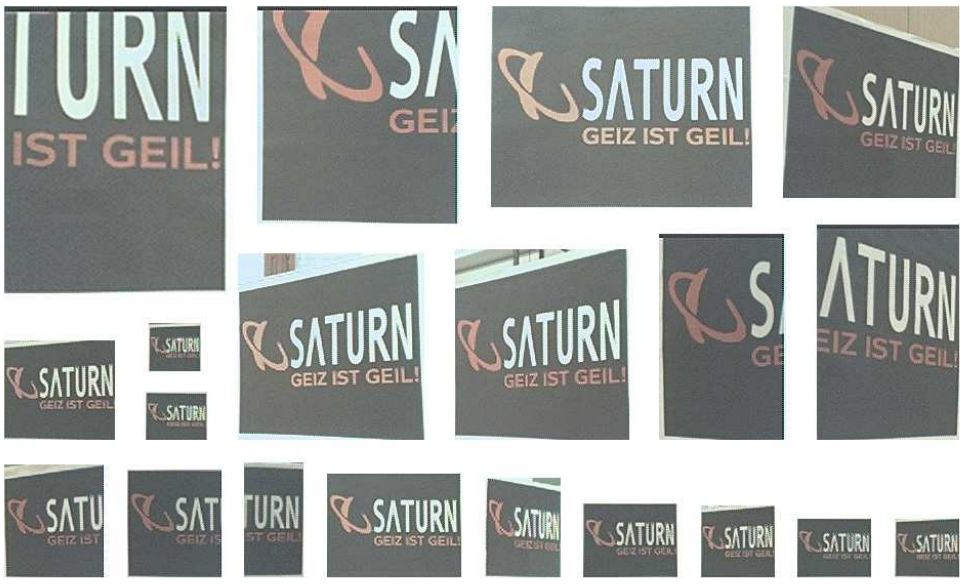


Figure 8.17: An example of some images used for evaluating RLPS.

tested these images by using our system and other state-of-the-art techniques to validate the suitability of the RLPS for our purpose. Finally, we analyzed the resulting data to evaluate RLPS with respect to other techniques.

8.7.1 Evaluation of The Logo Landmarks' Technique

In this experiment, we used a dataset of images that contains approximately 900 testing images (100 images for each of the logo landmarks: the C&A store, the Lidl supermarket, the Burger King restaurant, etc.). These images are captured during the robot's navigation in the miniature city. Figure 8.17 shows a subset of the tested images for the Saturn store landmark. Each tested image contains one or more instances of the landmarks, some of them with illumination changes, partial occlusions, or viewpoint variances. In addition to our proposed method, we selected three state-of-the-art object recognition methods, which are suitable to be adapted to the mobile robot's applications, to compare them with the performance of our proposed technique. We used the original SIFT, color histogram, and Speeded Up Robust Features (SURF) [1] techniques to compare them with the detection rate of the proposed system. Evaluation is done by comparing the number of detected objects, false negatives, and false positives resulting by applying these approaches to the processed dataset.

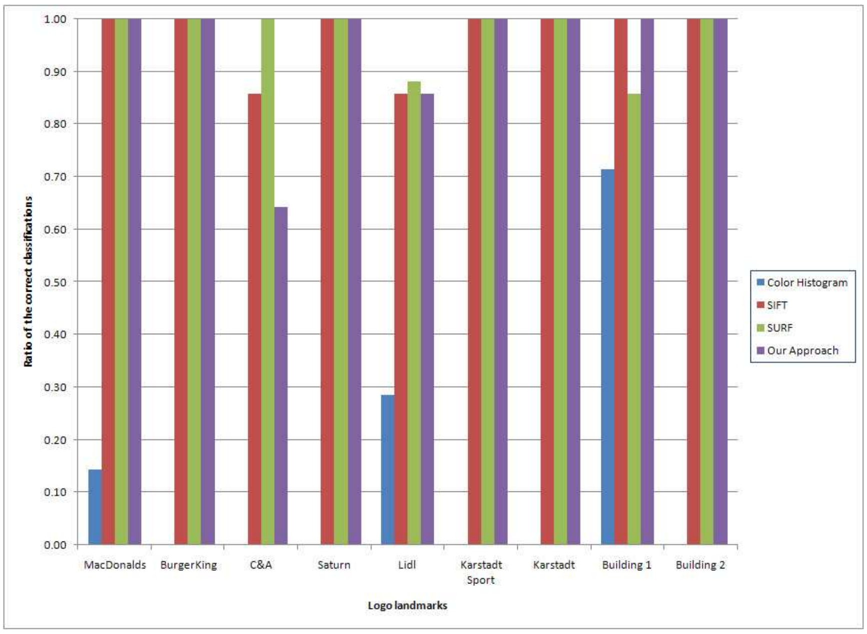


Figure 8.18: The ratio of the correct classifications for the landmarks with occlusions.

Landmarks with Occlusions

In this experiment, we tested 315 images that contain landmarks with occlusions. Figure 8.18 shows the recognition rate for each tested landmark by using four different object recognition techniques. It can be noticed that the color histogram method completely fails to detect and recognize some landmarks with occlusions, such as Burger King, C&A, and Saturn. The average detection rate of the color histogram is less than 15%. On the other hand, our system handles the cropped landmarks efficiently except in a few cases which result from false positives in the first recognition stage. The average detection rate for SIFT, SURF, and our system is more than 94%.

Landmarks with Viewpoint Angle Variations

For landmarks with viewpoint variations, we used 360 images to test the performance of our system and compare it with the performance of the other techniques. As seen in Figure 8.19, it can be observed that the recognition rate of our system is better than other classification methods. The detection rate of RLPS is more than 85%, whereas the other techniques are less than 75%. For the color histogram method, the detection rate changes depending on the landmark. For example, the detection rate of the MacDonald's landmark using a color histogram is 100%, whereas the Saturn landmark is not detected at all.

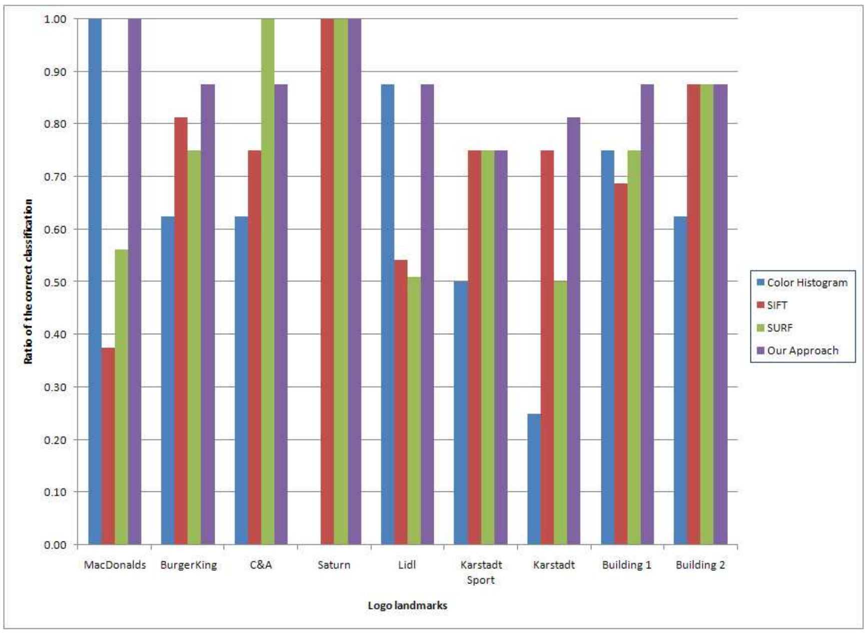


Figure 8.19: The ratio of the correct classifications for the landmarks with different viewing angles.

Landmarks with Illumination Variations

This experiment is conducted to know the effect of the illumination variations on the correct classification rate. The tested images are classified into five groups. The first one (Group N) was captured at the normal room lighting condition. The second and third groups (Group D1 & D2) were taken with darker illuminations. The last two groups (Group L1 & L2) were taken with brighter illuminations. Figure 8.20 shows an example of each group. The effect of the illumination variations to the detection rate is illustrated in Figure 8.21. It can be seen that our technique achieves a much better recognition rate than the other methods. It can be also noticed that the detection rate of the color histogram technique is strongly affected when illumination is increased.

Landmarks with Distance Variations

We tested the effect of the distance between the detected landmarks and the robot's position on the detection rate. Figure 8.22 illustrates the effect of the distance variation on the detection rate for the tested four techniques. The effect of the distance variations on the resulting number of the corresponding features in both SIFT and SURF techniques is shown in Figure 8.23.

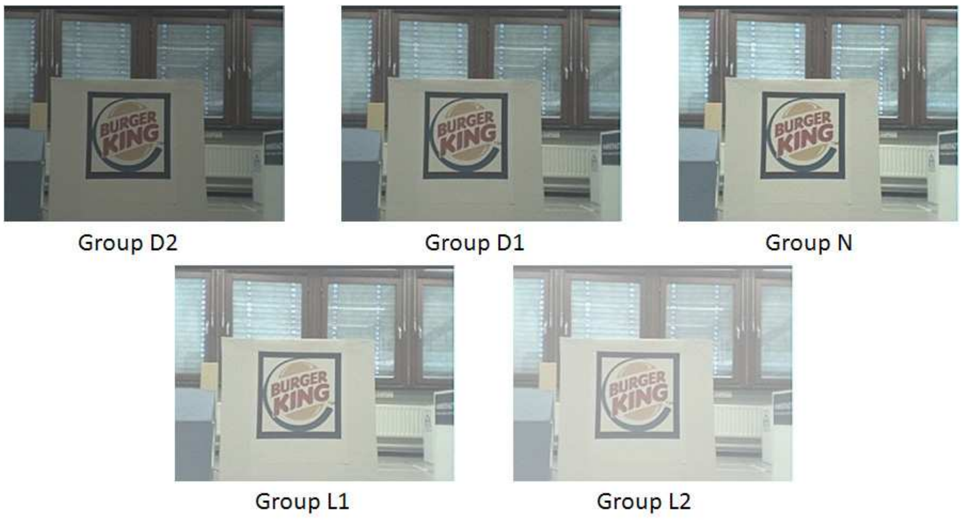


Figure 8.20: Images with different illumination conditions.

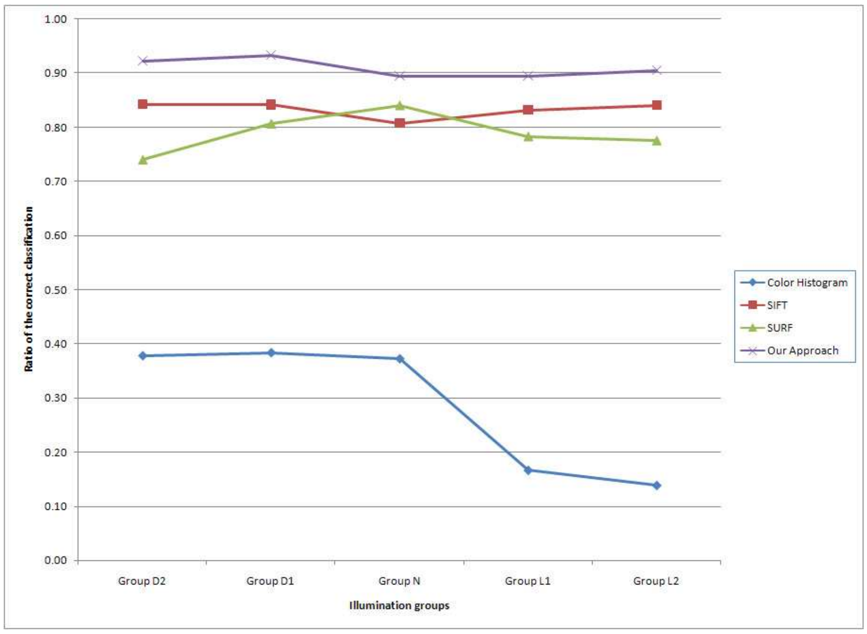


Figure 8.21: The ratio of the correct classifications for the landmarks with illumination changes.

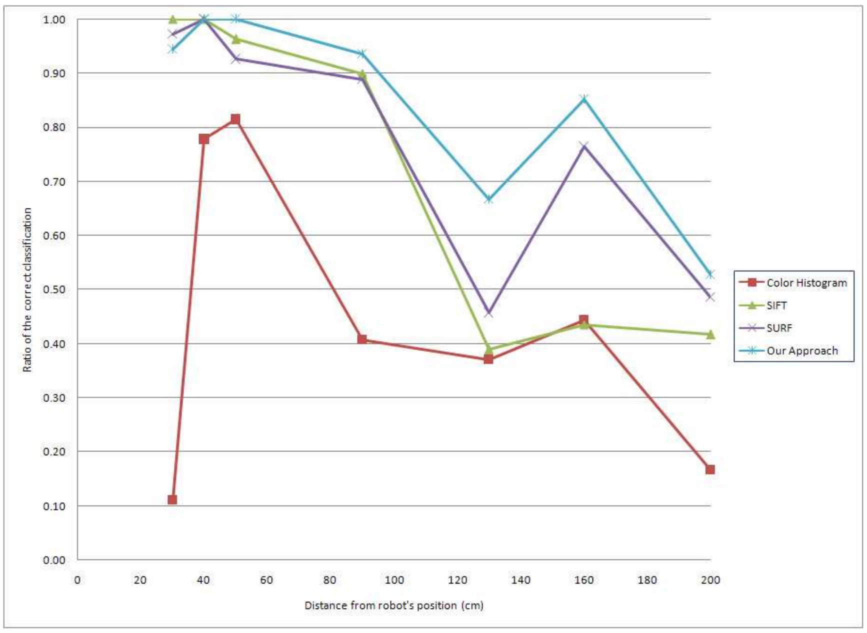


Figure 8.22: The detection rate vs. the distance from the robot's position.

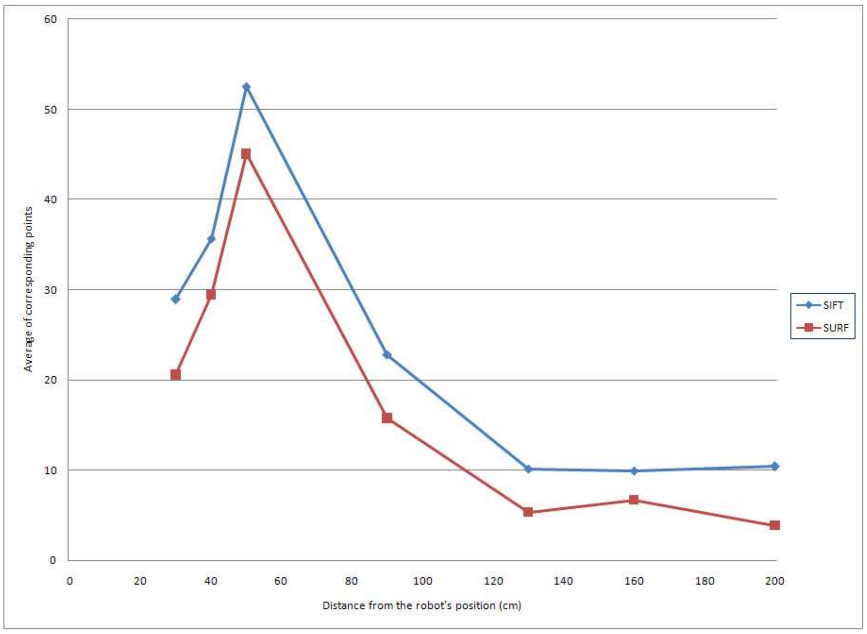


Figure 8.23: The number of the resulting corresponding features vs. the distance from the robot's position.

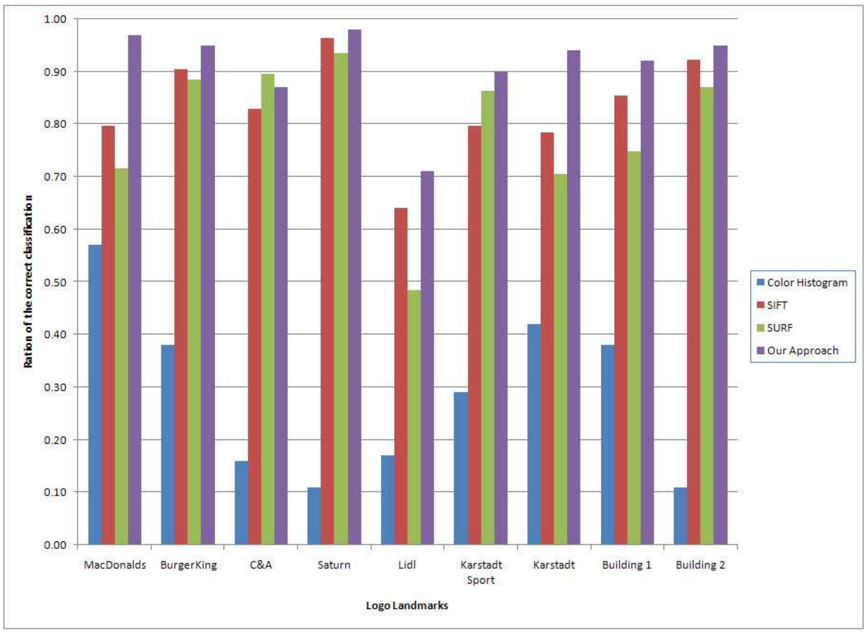


Figure 8.24: The ratio of the correct classifications for all tested images.

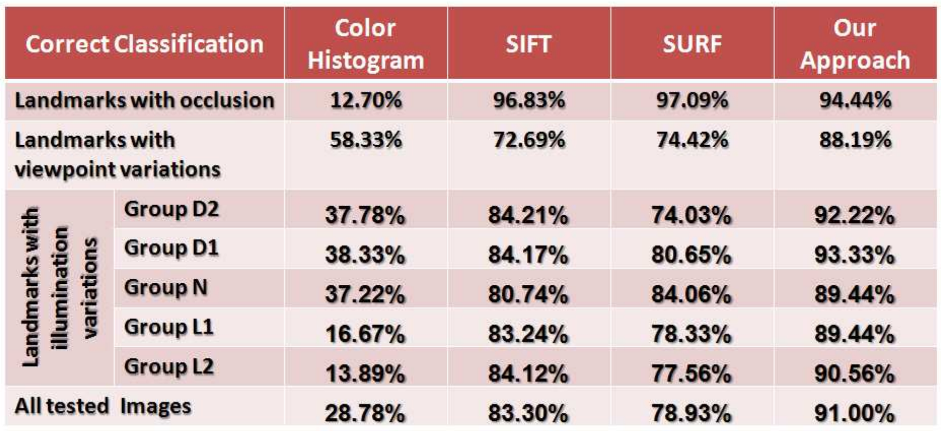
Average Classification Rate

To test the average performance of our proposed system, we calculated the average detection rate of all captured images for all tested classification techniques under different conditions. The ratio of the correct classifications for each tested landmark in the dataset is illustrated in Figure 8.24. It can be observed that the correct recognition rate of our proposed technique is more that 90% for most of the tested landmarks. It can be also noticed that the detection rate of the color histogram is the lowest rate among the other tested techniques. For SIFT and SURF methods, the landmark classification is based on the fact that the more matches are found in the image, the more likely it contains the landmark. Table 8.2 illustrates the detection rates for each technique under different conditions. As seen in this table, our proposed system was able to achieve 91% classification accuracy on the test set, yielding only 9% false positives and false negatives.

For each tested classification method, we measured the average time for detection, average classification time, and average total processing time. Table 8.3 lists the time consumed in each stage.

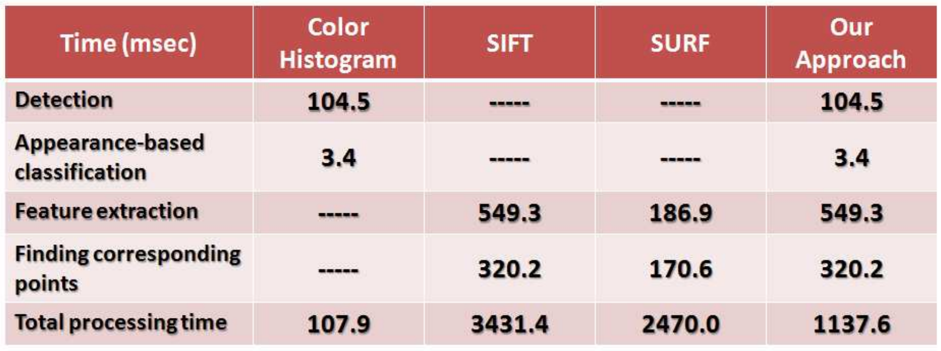
8.7.2 Evaluation of The Color Landmarks' Technique

For color landmarks, we capture 20 different images for each landmark to test the performance of the proposed technique. The captured images are processed to calculate both the consumed time in each stage and the detec-



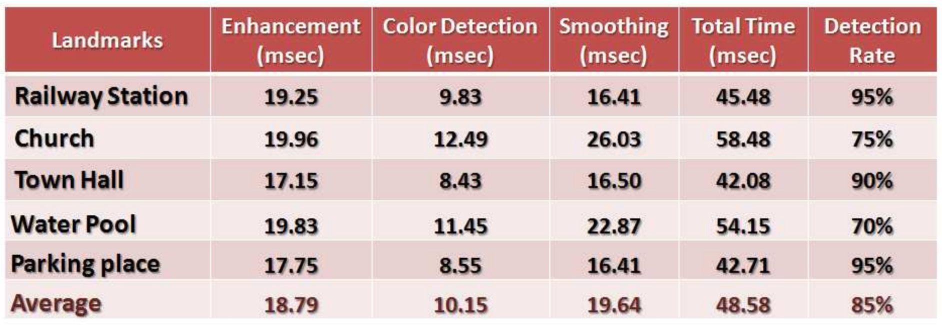
Correct Classification		Color Histogram	SIFT	SURF	Our Approach
Landmarks with occlusion		12.70%	96.83%	97.09%	94.44%
Landmarks with viewpoint variations		58.33%	72.69%	74.42%	88.19%
Landmarks with illumination variations	Group D2	37.78%	84.21%	74.03%	92.22%
	Group D1	38.33%	84.17%	80.65%	93.33%
	Group N	37.22%	80.74%	84.06%	89.44%
	Group L1	16.67%	83.24%	78.33%	89.44%
	Group L2	13.89%	84.12%	77.56%	90.56%
All tested Images		28.78%	83.30%	78.93%	91.00%

Table 8.2: Detection rate of the tested images for the four classification techniques.



Time (msec)	Color Histogram	SIFT	SURF	Our Approach
Detection	104.5	-----	-----	104.5
Appearance-based classification	3.4	-----	-----	3.4
Feature extraction	-----	549.3	186.9	549.3
Finding corresponding points	-----	320.2	170.6	320.2
Total processing time	107.9	3431.4	2470.0	1137.6

Table 8.3: The average consumed time of each stage for all tested classification techniques.



Landmarks	Enhancement (msec)	Color Detection (msec)	Smoothing (msec)	Total Time (msec)	Detection Rate
Railway Station	19.25	9.83	16.41	45.48	95%
Church	19.96	12.49	26.03	58.48	75%
Town Hall	17.15	8.43	16.50	42.08	90%
Water Pool	19.83	11.45	22.87	54.15	70%
Parking place	17.75	8.55	16.41	42.71	95%
Average	18.79	10.15	19.64	48.58	85%

Table 8.4: The detection time and rate of the color landmarks.

tion rate for each landmark. Table 8.4 shows the detection time and rate for all color landmarks. It also lists the average total time and detection rate for all color landmarks.

8.8 Discussion

In this chapter, we have presented our current effort toward building a robust and fast robot landmark processing system. We used a more natural approach in terms of computational efficiency to recognize the landmark online during robot navigation. The appearance-based techniques of the detected landmarks are used to provide the rough initial estimate of the landmark. Then, we processed the resulting hypotheses with a model-based approach to calculate an accurate estimation of the landmark.

In addition to the proposed two-step classification, we proposed multiple methods integration for landmark recognition to handle many types of landmarks in the environment. The robot selects an appropriate method to detect landmark according to the situation. All selected techniques are suitable to be adapted to mobile robots, and we evaluate them on a challenging dataset of landmarks.

Stereo vision is used to calculate the 3D position of the landmarks in the real world. We used stereo vision as a reliable and effective way to extract range information from the environment. The disparity map resulting from the stereo vision process is integrated with the landmark classification stage to obtain the position of the landmarks nearest to the robot.

On the other hand, we tested the performance of our robot landmark processing system. We captured approximately 1000 images for landmarks during robot navigation in the miniature city. We compared our proposed technique with three different state-of-the-art object recognition techniques. We measured the ratio of correct classification of the landmarks under different conditions, such as occlusions, viewpoint variations, distance variations, and illumination changes. The different advantages and drawbacks found for each method are highlighted, and some ideas for extending them are proposed. Evaluation is done comparing the number of detected objects and false positives for the tested approaches.

CHAPTER 9 Humanoid Robot Motion Planner

The autonomous navigation for humanoid robots comprises an increasingly important research area. The development of practical motion planning algorithms and obstacle avoidance techniques is considered as one of the most important fields of study in the task of building autonomous or semiautonomous robot systems. This leads to a rising demand for motion planning algorithms which are suited to the unique characteristics of bipedal humanoid robots and their typical operating environments. Therefore, motion planners which are implemented for wheeled mobile robots cannot be applied to bipedal humanoid robots. The motion planners designed for humanoid robots combine both path planning generation and the ability of executing the resulting path with respect to the kinodynamic characteristics of the humanoid robot. These planners should consider the specific dynamical constraints and stability problems which significantly reduce the motion range of the humanoid robots.

In this chapter, we present a time-efficient hybrid motion planning system for a Fujitsu HOAP-2 humanoid robot in indoor and miniature city environments. The proposed technique is a combination of sampling-based planner and D* Lite search to generate dynamic footstep placements in unknown environments. It generates the search space depending on non-uniform sampling of the free configuration space to direct the computational resources to troubled and difficult regions, such as turns and narrow passages. A modified cylinder model is used to approximate the trajectory for the robot's body-center during navigation. It calculates the actual distances required to execute different actions of the robot and compare them to the distances from the nearest obstacles. D* Lite search is then implemented to find dynamic and low-cost footstep placements within the resulting configuration space. The proposed hybrid algorithm reduces the searching time and produces a smoother path for the humanoid robot with low cost.

9.1 Overview of The Proposed Motion Planner

In general, the motion planning problem is characterized by the ability to compute a collision-free path for a mobile robot from its initial position to a final position through its workspace. We developed a motion planning system for a humanoid robot to execute route-based navigation tasks. The proposed motion planner allows the humanoid robot to take advantage of its bipedal capabilities and navigate in its surrounding environment to accomplish autonomous biped locomotion. The planner operates at the level of footsteps and ignores the lower-level details of leg movements and control.

To reach places in unknown environments, there is often a need to re-plan paths online based on the new findings extracted from the robot's visual perception. A natural way of updating plans is to first select a path based on the initially presented knowledge for the robot, then move along that path for a short time while collecting new information from the robot's vision. Based on the new findings, the path is then re-planned. As discussed in the last two chapters (Chapters 7 & 8), the motion planner is based on two main inputs. The first input is the initial path estimation which results from the symbolic representation of the processed route. It provides the motion planner with the relationships between locomotion actions, spatial relationships, and landmarks which are extracted from the processed route. The second input is the processed vision data from the stereo vision and landmark detection stage of our humanoid robot navigation system. It contains the information about the detected landmarks and their positions in the route environment.

The main task of the proposed motion planner is to find a sequence of actions as close to optimal as possible that causes the robot to reach its target location while avoiding the obstacles in an unknown environment. In other words, the motion planner is implemented to plan the motion and the footstep placements for the humanoid robot while moving in an unknown environment. It is also used to re-plan the robot's motion depending on the new findings sensed by stereo vision sensors during navigation. To accomplish this task, the planner should first have a way to connect the high-level cognitive processes, which perform abstract reasoning and generate plans for actions, with their corresponding sensorimotor processes that observe the physical world and execute actions in it. Secondly, the planner has to handle the uncertainties in the sensors and motors of the robot during navigation. Thirdly, it should plan the head motion of the robot depending on the processed route to detect the mentioned landmarks in the route description. Fourthly, it should have a suitable technique for the characteristics of the humanoid robot to avoid obstacles during path planning. Finally, it has to generate shortest low-cost footstep placements to execute the generated path. The used technique should be dynamic to handle new findings in the route environment without planning from scratch.

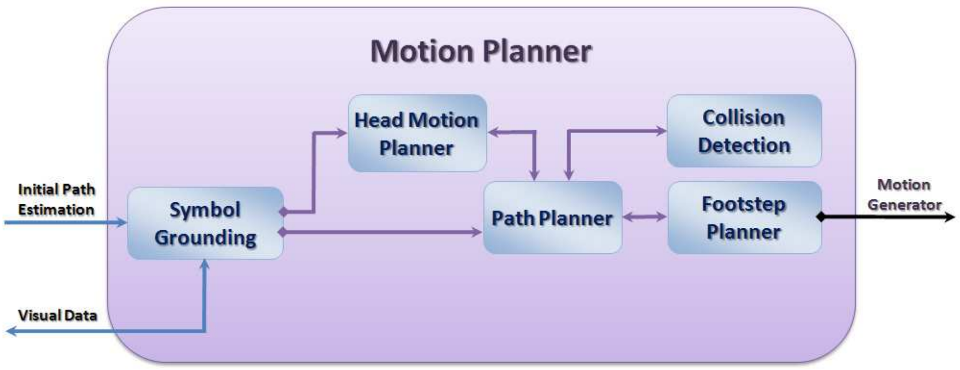


Figure 9.1: The architecture of the humanoid robot motion planner.

Our motion planner is based on a decoupled trajectory planning approach [14, 71] which is implemented to first extract the path for the robot without considering its dynamic constraints. Then, it applies the kinodynamic and physical constraints of the robot to the resulting path. The proposed planner is used to solve the motion-planning problem and handle the kinodynamic constraints of the HOAP-2 humanoid robot [41]. The planner is processed as two sequential phases. First, a sampling-based algorithm computes a collision-free path for the described route by ignoring system dynamics of the humanoid robot. Then, both the footstep planner and the motion trajectory generator are used to compute appropriate controls to implement the desired path and generate feasible motions for the humanoid robot.

Figure 9.1 shows the architecture of our humanoid robot motion planner. The motion planner consists of five basic components: symbol grounding, head-motion planner, collision avoidance, path planner, and footstep planner. The symbol grounding phase is used to connect the symbolic representations of landmarks and locomotion actions to their equivalent perceptual landmarks and motion procedures, respectively. The output of the symbol grounding stage is provided to the head motion planner to plan the motion of the robot's head with respect to the positions of the landmarks. It is also supplied to the path planner to generate the shortest feasible roadmap graph of the robot's path.

The path planner is implemented to extract the minimal feasible C_{free} and generate the shortest path to the target position. We used the Lazy Probabilistic RoadMap mechanism (Lazy-PRM) with a non-uniform sampling to avoid the computational complexity of generating a denser search area. The planner directs the computational resources to troubled and difficult regions, such as narrow passages, leaving the larger open spaces sparsely populated. A smoothing penalty is also associated to the nodes to encourage the generation of gentle paths along the middle of the empty spaces.

The sampling is increased in complex areas and leaves out simple areas with lower resolution density.

For collision detection, a cylinder model is used to approximate the trajectory for the body center of the humanoid robot during navigation. It calculates the actual areas required to execute different motion actions of the humanoid robot and compares them with the distances to the nearest obstacles. Collision detection is carried out off-line during the creation of C-space to speed up the actual search for the path.

Finally, the footstep planner is implemented to find smooth and low-cost footstep placements of the humanoid robot within the resulting C_{free}. It uses D* Lite search to reduce searching time and produces a smoother dynamic path for the humanoid robot at a low cost. In the following sections, the building blocks of the motion planner will be discussed in detail. The implementation of the hybrid algorithm for robot motion planning and the generation of the sequence of footstep location for the humanoid robot will be elucidated.

9.2 Symbol Grounding

As mentioned in Chapter 7, our humanoid robot navigation system is based on the route instructions provided by the user to describe the navigation task for the robot in miniature city or indoor environments. The resulting route description is processed to generate an initial path estimation for the robot in both symbolic and graphical representations. These representations are supplied to the motion planner as guidance during the planning process. They also provide the planner with the relationships between processed actions and mentioned landmarks in the route description. The second input to the motion planner is the processed vision data from the robot's cameras. The humanoid robot begins from the start point and moves along the estimated path to collect information about its route environment. As discussed in Chapter 8, the captured images are processed to detect, recognize, and localize the landmarks during the robot navigation. The resulting information is supplied to the motion planner to re-plan the robot's path depending on the new findings. The symbolic and graphical route representations and the resulting vision data are processed by the symbol grounding stage in the motion planner to connect each symbol in the route description to its corresponding perceptual data.

The main function of the symbol grounding stage is to incorporate the high-level cognitive processes with their corresponding sensorimotor processes. The high-level cognitive processes are presented in our system as a processed route description to perform abstract reasoning and generate plans for robot actions. They use symbols to denote both landmarks and robot actions. Otherwise, the sensorimotor processes observe the physical

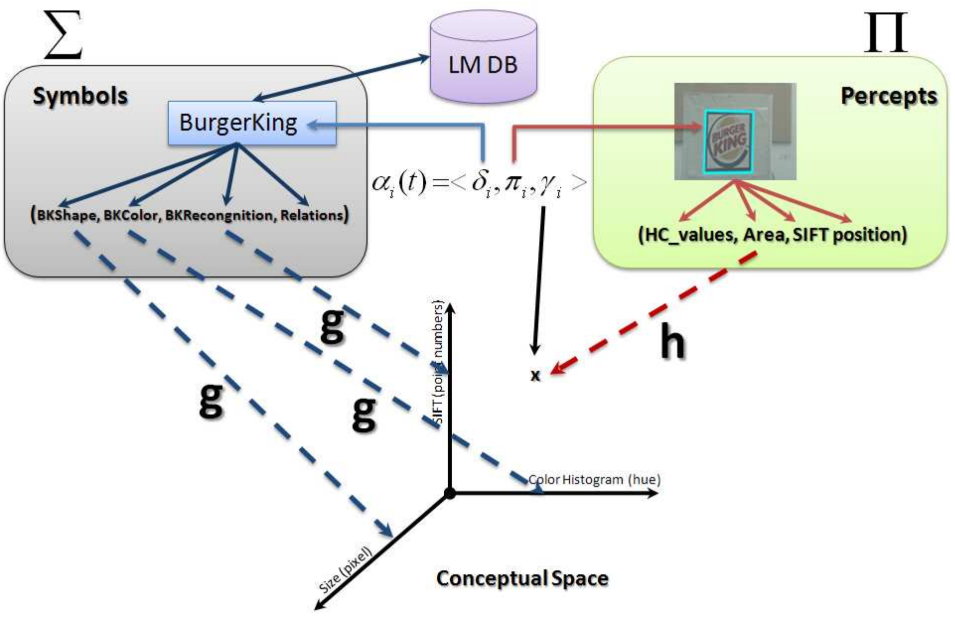

Σ
Π
LM DB
Symbols
BurgerKing
(BKShape, BKColor, BKRecongnition, Relations)
Percepts
(HC_values, Area, SIFT position)
$\alpha_i(t) = <\delta_i, \pi_i, \gamma_i>$
g
g
g
h
x
SIFT (point numbers)
Color Histogram (hue)
Size (pixel)
Conceptual Space

Figure 9.2: The anchoring process of symbolic and perceptual data for a landmark.

world and execute actions in the route environment. They operate from the cameras' data that originate from observing these landmarks. If the overall system is to perform its tasks successfully, it needs to make sure that these processes are successfully connected to indicate the same physical objects.

To solve the symbol grounding problem, a methodology is needed to resolve situations where the sensors detect several landmarks that are consistent with the symbolic description of a desired landmark. In order for an autonomous system to function robustly when faced with such ambiguous situations, it needs to reason and act in a way that allows it to distinguish between the perceived objects and determine their correct correspondents. The plan involves finding out relevant information about the landmarks until the correct landmark is identified. We used the perceptual anchoring via conceptual spaces to connect the symbolic cognitive system (Σ) with its corresponding sensorimotor perceptual system (Π). Figure 9.2 shows the anchoring process of the "BurgerKing" landmark that connects its symbolic representation with its perceptual region in the captured image. The conceptual space is a metric space whose dimensions, called qualities [48], are related with the quantities processed by the robot sensors. Points in a conceptual space, called knoxels [21], represent the epistemologically primitive elements at the considered level of analysis. For logo landmarks (such as BurgerKing), the conceptual space represents three different quantities: the correlation values to the hue component of the stored color histograms, landmark shape and size range, and the number of the matched SIFT points. Table 9.1 shows properties and attributes values of some landmarks from the stored landmark database.

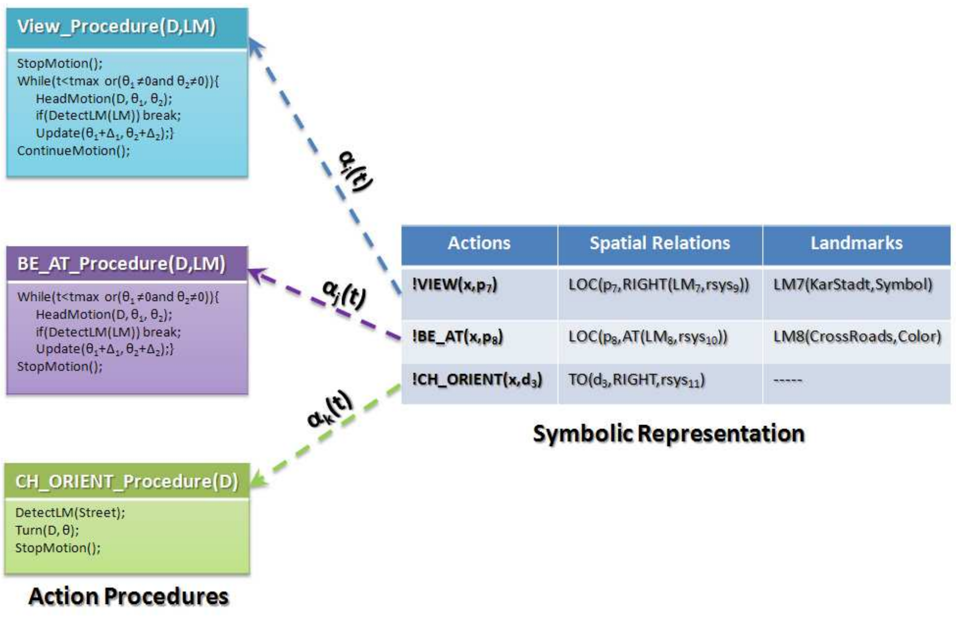
Property	Attribute	Value
BKColor	Color Histogram	{70, 54, 57, 52, 41, 27, 50, 69, 169, 200, 23, 40, 26, 23, 15, 27}
BKShape	Rectangle	[2000–70000]
BKRecognition	SIFT	Feature List
CRColor	Color	White
CRShape	Line	Dashed
CRRecognition	Hough	Line
THColor1	Color	Green
THColor2	Color	Yellow
THShape1	Rectangle	Big
THShape2	Rectangle	Small

Table 9.1: Database table of the landmark properties.

On the one hand, the symbolic system manipulates individual symbols for each landmark to denote its physical object. The predicate grounding function (g) associates each individual symbol with a set of symbolic predicates that assert properties of the corresponding landmark. It associates unary predicates in Σ to areas in the conceptual space. In other words, the g function gives semantics to symbolic predicates in terms of observable quantities in the conceptual space. On the other hand, the perceptual system generates percepts from the observation of physical landmarks that are represented as regions in the captured images. The sensor model function (h) associates each percept with its observed values of a set of measurable attributes. It transforms a measurement vector from the sensor system into a set of knoxels in the conceptual space.

Therefore, the correspondence between symbols and percepts is reified in a data structure called anchor ($\alpha(t)$) that contains pointers to the corresponding symbols (σ_i) and percepts (π_i). In addition to these pointers, it has a pointer to an estimate of the current values of some attributes of the landmark which it refers to. This pointer is called the signature and denoted by γ_i which indicates its corresponding knoxels in the conceptual space. An anchor can be considered as a model of a physical object that reflects the persistence of the object, and which can be shared across different subsystems of the agent. Once an anchor has been created, it should be continuously updated to account for changes in the landmark's attributes and handle the connections of this landmark with its neighbor landmarks in the route description. This connection is made depending on the relationships that are retrieved from the processed route to handle the uncertainty during robot navigation.

We extended the anchoring process to handle not only the landmarks but also robot actions. The anchoring process is used to connect the symbolic representation of the robot locomotion actions to their corresponding dynamic procedures which are controlled by both path and footstep planners. Figure 9.3 shows symbolic representations of some robot actions and their corresponding dynamic procedures.



```
StopMotion();
While(t<tmax or(θ₁≠0and θ₂≠0)){
    HeadMotion(D, θ₁, θ₂);
    if(DetectLM(LM)) break;
    Update(θ₁+Δ₁, θ₂+Δ₂);}
ContinueMotion();
```

```
While(t<tmax or(θ₁≠0and θ₂≠0)){
    HeadMotion(D, θ₁, θ₂);
    if(DetectLM(LM)) break;
    Update(θ₁+Δ₁, θ₂+Δ₂);}
StopMotion();
```

```
DetectLM(Street);
Turn(D, θ);
StopMotion();
```

Actions	Spatial Relations	Landmarks
!VIEW(x,p_7)	LOC(p_7,RIGHT(LM_7,$rsys_9$))	LM7(KarStadt,Symbol)
!BE_AT(x,p_8)	LOC(p_8,AT(LM_8,$rsys_{10}$))	LM8(CrossRoads,Color)
!CH_ORIENT(x,d_3)	TO(d_3,RIGHT,$rsys_{11}$)	-----

Figure 9.3: The symbolic and dynamic procedures of some robot locomotion actions.

9.3 Head-motion Planner

In real and unknown environments, usually motion planners only focus on how to find an optimal path to the destination, but it is also important to decide on where to explore and look in order to accomplish finding a path to the goal. Therefore, when the humanoid robot is following a path, its head should be moved according to the situation presented in the initial path estimation to detect and localize the landmarks. As a result, the motion planner should generate walk and head-motion commands and send them to the motion trajectory generator to execute them.

Accordingly, we implemented the head-motion planner to direct the robot's head to a suitable angle with respect to the landmark position in the processed route. The head-motion planner is used to plan the movement of the robot's head depending on the direction of landmarks in the estimated

Head Direction	Angle Range
UP	$0\,^\circ$ – $45\,^\circ$
DOWN	$-15\,^\circ$ – $0\,^\circ$
LEFT	$0\,^\circ$ – $45\,^\circ$
RIGHT	$-45\,^\circ$ – $0\,^\circ$

Table 9.2: The ranges of head movement for HOAP-2 humanoid robot.

initial path and the movement ranges of the neck motors. Table 9.2 shows the actual range of the neck motors for the HOAP-2 humanoid robot. The robot will tilt its head to the right or left to detect the landmarks which are located at the road sides. It also looks down to the floor to detect landmarks such as crossroads and street boundaries in the miniature city. The head orientations of the humanoid robot are divided into four actions: turn right, turn left, move up, and move down. The humanoid robot can turn its head by $45\,^\circ$ in both the right and left directions. It can also raise its head by $45\,^\circ$ and lower it by $15\,^\circ$.

Algorithm 3 shows the used technique in the head-motion planner. First, it checks the direction of the landmark from the estimated path. Then, the robot changes its head direction depending on the location of the processed landmark. If the robot fails to detect the landmark, the planner changes its head direction by $15\,^\circ$. It processes again until it detects the landmark or terminates if the angle of the robot's head equals $0\,^\circ$.

9.4 Collision Detection

Collision detection is considered to be one of the crucial factors in motion planning. For humanoid robots, there is an effective and simple way to detect collision by choosing an appropriate bounding volume approximating the shape of the robot. A trajectory for the body-center of a humanoid robot is computed by approximating its shape by using a cylinder surrounding its body as shown in Figure 9.4. A cylinder model is useful during humanoid robot turns and lateral walking to calculate the actual processing space required to execute robot actions. As the positions of the nearest obstacles to the robot are calculated by using triangulation, a cylinder model of the humanoid robot can be checked for obstacle avoidance in a constant time. Simply, if the distance between the robot and the obstacle is known, then it will be compared to the radius of the cylinder model.

The cylinder is a tight fit to the shape of the humanoid robot standing still. When the robot moves, however, additional space is needed for the transition in C-space as body and legs are swinging while walking. Figure 9.5 shows snapshots of the robot when moving forward, turning right and stepping sideways. We account for the additional space by enlarging the

Algorithm 3: Head-motion planner

```
input  : $LM \leftarrow$ the processed landmark
         $D_{LM} \leftarrow$ the direction of the processed landmark
output : $\theta_h \leftarrow$ the horizontal angle of the robot's head
         $\theta_v \leftarrow$ the vertical angle of the robot's head

$\triangle = 15$;
if $D_{LM} = Left$ then
    $\theta_h \leftarrow 45$ ;
    $\theta_v \leftarrow 0$ ;
    repeat
        LandmarkDetection(LM);
        $\theta_h \leftarrow \theta_h - \triangle$;
    until (LandmarkDetection(LM) $\neq$ NIL OR $\theta_h = 0$) ;
    if (LandmarkDetection(LM) = NIL) then return failure;
else if $D_{LM} = Right$ then
    $\theta_h \leftarrow -45$ ;
    $\theta_v \leftarrow 0$ ;
    repeat
        LandmarkDetection(LM);
        $\theta_h \leftarrow \theta_h + \triangle$;
    until (LandmarkDetection(LM) $\neq$ NIL OR $\theta_h = 0$) ;
    if (LandmarkDetection(LM) = NIL) then return failure;
else
    $\theta_h \leftarrow 0$ ;
    $\theta_v \leftarrow -15$ ;
    if (LandmarkDetection(LM) = NIL) then RobotBowing();
    if (LandmarkDetection(LM) = NIL) then return failure;
```

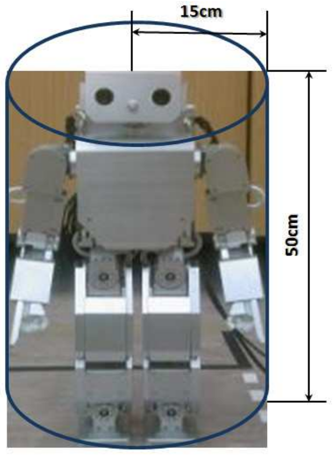


Figure 9.4: HOAP-2 humanoid robot cylinder model.

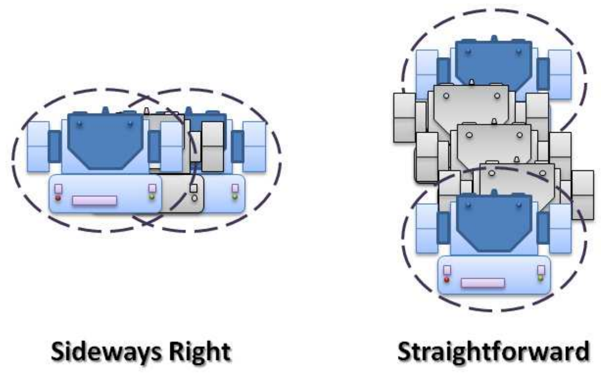


Figure 9.5: Different motion actions of the humanoid robot by using a cylinder model.

cylinder at the start and end configurations depending on the action which will be processed by the robot. When the robot walks sideways, no additional space is needed. If the robot walks straightforward, it needs additional space with respect to the swinging of its legs while walking. Therefore, the obstacle distance is compared to the enlarged cylinder radius plus the expected step distance. Otherwise, as humanoid robots are non-holonomic, they cannot turn in place without requiring additional space. For turns, the humanoid robot wants extra turning space in a cylinder model, and then the cylinder model is enlarged by twice the turn radius to let the humanoid robot turn in a specific direction without collision. Therefore, such an approximation enables a humanoid robot to find paths in real-time and include actions such as walking sideways through a narrow space.

9.5 Path Planner

The path planner can be considered as the core stage of our motion planner. It generates the shortest roadmap graph (G) of the robot's path. It only returns the path graph, not the ability to execute that path. In our system, the path planner only depends on both the processed route description, as initial path estimation, and the retrieved data from vision. As the robot navigates in an unknown environment, the path planner processes the path as segments from the generated topological map (see Chapter 7). Each segment represents the distance between two adjacent landmarks in the robot's estimated path. The planner processes each segment as an independent path with its own start and end points. It is continuously evaluating the current C_{free} and sends the resulting roadmap graph (G) of the processed path segment to a footstep planner which computes the next footsteps of the humanoid robot. The resulting footstep placements and robot actions are fed to the robot's motion trajectory generator to execute the motion of the humanoid robot.

As the sampling-based motion planning algorithms present practical and efficient solutions for the motion planning problem, they are extensively applied to many problems in high-dimensional configuration spaces. Therefore, we implemented our path planner by using a modified version of the Probabilistic RoadMap planner (PRM) [92, 88]. We applied the lazy evaluation [6, 25] to the PRM planner with non-uniform sampling to handle narrow passages. The main theme of the planner is to minimize the number of collision-checks performed during planning. By avoiding local planning and instead keeping the global view, only the part of C-space that is essential in answering a query is explored.

The path planner minimizes the running time by reducing the number of collision checks performed during planning. It initially assumes that all nodes and edges in the roadmap are collision-free, and searches the roadmap for the shortest path between the start and goal nodes. The nodes and edges along the path are then checked for collision. If a collision with an obstacle occurs, the corresponding nodes and edges are removed from G. It updates the roadmap with new nodes and edges, and then searches for the shortest path. The above process is repeated until a collision-free path is returned. To avoid bad estimations for the path planner, we used limited time for processing and generating G.

Consequently, the planner is tailored to efficiently answer single planning queries, but can also be used for multiple queries. Algorithm 4 illustrates the functionality of the proposed path planner and is explained in the next subsections.

9.5.1 Grid Generation

At the beginning, a low resolution regular grid is applied to C-space to generate a search grid. The C-space is divided into small cells of 7.0 cm (width) x 5.0 cm (height). Each cell has an associated location in the grid (x, y) and an information value. The grid generation will help in maintaining the connectivity of the graph by defining a minimum discretization for the open spaces. The discretization density is adjusted to suit the environment, selecting as sparse a grid as possible. Up to this stage, the cells hold only position information.

9.5.2 Sampling Process

A crucial ingredient of the path planner is a sampling algorithm. Its aim is to minimize the on-line computation by pre-generating a search space to contain all the information that will be used during the on-line path planning, while at the same time avoiding the generation of an unnecessarily complete and complex space. It samples the C_{free} by using a non-uniform

Algorithm 4: The proposed path planner

```
input  : n← the number of nodes in the roadmap
         k← the number of the closest neighbors to c
output: A roadmap $G = (V, E)$
         $P \leftarrow$ a path from $c_{init}$ to $c_{goal}$

foreach $S_i \in$ Initial Path Estimation do
    $V \leftarrow \{\}$, $E \leftarrow \{\}$;
    $c_{init} \leftarrow$ the initial configuration of $S_i$;
    $c_{goal} \leftarrow$ the goal configuration of $S_i$;
    $V \leftarrow V \cup \{c_{init}\}$;
    GridGeneration();
    repeat
        $c \leftarrow$ a random configuration in C-space;
        if $c \in C_{obst}$ then
            $\acute{c} \leftarrow$ a random configuration in C-space;
            if $\acute{c} \in C_{obst}$ then
                $c_m \leftarrow$ the midpoint of $c\acute{c}$ line segment;
                if $c_m \in C_{free}$ then $V \leftarrow V \cup \{c_m\}$;
        else
            $V \leftarrow V \cup \{c\}$;
    until $|V| > n$ ;
    $V \leftarrow V \cup \{c_{goal}\}$;
    forall $c \in V$ do
        $N_c \leftarrow$ the k nodes of c from V;
        forall $\acute{c} \in N_c$ do
            $c_s \leftarrow$ closest neighbor of c from $N_c$ in $c_{goal}$ direction;
            if $(c, c_s) \notin E$ AND $\triangle\ (c, c_s) \neq NIL$ then
                $E \leftarrow E \cup \{(c, c_s)\}$;
                break;
            else
                $N_c \leftarrow N_c \setminus \{c_s\}$;
    $P \leftarrow$ shortest path $(c_{init}, c_{goal}, G)$;
    if $P \notin \emptyset$ AND $P \in C_{free}$ then
        return P;
    else if $t < t_{max}$ then
        PathEnhancement();
    else
        return failure;
```

approach returning with the minimal free search space to avoid the computational complexity of generating a denser search area. The C-space is sampled by using the bridge test approach [58] that was introduced to boost the sampling density inside narrow passages using only a simple test of the local geometry. The idea is to take two random samples, where the distance between the samples is chosen according to Gaussian distribution. Only if both samples lie in C_{obst} and the middle point between them lies in C_{free}, the free sample is added. Increasing the density of sampling around narrow passages increases the chances of finding samples in areas that are hard to reach and are likely to be needed for finding a solution. On the other hand, the samples in open space are randomly chosen in the medial of C-space with lower density.

9.5.3 Roadmap Generation

The purpose of the path planner is to build a roadmap graph (G) of a feasible path. The idea is to lazily evaluate the feasibility of the roadmap as planning queries are processed. The c_{init}, c_{goal}, and a number of non-uniformly distributed samples are used to form nodes in a roadmap. They are connected by edges in which each pair of nodes is sufficiently close together and in c_{goal} direction. In other words, each node in G is connected by edges to a set of neighbor nodes. An edge represents the straight line path in C-space between two nodes.

The second step in the algorithm is to find the shortest path in G between c_{init} and c_{goal}. Given a procedure that estimates the length of a path, the shortest feasible path in the roadmap is found by repeatedly searching for the shortest path by using D* Lite search. Therefore, the path is checked to know if it is collision-free or not. If the resulting path lies in C_{free}, it will be supplied with the grid of cells of the C-space to the footstep planner to retrieve the actual footstep placements for the humanoid robot.

On the other hand, if no path exists in the roadmap, the planner either reports failure or goes to the path enhancement stage to add more nodes to the roadmap and start searching again. The choice is determined by the overall time allowed to solve the problem. If the planner reports an occurrence of a collision, the corresponding node or edge from the roadmap is removed. Then, the planner adds new nodes and edges and searches again for the new shortest path.

9.6 Footstep Planner

As discussed in Chapter 5, many researchers have concentrated on various approaches to generate reliable and stable gaits with feedback, and also on developing global navigation autonomy for humanoid robots. Emphasis has primarily been laid on pre-generating walking trajectories, online trajectory generation, and dynamic balance, without accounting for obstacles. Most of them use one of the two main approaches of bipedal locomotion: static or dynamic [122]. The main objective, in any case, is to produce a gait as natural and stable as possible. Static walkers rely on the static equilibrium condition: maintain the CoG on the convex hull within the contact area with the ground. This approach denies inertial forces. Therefore, it can be applied only if robot movements are very slow. Dynamic walkers achieve fast and natural walking motion following the principle of dynamic equilibrium: they use ZMP [155] instead of CoG, so that inertia components and gravity are considered.

The footstep planner is a high-level planner implemented to calculate footstep placements under humanoid robot stability and motion constraints. It ignores as much of the underlying details of leg movement and trajectory generation as possible, and works instead from a description of the robot's capabilities. By using the roadmap graph as a reference path, it returns a sequence of footholds that the robot can reach carrying it from the initial to the goal location.

The footstep planner takes the roadmap graph (G) of the current segment, the initial and goal points, and the grid of feasible cells as inputs. It returns the solution as an ordered list of the footstep placements that should be executed to reach the goal position. The D* Lite search [72, 97] is used to determine repeatedly the shortest paths between the current footstep of the humanoid robot and the goal location as the edge costs of a graph change while the robot moves towards the goal position. D* Lite search provides accurate and fast solutions for humanoid robot motion in dynamic and unknown environments. It generates the shortest and the lowest-cost sequence of footstep locations to reach the target point. D* Lite works by exploring grid nodes (cells) that are provided by the path planner and calculates the cost function $F(n)$ for each cell in the roadmap graph. The cost function is calculated as the sum of the following three costs:

Step costs ($G(n)$ & rhs(n)): They are the costs of making the desired step from the start node to the current node (n) (for more details see Appendix C). Figure 9.6 shows two examples of the cost grid that is used in calculating the step costs. The grid is an eight-connected grid whose edge costs are initially one. The value of the cell is changed to infinity when the robot discovers that this cell cannot be traversed.

Estimated heuristic cost $H(n)$: It is the estimated cost from the current node (n) to the goal node. It uses a heuristic search to estimate the cost of the goal node and it minimizes the cost of the path so far. D* Lite search is optimal if the estimated cost to the goal is always underestimated. Since the shortest distance between two points is a straight line, Euclidean distance serves as a very accurately estimated cost to the goal, making D* Lite well-suited for fast computations.

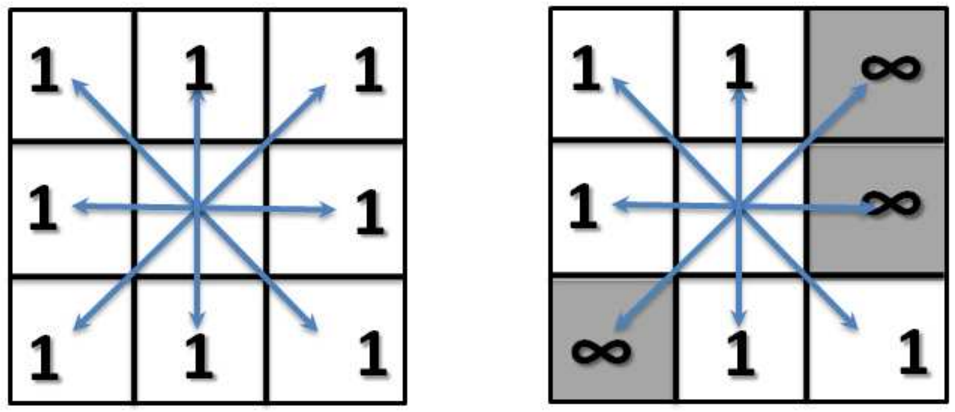


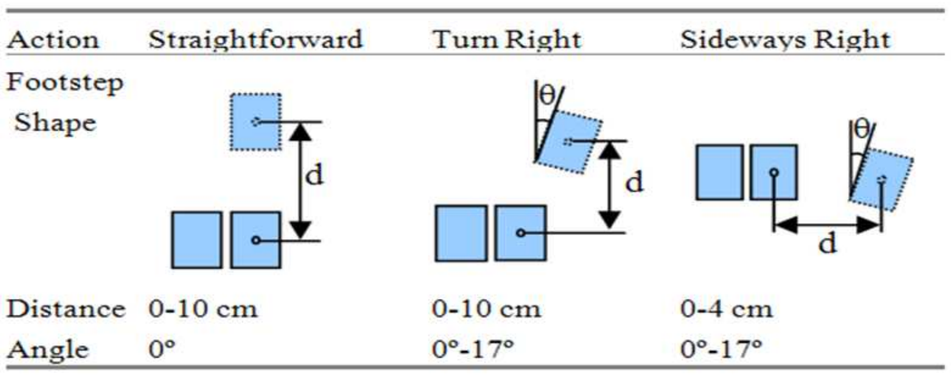
Figure 9.6: Examples of eight-connected grids.

Clearance cost $C(n)$**:** It is used to insure that the generated footsteps are directed to the middle of C_{free} and are not adjacent to the obstacles. It calculates the clearance cost of following the roadmap graph (G) that is generated from the path planner.

After the cost functions have been calculated, the planner computes the optimal sequence of footstep locations to reach the desired goal. The robot actions are modeled by storing a symmetric collection of candidate footstep transitions for both feet. A sequence of footstep placements to reach a goal in the route environment is computed from a discrete set of feasible footstep locations corresponding to stable candidate stepping motion trajectories. The planner returns the solution as an ordered list of the footsteps which should be processed to reach the goal. To achieve smooth walking, the parameters of the next step must be known before the current step of the robot ends.

D* Lite search can efficiently recalculate the shortest path from the current position of the robot to the goal position. It only recalculates those goal distances that have been changed or have not been calculated before. It achieves a speed up of one to two orders of magnitude over repeated A* search by modifying previous search results locally. On the other hand, the footstep search can fail in one of the following situations:

- These is no roadmap graph applied to the footstep planner.
- No more valid successor nodes can be generated. In this case, no collision-free footstep sequence exists using the given discrete set of relative footstep placements.
- The running time of the search exceeds the maximum allowable time. In this case, the planner fails to reach the goal, but the lowest cost path computed so far can be returned.



Action	Straightforward	Turn Right	Sideways Right
Footstep Shape			
Distance	0-10 cm	0-10 cm	0-4 cm
Angle	0°	0°-17°	0°-17°

Figure 9.7: Footstep placements for HOAP-2.

9.7 Motion Trajectory Generator

After estimating the footstep placements and the head movements for the humanoid robot, they are submitted to the motion trajectory generator to execute the desired actions. The motion trajectory generator has three fundamental functions: (i) keeping the humanoid robot in balance (static or dynamic stability), (ii) moving the swing leg, and (iii) controlling visual attention. The motion trajectory generator calculates the walking parameters and sends them to the robot's actuators to execute these actions.

We considered the walking process of the humanoid robot as a symmetric, periodic and smooth motion. The motion trajectory generator is used in order to output a final dynamically-stable trajectory. We use the ZMP trajectory in order to maintain overall dynamic stability. The ZMP walking pattern is used to produce humanoid robot gaits as dynamic and stable as possible. The foot placement actions indicate the motion, turns, and change of orientation actions. These actions are divided into six footstep placement actions for the humanoid robot: straightforward, straight backward, turn right, turn left, sideways right, and sideways left. All of these actions have two parameters which have real values. Figure 9.7 shows some foot placement actions and their parameters for the HOAP-2 humanoid robot. It is worth noting that the humanoid robot does not necessarily need to be able to exactly perform these six actions. For example, the robot could well use several footsteps for performing the 45 ° rotation in turn actions.

On the other hand, the robot's head is moved depending on the direction of the processed landmarks and the movement range of the neck's motors. The robot will tilt its head to the right and the left to detect the landmarks which are located at the road sides. It also looks down to the floor to detect landmarks such as crossroads and street boundaries in the miniature city. The head orientations of the humanoid robot are divided into four actions: turn right, turn left, move up, and move down. Table 9.2 shows the actual

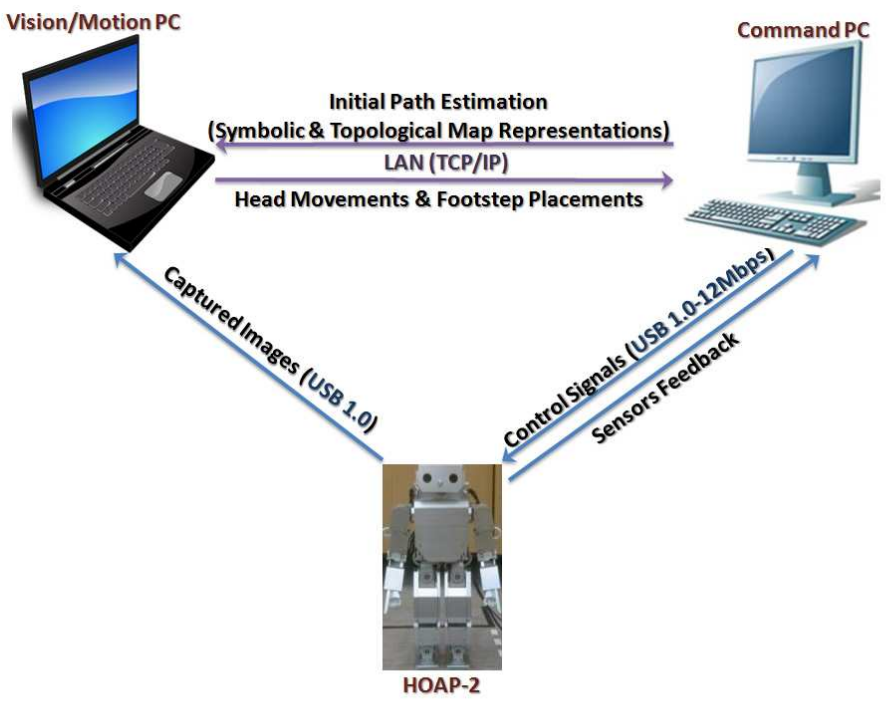

Vision/Motion PC
Command PC
Initial Path Estimation
(Symbolic & Topological Map Representations)
LAN (TCP/IP)
Head Movements & Footstep Placements
Captured Images (USB 1.0)
Control Signals (USB 1.0-12Mbps)
Sensors Feedback
HOAP-2

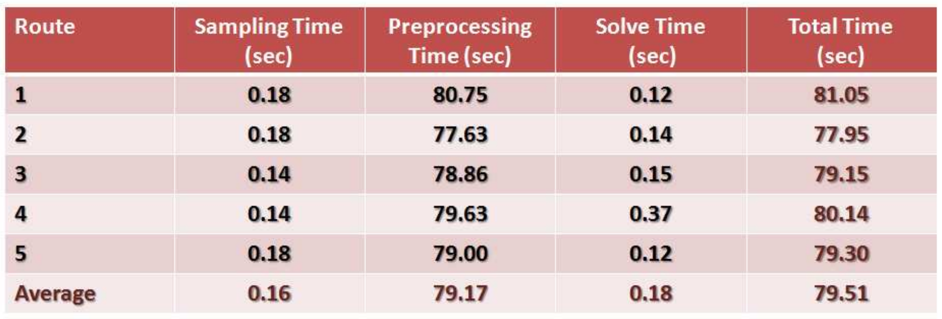
Figure 9.8: The hardware components of the humanoid robot navigation system.

ranges of head motion actions for the HOAP-2 humanoid robot.

9.8 Experimental Results

As mentioned previously, our humanoid robot navigation system is run on the Fujitsu HOAP-2 humanoid robot. The humanoid robot motion planner is implemented on an Intel 1.73 GHZ Duo laptop with 3 GB RAM running Linux. This computer is connected to the robot's command PC which is a 1.8 GHz Pentium III PC with 256 MB RAM running openSUSE Linux 9.0 with RT-Kernel. Figure 9.8 shows the hardware components of our humanoid robot navigation system. The command and vision/motion computers exchange data via a 100 MBit/s LAN based on TCP/IP. The robot's cameras and control are connected to the vision/motion and command computers via USB 1.0 connections with 12 Mbps, respectively. The command PC coordinates data transfer to and from the vision/motion computer. It also sends the calculated footstep placements and head movements to the robot's actuators for execution.

We have implemented the humanoid robot motion planner in C/C++ programming language. To test the performance of our proposed algorithm,



Route	Sampling Time (sec)	Preprocessing Time (sec)	Solve Time (sec)	Total Time (sec)
1	0.18	80.75	0.12	81.05
2	0.18	77.63	0.14	77.95
3	0.14	78.86	0.15	79.15
4	0.14	79.63	0.37	80.14
5	0.18	79.00	0.12	79.30
Average	0.16	79.17	0.18	79.51

Table 9.3: Total consumed time in the path planning phase for the tested routes using OOPSMP.

we used the Object-Oriented Programming System for Motion Planning (OOPSMP). OOPSMP is developed at the Physical and Biological Computing Group at Rice University [115]. It is an open-source programming infrastructure that provides implementations of various existing algorithms in a modular, object-oriented fashion that is easy to extend. We integrated our proposed motion planning algorithm in OOPSMP to test its performance and calculate the consumed time in the sampling, preprocessing, and solving stages.

To test the performance of the proposed path planner, we chose five different routes in the miniature city to retrieve the shortest roadmap graph for each one. Figure 9.9 illustrates the resulting roadmap graphs of the feasible paths for the five tested routes. We applied the proposed path planner without any previous information about the user's route descriptions. The path planner calculates the shortest feasible path for the robot from the start position to the destination by using non-uniform sampling. We substitute the shape of the robot by a hexagon with 15 cm radius and 50 cm height. Table 9.3 shows the total consumed time for the sampling, preprocessing, and solving phases of the path planner for each route. The number of neighbors to sample and the number of neighbors to query are chosen to be 15 and 50, respectively.

To test the proposed motion planner, the experiments were performed on an evenly flat surface in the miniature city with static obstacles. We made some simplifying assumptions while conducting the experiments. First, the environment floor is flat and contains non-moving obstacles. Second, a discrete set of feasible footstep placement positions and head movements are computed for the current processing path segment during the robot navigation.

In our system, the time required for finding a path on the resulting C_{free} depends on both the route description supplied by the user and the path length. As mentioned previously, the path is processed as segments. Each

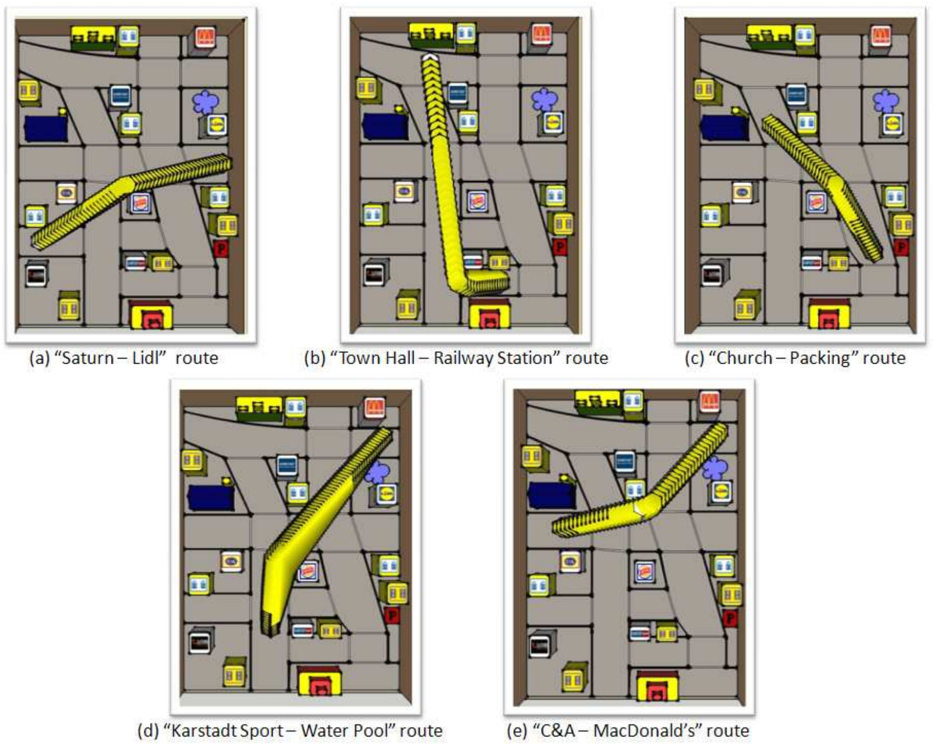

(a) "Saturn – Lidl" route
(b) "Town Hall – Railway Station" route
(c) "Church – Packing" route
(d) "Karstadt Sport – Water Pool" route
(e) "C&A – MacDonald's" route

Figure 9.9: The resulting roadmap graphs of the tested routes using OOPSMP.

segment has its own start and goal points. The number of neighbors to sample and the number of neighbors to query are chosen to be 5 and 10, respectively. The average time for planning a path for a segment in the route with 1.0-1.5 m is approximately 1.4 sec. The footstep planner is supplied by the retrieved roadmap graph of the current path segment and the data of all feasible grid cells from the path planner. The average consumed time to calculate one footstep placement is approximately 50 msec.

Figure 9.10 shows some snapshots of the HOAP-2 humanoid robot while executing the "Saturn–Lidl" route in the miniature city. First, the robot turns its head to the right to detect the Saturn store, and then it tilts its head down to detect the crossroads. It calculates the path for this segment and the feasible footstep placement to execute. After reaching crossroads, the robot turns to the left and then detects the crossroads, the C&A store, and the Burger King Restaurant. It walks to the next crossroads and turns to the right. Finally, HOAP-2 detects the crossroads and walks until it reaches the Lidl supermarket.

9.9 Discussion

In this chapter, we presented the anatomy of our proposed motion planning system for a Fujitsu HOAP-2 humanoid robot. The proposed technique is a combination of a sampling-based planner and a D* Lite search to generate fast and dynamic footstep placements for the humanoid robot in unknown environments. The robot navigation is based on the route described by the user to generate initial path estimation to the navigation task. The humanoid robot begins from the start point and moves along that path to collect information and recognize the landmarks by using its stereo vision and implemented techniques of landmark recognition. Based on the new findings and the processed route, the path is then re-planned to adjust the robot's position during navigation. The planner operates at the level of footsteps and it ignores the lower-level details of leg movements and control.

The main task of the proposed motion planner is to find a sequence of actions as close to optimal as possible that causes the robot to reach its goal location while avoiding the obstacles in an unknown environment. The symbol grounding phase is used to connect the symbolic representations of landmarks and locomotion actions to their equivalent perceptual landmarks and motion procedures, respectively. The output of the symbol grounding stage is provided to the head motion planner to plan the motion of the robot's head with respect to the positions of the landmarks. It also supplied to the path planner to generate the shortest feasible roadmap graph of the robot's path.

The path planner is implemented to extract the minimal feasible C_{free} and generate the shortest path to the target position. We used a modified

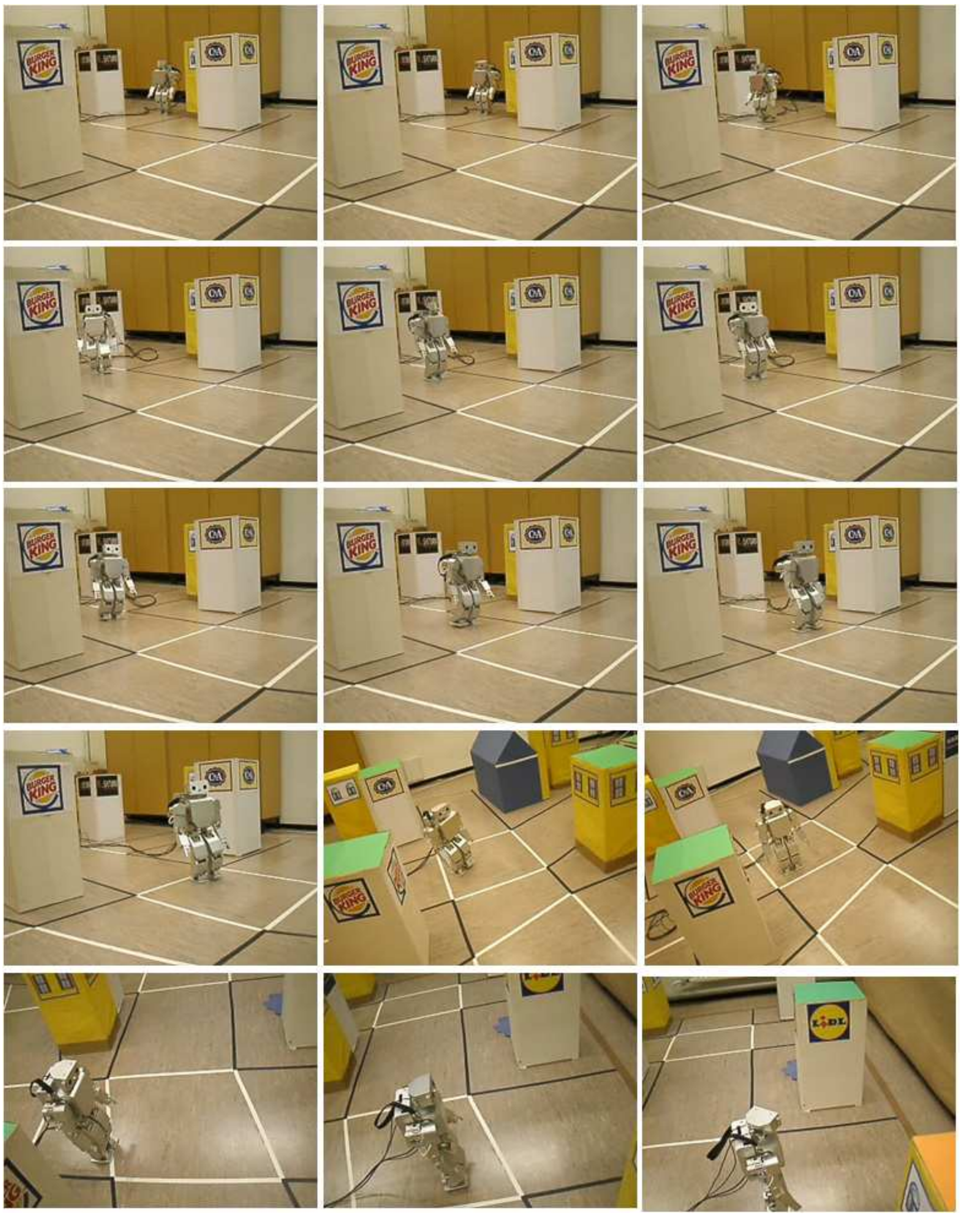


Figure 9.10: HOAP-2 executing a route from the Saturn store to the Lidl supermarket in the miniature city.

version of the Lazy-PRM technique with a non-uniform sampling to avoid the computational complexity of generating a denser search area. The planner directs the computational resources to troubled and difficult regions, such as narrow passages, leaving out the larger sparsely populated open spaces. A smoothing penalty is also associated to the nodes to encourage the generation of gentle paths along the middle of the empty spaces. The sampling is increased in complex areas and leaves out simple areas with lower resolution density.

For collision detection, a cylinder model is used to approximate the trajectory for the body center of the humanoid robot during navigation. It calculates the actual areas required to execute different motion actions of the humanoid robot and compares them with the distances to the nearest obstacles. Collision detection is carried out off-line during the creation of C-space to speed up the actual search for the path.

The footstep planner is implemented to find smooth and low-cost footstep placements of the humanoid robot within the resulting C_{free}. It uses D* Lite search to reduce searching time and produces a smoother dynamic path for the humanoid robot at a low cost.

Finally, we tested our proposed path planner algorithm by using the Object-Oriented Programming System for Motion Planning (OOPSMP). We tested five different routes in the 3D model of the miniature city and we measured the consumed time in each processing stage. We also measured the processing time for the footstep planner.

CHAPTER 10 Conclusion

In mobile robotics, natural language interaction is considered as a challenging problem, not only because it requires sophisticated speech recognition and language understanding, but also because it inevitably includes issues of mixed-initiative interaction, multimodal interaction, and cognitive modeling. Natural language can express rules and sequences of commands in a very concise way. It uses symbols and syntactic rules to interact with robots that have knowledge represented at the symbolic level. Such symbolic communication can help robots to learn faster when they learn at the sensory–motor association level. Spatial reasoning gives robots the ability to use human-like spatial language and provides the human user with an intuitive interface that is consistent with his innate spatial cognition. It can also accelerate learning by using symbolic communication. A robot capable of understanding spatial language could be controlled by a novice user naturally to perform complex tasks using succinct, intuitive commands.

On the other hand, the problem of autonomous robot navigation is one of the most challenging tasks in robotics. The applications of mobile robot navigation to unknown and dynamic indoor environments are receiving more and more attention. The set of navigational strategies found in mobile robots mirrors the complexity of navigational strategies employed by biological organisms. Autonomous navigation has always been an interdisciplinary topic of research. It combines many fields of research to produce a feasible navigation system for a mobile robot in its surrounding environment.

In general, robot navigation can be processed in four basic steps. First, the robot should perceive its environment by using its sensors, such as a laser range finder, IR, and stereo cameras. Second, it builds a digital representation of the environment and represents the navigation task in an abstract form with starting and ending positions. Third, it extracts the landmarks from the environment in real-time. Finally, the robot specifies motion paths and locations of landmarks during navigation.

Therefore, we can summarize the major sources of difficulties in autonomous humanoid robot navigation in the following points. First, how can the user describe a navigation task for the robot in a simple and easy way without any ambiguities and misunderstandings during the description process? Second, a light-weight, robust object classification technique should be chosen to recognize different types of landmarks during robot navigation in real-time. The chosen technique should handle the object recognition problems in real scenes, such as viewpoint variation, occlusions, and illumination changes. Third, the high-level cognitive processes and their equivalent sensorimotor processes should be integrated at different levels of abstraction. Forth, how can the shortest, feasible path for the robot be calculated, under consideration of robot dynamic and control constraints? Finally, we have to determine the footstep placements for the humanoid robot in a dynamic way to handle different situations in unknown environments.

The work reported in this thesis is meant to advance the state of the art in the field of autonomous humanoid robot navigation. This dissertation has focused on the problem of autonomous robot navigation in unknown indoor environments. It concentrates on investigating a framework which could combine and extend the existing technology by accommodating new algorithms and techniques to achieve autonomous navigation for mobile robots in indoor environments. Therefore, the research work in this thesis aims to address the autonomous navigation problem and contribute to the development of practical humanoid robot navigation systems. The major contribution of this dissertation is to develop a complete humanoid robot navigation system in unknown environments based on a cognitive multimodal interface which can handle many problems efficiently. We can summarize our contributions into four points: building a multimodal cognitive interface for robot navigation; implementing an instruction interpreter to create abstract representations for the route; developing a robot landmark processing system to detect, localize, and classify different types of landmarks; and implementing a dynamic motion planning technique for humanoid robots.

For route description, we proposed a multimodal interface which can be used easily by inexpert users to describe navigation tasks for mobile robots. The routes can be described to the robot verbally or graphically. Our proposed route instruction language (RIL) is intended as a semi-formal language for instructing the robot to execute a route which is used via a structured graphical user interface. We conducted some experiments to evaluate the routes which are written by using the RIL instructions. The results of the experiments confirmed that RIL is simple to learn and it is well-suited to describe the route in indoor environments. The GUI facilitates the route description and lets the novice user describe the routes easily without ambiguities and misunderstandings. On the other hand, we have found that most of the commands that participants choose can be classified or decomposed into the RIL categories, and by considering only such commands, we can replicate the paths with reasonable accuracy.

For route analysis, we implemented an instruction interpreter to process the route description and generate its equivalent symbolic and topological map representations. A topological map is generated to describe relationships among features of the environment in a more abstract form without any absolute reference system. It is mainly used to treat the ambiguity which can occur when the robot cannot recognize the current landmark. It handles these situations by considering the relations between the landmarks. Therefore, the topological map is used to comprehend the route instructions by building up a mental representation of the processed route. These high-level cognitive representations – i.e., symbolic and topological map representations – are supplied to other system components as an initial path estimation to guide the robot while it plans its navigation task.

For object recognition, we developed an online robot landmark processing system (RLPS) to detect, classify, and localize different types of landmarks during robot navigation. The RLPS is based on a two-step classification stage which is robust and invariant towards scaling and translations. It provides a good balance between fast processing time and high detection accuracy by combining the strengths of appearance-based and model-based object classification techniques. The experimental results showed that the RLPS is more powerful as it recognizes a wide range of landmarks and finds them even when there are landmarks of similar colors in the field of view. It efficiently handles landmarks with occlusions, viewpoint variances, and illumination changes.

For motion planning, we proposed a hybrid motion planner for a humanoid robot which is a combination of sampling-based planner and D* Lite search to generate dynamic footstep placements in unknown environments. The proposed planner calculates the shortest path in a reasonable time and it handles the re-planning process for the footstep time efficiently. A modified cylinder model is used to approximate the trajectory for the robot's body-center during navigation. It calculates the actual distances required to execute different actions of the robot and compare them to the distances from the nearest obstacles.

As can be seen, the navigation system presented in this research provides a feasible framework for a humanoid robot navigation system. However, this framework has some limitations. First, the floor of the environment is flat and contains static obstacles. We did not deal with uneven surfaces or moving obstacles. Second, the humanoid robot can avoid obstacles, but cannot step over or onto the objects. Third, we ran our system on a miniature city and we did not test it in any other indoor environment.

Although the framework that we have developed in this thesis has provided satisfactory theoretical and experimental results, our analysis has uncovered several issues that need further investigation. Future research could focus on the following points. First, building a memory model for the mobile robot to store the successfully executed routes and the retrieved parameters. These stored data can be used to handle future robot tasks more efficiently and reduce the processing time. Second, the question of how to achieve accurate object detection and classification of various types of objects in human environments with complex backgrounds, cluttered environments, and under different illuminations conditions. Third, dealing with different types of dynamic obstacles which can be found in human environments. Fourth, designing the footstep placements of the humanoid robot to handle stepping over and onto obstacles and designing special placements to deal with the narrow passage problem. Finally, operating the humanoid robot in the wireless mode and replacing the robot's cameras with other high-resolution cameras that can operate in the wireless mode.

APPENDIX A Symbols and Acronyms

Symbol	Description
f	The focal length of the camera
f_x	The product of the physical focal length of the lens and the size of the individual imager elements in the x direction
f_y	The product of the physical focal length of the lens and the size of the individual imager elements in the y direction
P_x, P_y	The pixel dimension
k	The radial lens distortion coefficient
C_x, C_y	The center of radial lens distortion
S_x	The scale factor to account for any uncertainty due to imperfections in hardware timing for scanning and digitization
O_x, O_y	The center of the image
α_x	The scale factor
T	The translation vector
T_x, T_y, T_z	The translation components for the transformation between the world and camera coordinates
R	The rotation matrix
R_x, R_y, R_z	The rotation angles for the transformation between the world and camera coordinates
P	The projection matrix
k_c	The correlation coefficient
Σ	The symbolic cognitive system
Π	The sensorimotor perceptual system
g	The predicate grounding function
h	The sensor model function
$\alpha(t)$	The anchor
γ_i	The anchor's signature

Symbol	**Description**
c	A configuration
c_{goal}	The goal configuration
c_{init}	The initial configuration
C_{free}	The free configuration space
C_{obst}	The obstacle configuration space
C-space	The configuration space
W	The workspace
$G = (V, E)$	A roadmap graph
k	The number of the closest neighbors
n	The current node
$F(n)$	The cost function
$C(n)$	The Clearance cost
$G(n)$	The step cost function
$H(n)$	The estimated cost function
rhs(n)	The one-step look ahead values based on the g-values

Acronyms	Description
ANNs	The Artificial Neural Networks
BoF	The Bag of Features Approach
CCHs	The Color Cooccurrence Histograms
CoG	Center of Gravity
CRIL	Conceptual Route Instruction Language
DOF	Degree of Freedom
DRRT	Dynamic RRT
DSP	Double Support Phase
ERRT	Execution-extended RRT
EST	Expensive-Spaces Tree
FSM	Finite-State Machine
FSRs	Force Sensing Registers
GA	A cognitive-oriented Geometric Agent
GUI	Graphical User Interface
HCI	Human–Computer Interaction
HOAP-2	The second generation of Fujitsu's Humanoid for Open Architecture Platform
HRI	Human–Robot Interaction
HRNS	Humanoid Robot Navigation System
HT	The Hough Transform
IBL	Instruction-Based Learning
ICS	Inevitable Collision State
LOA	Levels of Autonomy
LPA*	The Lifelong Planning A* Search Algorithm
OOPSMP	The Object-Oriented Programming System for Motion Planning
PPHT	The Progressive Probabilistic Hough Transform
PRM	Probabilistic Roadmap Planners
PSF	Performance Shaping Factors
RFCH	The Receptive Field Cooccurrence Histograms
RIL	Route Instruction Language
RL	Reinforcement Learning
RLPS	Robot Landmark Processing System
ROPs	Reactive-Odometric Plans
RPP	Randomized Path Planner
RRF	Reconfigurable Random Forest
RRTs	Rapidly-exploring Random Trees
SA	Situation Awareness
SIFT	The Scale Invariant Features Transform
SSP	Single Support Phase
SURF	Speeded Up Robust Features
UAVs	Unmanned/Uninhabited Air Vehicles
ZMP	Zero-Moment Point

APPENDIX B Symbol Grounding in Autonomous Robotics

In HRI, many of the resulting errors can be explained as failures of the grounding process, in which users and robots lack enough evidence to coordinate their distinct knowledge states. Understanding the grounding process provides not only a systematic framework to understand and improve HRI, but also a testbed to model the effects of different contexts and media upon language use.

For robotic systems embedded in the physical world, high-level cognitive processes and sensorimotor processes are typically incorporated to let the robot navigate autonomously. High-level cognitive processes perform abstract reasoning and generate plans for actions, whereas sensorimotor processes observe the physical world and execute actions in it. These processes have different ways to refer to physical objects in the environment. Cognitive processes typically use symbols to denote objects and actions. On the other hand, sensorimotor processes typically operate from sensor data that result from observing these objects. If the overall system has to successfully perform its tasks, it needs to make sure that these processes are successfully connected to indicate the same physical objects.

B.1 Symbol Grounding Problem

In everyday life, humans constantly use words to refer to objects in their physical world. These words are used as symbols to reflect specific references to other humans or agents. This procedure combines two different types of processes: one that reasons about abstract representations of objects, and one that has access to perceptual data. One of the prerequisites for the successful cooperation between these processes is that they agree about the objects they talk about, i.e., that there is a correspondence between the abstract representations and the perceptual data which refer to the same physical objects. In other words, there must be a correspondence between the names of things and their perceptual image. The problem of connecting linguistic descriptions of objects to their physical referents is known as the symbol grounding problem [50, 51].

On the other hand, the symbolic system can be defined as a set of symbols and rules for manipulating objects on the basis of their shapes (not their meanings). The symbols are systematically interpretable as having meanings, but their shape is arbitrary in relation to their meaning. The Symbol Grounding Problem is related to the problem of how symbols get their meanings, and of what the meanings are. The problem of meaning is in turn related to the problem of consciousness, or how it is that mental states are meaningful [51].

To ground symbols in their corresponding physical objects, the symbolic system should have the capacity to interact autonomously with non-symbolic sensorimotor capacities and handle the objects, events, properties and states of systematically interpretable symbols. It would have to be able to pick out the referents of its symbols, and its sensorimotor interactions with the world would have to fit coherently with the symbols' interpretations. Connectionism [50, 139] is one natural candidate for the mechanism that learns the invariant features underlying categorical representations, thereby connecting symbols to the proximal projections of the distal objects they stand for. In this way, connectionism can be seen as a complementary component in a hybrid non-symbolic/symbolic model of the mind, rather than a rival to purely symbolic modeling.

As a result, the symbol grounding problem is a major issue for computational models of language. Without the grounding of meanings in the world, symbols refer only to other symbols with no association between the symbols and the world. One way to address the symbol grounding problem in computational models of language is to conduct language research with real or simulated robots. For example, Schulz et al. [127] used symbol grounding to connect language statements to the vision in RatChat system. Their system aimed to evolve a shared lexicon between robots grounded in perceptions, local views, and behaviors using a language game framework. They tried to categorize and label the robot's internal representations with appropriate generalization and variability.

B.2 Symbol Grounding in Robotics

One way to extend robot language research is to use mobile robots that interact with a real world environment, using navigation systems to build up internal maps of the world. The use of mobile autonomous agents that move in a real environment enables the evolution of spatial languages using both relative and absolute frames of reference. The visual input of the robot would be used in a relative frame of reference, where the scenes can be categorized with respect to what the world looks like from the perspective of the robot. The internal maps would be used in an absolute frame of reference to indicate the physical objects. Consequently, autonomous robotic systems embedded in the physical world should typically incorporate two different types of processes. The first is high-level cognitive processes that perform abstract reasoning and generate plans for actions. The second is sensorimotor processes that observe the physical world and execute actions in it. The crucial observation here is that these processes have different ways of referring to the same physical objects in the environment. Cognitive processes typically use symbols to denote objects, while sensorimotor processes typically operate from sensor data that originate from observing these objects. If the overall system has to successfully perform its task, it needs to make sure that these processes point to the same physical objects, i.e., it has to perform anchoring between the symbolic representation and its corresponding physical objects [29, 66]. In other words, the autonomous robot that uses symbolic reasoning, sensing and acting in a real environment needs the ability to create and maintain the connection between symbols representing objects in the world and the corresponding perceptual representations given by its sensors. This connection has been named perceptual anchoring. Figure B.1 shows a graphical representation of the anchoring problem.

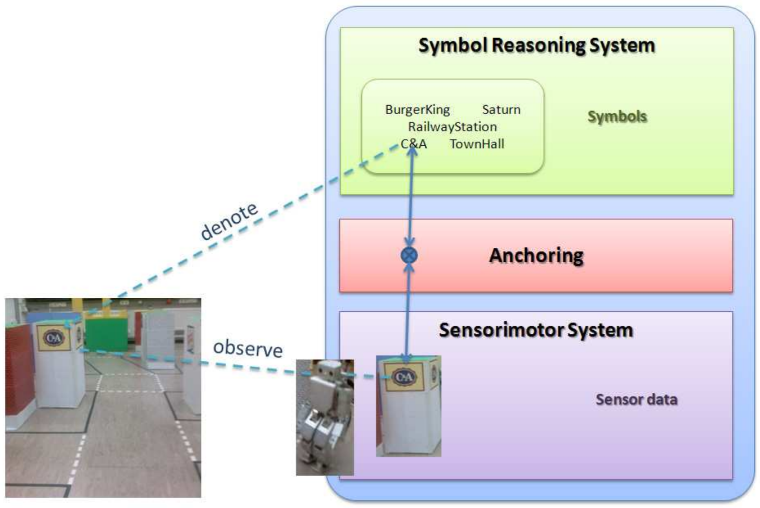


Figure B.1: Graphical illustration of the anchoring problem.

On the other hand, symbol grounding can be defined as the problem of finding a semantics for a symbolic system that is not in its turn a symbolic system [84]. Symbol grounding is a more general problem than anchoring. It concerns the philosophical issues related to the meaning of symbols in general. Anchoring is concerned with the practical problem of connecting symbols referring to physical objects to the sensor data originating from those physical objects in an implemented robotic system. In particular, anchoring focuses on perceivable physical objects, while symbol grounding needs to consider all kind of symbols. For some kinds of symbols it would be difficult to find appropriate sensor measurements, while the presence of sensor measurements is essential in anchoring [29].

From a more practical point of view, there are two research problems in the fields of robotics and AI which are related to the anchoring problem: pattern recognition and symbol grounding. Pattern recognition can be defined as the problem of interpreting data provided by sensors by assigning them to predefined categories [39, 137]. Taking pattern recognition in its most general sense, anchoring can be considered a sub-problem of pattern recognition. However, the anchoring problem emphasizes several peculiar aspects, which are not usually the focus of pattern recognition. First, the presence of symbols is an essential aspect of anchoring, while this is not the case in pattern recognition. Second, a goal of anchoring is the dynamic maintenance of the anchor in time, while pattern recognition is mostly used in applications where this dynamic aspect is not relevant. Finally, anchoring focuses on the creation and maintenance of the anchor as a shared representation to link several subsystems of the agent, such as motor control, sensor processing, and reasoning. Figure B.2 shows a simplified view of the relation among anchoring, symbol grounding and pattern recognition. Anchoring is included in the intersection between the other two problems and can be represented as a bridge between them. One can in fact find numerous cases of pattern recognition where no symbols are present, and one can study the symbol grounding problem without taking measurements in consideration. Anchoring by contrast implies the presence of both symbols and measurements and the possibility of establishing a connection between the two [29].

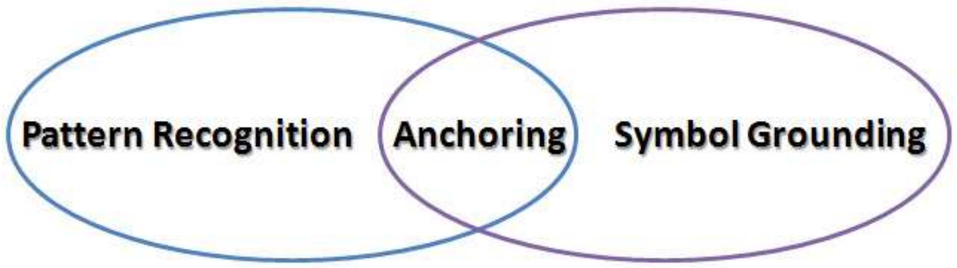


Figure B.2: Relations among anchoring, symbol grounding, and pattern recognition [29].

B.3 Perceptual Anchoring

Perceptual anchoring is one of the facets of the general problem of integrating symbolic and non-symbolic processes in an intelligent system. It is the problem of how to create and maintain in time the right correspondence between symbols and sensor data that refer to the same physical objects [21, 28]. It is considered as an important aspect of the connection between symbolic and sensory based processes in an autonomous robot. An example is the problem of connecting the symbol used by a planner to refer to an object needed for an action to the data that correspond to that object in the sensorimotor system. This connection must be dynamic since the same symbol must be associated to new entities in the perceptual stream in order to track the object over time or to re-acquire it at a later moment. Anchoring can be seen as an important special case of symbol grounding where the symbols denote individual physical objects. In perceptual anchoring, the symbol data correspondence for a specific object is represented by a data structure called an anchor [66]. The anchor includes pointers to the symbol and sensor data being connected together with a set of properties useful to re-identify the object, e.g., its color and position. These properties can also be used as input to the control routines.

From another perspective, if the anchoring module is considered as part of a larger cognitive system, it can be interpreted as part of the short term memory where perception and prior information, including the one used for prediction, are integrated over time. It also provides the necessary information for action and immediate decision. It interacts with the long term memory by accessing the prior information contained there [21].

When manipulating the anchoring process, some important aspects of anchors that should be considered. First, anchors can be shared across different subsystems of the agent in order to provide them with a common handle referring to a specific physical object [66]. Second, physical objects persist in time and space, and some of their properties are preserved across time or evolve in predictable ways. Therefore, the anchoring process must take this temporal dimension into account. Consequently, anchoring cannot be modeled as a one-shot process, but it must take into account the flow of continuously changing sensor input [29]. Finally, the matching between the symbolic description given by the planner and the attributes of precepts generated by the sensor system is needed to decide which precepts to use to create or update the anchor for a given symbol. Matching can be done partially or completely depending on the nature of processed object [84].

B.3.1 Mechanisms of Anchoring

Anchoring can be created using three main mechanisms: top-down, bottom-up, or in both directions simultaneously [21, 29]. The first mechanism is

Top-down, or goal-driven. This happens, for instance, when the symbolic system needs to anchor a symbol to perceptual data, e.g., in order to make an action executable. In other words, the symbolic system identifies the right object to be used for a given task, and allows the sensorimotor subsystem in the robot to operate on that specific object. For example, suppose that the symbol system decides to execute the action to grasp "RedBall". The symbol "RedBall" needs to be connected to the perceptual knowledge about the red ball in order to successfully do the grasping. The anchor can be created either when the connection with the perceptual knowledge is acquired or previously using the prior knowledge about the object. In the case of the ball, the anchor is first created when the planner decides to catch that ball. The prior knowledge stored in the knowledge base about the red ball is put in the anchor and is used by the controller to grasp the ball. When the ball is perceived and recognized as "RedBall", the actual perceptual data of the ball are put in the anchor and used by the controller for executing the catching ball behavior.

The second mechanism to create an anchor is bottom-up, or event-driven. When the perceptual system perceives an object that is or could be of interest, it creates an anchor for it. The aim is to keep in memory perceptual information about objects that can be used later on in the anchoring process. An example of this can be found in [140], which focuses on the interpretation of scenes using linguistic terms.

Finally, the third mechanism is the combination between the last two techniques. In this type of systems, the symbolic system is trying to connect the symbol to its corresponding perceptual data. If the detected object cannot be matched to any symbol in the knowledge base, the system creates a new symbol for it to keep it in memory for future use.

B.3.2 The Challenges of Anchoring

Anchoring is a problem that can be studied from a number of different perspectives and within several disciplines. A study of the anchoring problem can raise a number of very challenging issues from each of these perspectives. Therefore, some challenges for the anchoring process in the robotic system will be addressed in the following paragraphs.

A first challenge is represented by the presence of uncertainty and ambiguity [66]. Uncertainty and ambiguity obviously arise when anchoring is performed using real sensors, which have intrinsic limitations, and in an environment which cannot be optimized in order to reduce these limitations. The anchoring process might incorporate provisions to deal with these limitations, for instance by managing multiple hypotheses. Alternatively, it can rely on the perceptual system to filter out the uncertainty, or it can delegate the resolution of ambiguities to the symbolic level.

Another challenge of anchoring is that, at the symbolic level, there are several ways to refer to objects. An important distinction is that between definite and indefinite symbolic descriptions [29]. A definite description implies the existence of a unique object satisfying the description in the current context. An indefinite description denotes an object having a number of properties, without any assumption about its uniqueness. The importance of this distinction appears mainly when more than one object satisfies the description: this can be a problem in the case of definite descriptions, but not in the case of indefinite ones.

Difficult issues of communication and negotiation may arise if several robots need to not only anchor symbols internally but also exchange information among them and agree on a shared language [128, 140]. Common agreement about the meaning of the symbols used to refer to objects in the environment is also needed for efficient human-robot cooperation.

Finally, a fundamental challenge of the anchoring problem is to investigate the formal properties of the anchoring process [29]. Intuitively one may feel that some correspondences between the symbols and the sensor data are correct while some are not. How to express this formally and prove the correctness of a specific system are unresolved problems. Engaging in this study would probably require the ability to model both the anchoring system and physical environment in the same formal system, in which formal properties can be defined and proved.

In the last ten years, numerous research groups worldwide have been concentrating on dealing with the anchoring problem in autonomous robot navigation. For example, Coradeschi and Saffiotti [28] proposed a domain-independent definition of the anchoring problem, and identified its three basic functionalities: find, reacquire, and track. They illustrated their proposed anchoring definition on two systems operating in two different domains: an unmanned airborne vehicle for traffic surveillance and a mobile robot for office navigation.

Shapiro and Ismail [128] considered how the anchoring problem is addressed in grounded layered architecture with integrated reasoning (GLAIR), a three-level architecture for cognitive robots. The robot used in the experiments interacts with humans using natural language, and in order to answer the user's queries it needs to connect its visual input to the linguistics terms used by the human. The robot uses abstract knowledge of objects and persons to make this connection.

Lang et al. [84] dealt with the problem of anchoring a composite object from the data provided by several sensors, each one of which can only observe part of the object. The authors consider the case of anchoring a human by aggregating the two anchors separately created for the face and for the legs. Face recognition is based on image data, while leg recognition relies on data from a laser range finder. Their system can be seen as a special case of cooperative anchoring, in which a common anchor must be established between two perceptual systems.

Steels and Baillie [140] considered the anchoring not only of objects but also of events. Their system anchors objects seen in the images bottom-up, and keeps track of them over time. On the basis of this information, events are recognized. This work is affected in the context of a language game between two robotic systems with the aim of learning a shared language. One of the systems sees an event, like a ball rolling, through a static camera in an otherwise static environment. It then formulates a sentence describing the event. The other system hears the sentence and interprets it. If the interpretation is considered appropriate with respect to one of the events recently seen, the game succeeds.

B.4 Conceptual Spaces

Conceptual spaces have been recently introduced as a way to bridge the gap between symbolic and sub-symbolic AI by providing a geometric treatment of concepts and knowledge representation [48]. A conceptual space has dimensions that are related with the concepts managed at the symbol level as well as with the quantities processed by the sensors. On the one hand, conceptual spaces can be used to represent discrete concepts which are the main entities manipulated at the symbol level. On the other hand, continuous observable quantities, which are the main entities provided by the perceptual system, can be placed inside a structure. Therefore, conceptual spaces provide an intermediate representation in which both symbolic and sensor-based information can be integrated.

In addition, conceptual spaces are endowed with a geometric structure that permits the performing of topology- and similarity-based reasoning inside the space itself. They are therefore well suited to formalize the types of reasoning needed for perception. Because of these reasons, it was pointed out in [22] that conceptual spaces could offer a fruitful setting for the study, formalization and implementation of perceptual anchoring.

Gärdenfors [48] has introduced a conceptual space as a metric space whose dimensions, called qualities, are related with the quantities processed by the robot sensors. Examples of dimensions are color coordinates (HSV) and spatial coordinates. Concepts are represented by regions in a conceptual space: a concept corresponds to the region of the space in which the points that are considered instances of that concept are located. A special role is played by so-called natural concepts, which correspond to convex regions in a conceptual space. For a natural concept, points in a conceptual space, called knoxels, represent the epistemologically primitive elements at the considered level of analysis. For instance, a knoxel can represent an individual object, which is characterized by a given value for each dimension of the conceptual space. The distance between two knoxels, according to the given metric, is interpreted as a measure of similarity between the entities represented by the robot's sensors.

Chella et al. [21] proposed a computational framework for anchoring based on conceptual spaces. Their framework exploits the geometric structure of conceptual spaces for many of the crucial tasks of anchoring, like matching precepts to symbolic descriptions or tracking the evolution of objects over time. This framework builds on the one proposed in [28], reformulated in a conceptual space setting. As it turns out, the new framework has a number of advantages over the original one. First, it clarifies the integration between perceptual and symbolic information since both types of information are represented in the same formal structure. Second, it clarifies the dynamic aspect of anchoring by modeling objects as trajectories in the conceptual space. Finally, it uses generic functions that exploit the geometric structure of the conceptual space. They represented the individual objects by knoxels in the conceptual space. However, in a dynamic perspective, objects can be more profitably seen as trajectories in the conceptual space indexed by time. The properties of objects usually change with time: objects may move, an object can alter its shape or color, and so on. As the properties of an object are modified, the point representing it in a conceptual space moves and describes a certain trajectory. Several assumptions can be made on this trajectory, e.g., smoothness and obedience to physical laws.

The interest of conceptual spaces in [21] is that they constitute an intermediate level between the symbolic system and the perceptual system. From the perceptual side, knoxels in a conceptual space can represent the entities coming from the perceptual system, together with their measured attributes. These knoxels are abstractions of the sensor data, since they represent a summary of the information regarding a certain object coming from different sensors. For instance, a knoxel can represent the information about the position, size and color of a given door as measured by a laser system plus a vision system. From the symbolic side, the conceptual space can be seen as an internal semantics for the symbol system. Predicates in the symbol system are mapped to regions of the conceptual space, and individual constants are mapped to knoxels. This semantics is perceptually grounded, since the elements of the conceptual space are directly related to perception.

Figure B.3, adopted from [21], shows the connection between a symbol and its corresponding percept by using conceptual space. The conceptual space in this example only has the two qualities Hue and Size. The h function transforms a measurement vector from the sensor system into a set of knoxels in the conceptual space. The g function gives semantics to symbolic predicates in terms of observable qualities in the conceptual space. The anchor, denoted by $\alpha(t)$, connects the symbol "cup-22" to a knoxel derived by the observation of a given cup. The concepts Blue and Small constrain the region in the conceptual space where this knoxel can be found.

Once an anchor is found, it should be updated in order to keep the symbol aligned to the corresponding perceptual data as those data change with time. The anchor should therefore account for the object persistence in face of a flow of different knoxels from the perceptual system that all originate from the same object, and in face of changes in the properties of the object, e.g., its position.

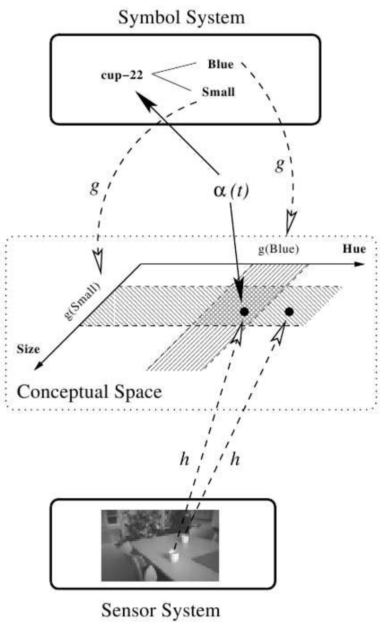


Figure B.3: Anchoring a symbol by using conceptual space [21].

APPENDIX C Heuristic Search Algorithms

To compute footstep placements for biped humanoid robots, a search method is needed to calculate the shortest feasible footstep sequence from the initial position to the target position. Heuristic search methods can be used to follow the roadmap graph and retrieve a shortest low-cost footstep sequence for the humanoid robots. They are fast enough to sense uncertainty, modeling errors, and handle obstacles for real-time re-planning in dynamic and unknown environments.

Heuristic algorithms use task-specific information in the form of approximations of goal distances to focus the search and typically solve search problems much faster than uninformed search methods [24]. They involve an iterated discrete search over a set of valid footstep placements. The result of the computation is a sequence of footstep placements that reach a goal region while minimizing encoded heuristics for effort, risk, or the number and complexity of the steps taken. As with other large search domains, computing true optimal solutions for biped navigation is computationally intractable. The challenge then becomes to exploit the problem structure and design efficient cost metrics and heuristics that improve search performance.

C.1 A* Algorithm

The A* search algorithm [52] is a well-known technique, well-regarded for its accuracy and calculation speed in searching for an optimal solution. It works by exploring nodes based on a cost function $f(n)$ which is the sum of the step cost $g(n)$, the cost from the start node to the current node n, and the estimated cost $h(n)$ from node n to the goal. It uses a heuristic search to estimates the cost to the goal node and minimizes the cost of the path so far. A* is optimal if the estimated cost to the goal is always underestimated. Since the shortest distance between two points is a straight line, Euclidean distance serves as an excellent estimated cost to the goal, making A* well-suited for fast computations.

Many researchers [46, 23, 143] used the A* search algorithm in planning the footstep placements for their humanoid robots. For example, Michel et al. [103] performed an A* search on the possible sequences of the footstep placements. Their ASIMO robot can be commanded to affect these footsteps, until an obstacle-avoiding path to the goal is found or a specified computation time limit is exceeded. Their planner computes the cost of each candidate footstep location using three cost metrics. First, the location cost determining whether the candidate location is safe or a part of an obstacle in the environment. Second, the step cost which prefers easy stepping actions. Finally, the estimated cost-to-go providing an approximation of the candidate's proximity to the goal using a standard mobile robot planner. They combine the set of humanoid robot actions into a sequence of footstep placements by mapping both the knowledge of the current environment and the commanded actions to the resulting states.

C.2 D* Algorithm

Incremental search methods reuse information from previous searches to find solutions to a series of similar tasks which will be much faster than solving each search task from scratch. Focused Dynamic A* (D*) [141] is a heuristic search method that repeatedly determines a shortest path from the current robot coordinates to the goal coordinates while the robot moves along the path. It combines the efficiency of heuristic and incremental searches, yet still finds shortest paths. D* uses a clever heuristic to speed up re-planning by one or two orders of magnitude over repeated A* searches by modifying previous search results locally. Consequently, it has been extensively used in mobile robotics.

On the other hand, D* search is an established algorithm but few people understand how it works. It is very complex and thus hard to understand, analyze, and extend (with the exception of different implementations by the author Antony Stenz himself) [73]. Consequently, while D* has been widely used as a black-box method, it has not been extended by other researchers since it was first published.

C.3 LPA* Algorithm

The Lifelong Planning A* (LPA*) [74] is an incremental version of the A* algorithm and shares many similarities with it. LPA* is an incremental heuristic search method that repeatedly determines shortest paths between two given vertices as the edge costs of a graph change. It uses heuristics to focus the search and reduce re-planning times. An incremental search tends to only recalculate those start distances (that is, distance from the start vertex to a vertex) that have changed (or have not been calculated before) and a heuristic search tends to only recalculate those start distances that are relevant for recalculating a shortest path from the start vertex to the goal vertex. Thus, LPA* recalculates only very few start distances.

LPA* search maintains an estimate $g(n)$ of the start distance of each vertex n. It also maintains rhs-values, a second kind of estimates of the start distances. The rhs-values are one-step look ahead values based on the g-values and thus potentially better informed than the g-values. A vertex is called consistent if and only if its g-value equals its rhs-value, otherwise it is called inconsistent. This concept is important because the g-values of all vertices equal their start distances if and only if all vertices are consistent. However, LPA* does not make every vertex consistent after some of the edge costs have changed. First, it does not re-compute the start distances that has been computed before and has not changed. Second, LPA* uses heuristic knowledge, in the form of approximations of the goal distances, to focus the search and determine that some start distances need not be computed at all (heuristic search), similar to A*.

The priorities are compared according to a lexicographic ordering. LPA* recalculates the g-values of vertices in the priority queue in the order of increasing first priority components, which correspond to the f-values of an A* search, and vertices with equal first priority components in the order of increasing second priority components, which correspond to the g-values of an A* search. Thus, it expands vertices in a similar order as an A* search, that expands vertices in the order of increasing f-values (since the heuristics are consistent) and vertices with equal f-values that are on the same branch of its search tree in the order of increasing g-values.

C.4 D* Lite Algorithm

The D* Lite algorithm [72, 97] is an application of a modified LPA* for the goal-directed robot navigation task in unknown terrain. It determines repeatedly the shortest paths between the current vertex of the robot and the goal vertex as the edge costs of a graph change while the robot moves towards the goal vertex. Consequently, it can be introduced as an alternative to D* that implements the same navigation strategy but is algorithmically different. Since D* Lite is based on LPA*, it is simple, easy to understand, easy to analyze and easy to extend. It also inherits all of the properties of LPA* and can be extended in the same way as LPA*. In addition, it has more than thirty percent fewer lines of code than D* [73].

D* Lite searches from the goal vertex to the start vertex and thus its g-values are estimates of the goal distances. It is derived from LPA* by exchanging the start and goal vertex and reversing all edges in the pseudo code. On the other hand, D* Lite borrowed the priority handling method from the D* algorithm to avoid repeatedly reordering the priority queue. One can efficiently recalculate a shortest path from the current vertex of the robot to the goal vertex by recalculating only those goal distances that have been changed (or have not been calculated before) and are relevant for recalculating the shortest path. This is what D* Lite does. The challenge is to identify these vertices efficiently. D* Lite does not make any assumptions about how the edge costs change, whether they go up or down, whether they change close to the current vertex of the robot or far away from it, or whether they change in the world or only because the robot revised its initial estimates. The goal-directed navigation problem in unknown terrain then is a special case of this problem, where the graph is an eight-connected grid whose edge costs are initially one and change to infinity when the robot discovers that they cannot be traversed.

The complete list of D* Lite is illustrated in Algorithm 5. The finite set of vertices of the graph is denoted by S. $Succ(s) \subseteq S$ denotes the set of successors of vertex $s \in S$. Similarly, $Pred(s) \subseteq S$ denotes the set of predecessors of vertex $s \in S$. $0 < c(s, s^{'}) \leq \infty$ denotes the cost of moving from vertex s to vertex $s^{'} \in Succ(s)$. The priority of a vertex in the priority queue (U) is denoted by $k(s) = [k_1(s); k_2(s)]$. On the other hand, D* Lite uses heuristics $h(s, s^{'})$ that approximate the goal distance of the vertex s. The heuristics need to be nonnegative and satisfy $h(s, s^{'}) \leq c^*(s, s^{'})$ and $h(s, s^{''}) \leq h(s, s^{'}) + h(s^{'}, s^{''})$ for all vertices $s, s^{'}, s^{''} \in S$, where $c^*(s, s^{'})$ denotes the cost of a shortest path from vertex $s \in S$ to vertex $s^{'} \in S$.

Algorithm 5: D* Lite Search

```
Procedure CalculateKey(s)
  return $[min(g(s), rhs(s)) + h(s_{init}, s) + k_m; min(g(s), rhs(s))]$;
Procedure Initialize()
  $U = \emptyset$;
  $k_m = 0$;
  forall $s \in S$ do $rhs(s) = g(s) = \infty$;
  $rhs(s_{goal}) = 0$;
  $U.Insert(s_{goal}, CalculateKey(s_{goal}))$;
Procedure UpdateVertex(u)
  if $(u \neq s_{goal})$ then $rhs(u) = min_{\acute{s} \in Succ(u)}(c(u, \acute{s} + g(\acute{s}))$;
  if $(u \in U)$ then $U.Remove(u)$;
  if $(g(u) \neq rhs(u))$ then $U.Insert(u, CalculateKey(u))$;
Procedure ComputeShortestPath()
  while $(U.TopKey() < CalculateKey(s_{init})\ OR\ rhs(s_{init}) \neq g(s_{start}))$
  do
      $k_{old} = U.TopKey()$;
      $u = U.Pop()$;
      if $(k_{old} < CalculateKey(u))$ then
          $U.Insert(u, CalculateKey(u))$;
      else if $(g(u) > rhs(u))$ then
          $g(u) = rhs(u)$;
          forall $s \in Pred(u)$ do $UpdateVertex(s)$;
      else
          $g(u) = \infty$;
          forall $s \in Pred(u) \cup \{u\}$ do $UpdateVertex(s)$;

Procedure Main()
  $s_{last} = s_{init}$;
  $Initialize()$;
  $ComputeShortestPath()$;
  while $s_{init} \neq s_{goal}$ do
      $s_{init} = arg\ min_{\acute{s} \in Succ(s_{init})}(c(s_{init}, \acute{s}) + g(\acute{s}))$;
      Move to $s_{init}$;
      scan graph for changed edge costs;
      if any edge costs changed then
          $k_m = k_m + h(s_{last}, s_{init})$;
          $s_{last} = s_{init}$;
          forall directed edges $(u, v)$ with changed edge costs do
              Update the edge cost $c(u, v)$;
              $UpdateVertex(u)$;
          $ComputeShortestPath()$;
```

APPENDIX D

Publications

The chapters of this thesis are based on the following papers:

1. Elmogy, M., Habel, C., and Zhang, J. Robot topological map generation from formal route instructions. In *proceedings of the 6th International Cognitive Robotics Workshop at 18th European Conference on Artificial Intelligence (ECAI)* (Patras, Greece, July 2008), IOS Press, pp. 60–67. July 21-22.

2. Elmogy, M., and Zhang, J. Robust real-time landmark recognition for humanoid robot navigation. In *proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics (ROBIO'08)* (Bangkok, Thailand, December 2008), IEEE, pp. 572–577. December 14-17.

3. Elmogy, M., Habel, C., and Zhang, J. Spatial language for route-based humanoid robot navigation. *Cognitive Processing* 10, 2 (September 2009), 208–211.

4. Elmogy, M., Habel, C., and Zhang, J. Cognitive instruction interface for mobile robot navigation. In *proceedings of the International Conference on Computer Engineering and Systems (ICCES'09)* (2009), pp. 115–120.

5. Elmogy, M., Habel, C., and Zhang, J. Online motion planning for hoap-2 humanoid robot navigation. In *proceedings of the 2009 IEEE International Conference on Intelligent Robots and Systems (IROS'09)* (St. Louis, Missouri, USA, October 2009), pp. 3531–3536.

6. Elmogy, M., Habel, C., and Zhang, J. *A Cognitively Motivated Route-Interface for Mobile Robot Navigation*, vol. 6 *of Cognitive Systems Monographs.* Springer Berlin/Heidelberg, 2009, pp. 73–82.

7. Elmogy, M., Habel, C., and Zhang, J. Time efficient hybrid motion planning algorithm for hoap-2 humanoid robot. In *proceedings of the 2010 ISR/ROBOTIK Conference* (Munich, Germany, 2010), pp. 1046–1053.

8. Elmogy, M., Habel, C., and Zhang, J. Multimodal Cognitive Interface for Robot Navigation. *(Submitted)*.

9. Elmogy, M., Habel, C., and Zhang, J. Landmark Processing System for Mobile Robot Navigation. *(Submitted)*.

10. Elmogy, M., Habel, C., and Zhang, J. A Dynamic Sampling-based Motion Planner for Humanoid Robot. *(Submitted)*.

BIBLIOGRAPHY

[1] Bay, H., Ess, A., Tuytelaars, T., and Van Goo, L. Speeded-up robust features (surf). *Computer Vision and Image Understanding (CVIU) 110*, 3 (2008), 346–359.

[2] Belur, R. Investigation of the use of humanoids for industrial robot safety standards. Tech. rep., Intelligent Systems Division, National Institute of standards and technology, August 2007.

[3] Bianchi, R., Ramisa, A., and Lopez de Mantaras, R. Learning to select object recognition methods for autonomous mobile robots. In *proceedings of the 18th European Conference on Artificial Intelligence* (Patras, Greece, July 2008), pp. 927–928.

[4] Birchfield, S., and Tomasi, C. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision 35*, 3 (1999), 269–293.

[5] Bischoff, R., and Jain, T. Natural communication and interaction with humanoid robots. In *proceedings of the 2nd International Symposium on Humanoid Robots* (Tokyo, October 1999).

[6] Bohlin, R., and Kavraki, L. E. Path planning using lazy prm. In *proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00)* (San Francisco, CA, USA, April 2000), vol. 1, pp. 521–528.

[7] Bouilly, B., and Simeon, T. *A Sensor-based motion planner for mobile robot navigation with uncertainty*, vol. 1093/1996 of *Lecture Notes in Computer Science.* Springer Berlin/Heidelberg, 1996, pp. 235–247.

[8] Bourgeot, J.-M., Cislo, N., and Espiau, B. Path-planning and tracking in a 3d complex environment for an anthropomorphic biped robot. In *proceedings of the IEEE/RSJ International Conference on ntelligent Robots and Systems.* (2002), vol. 3, pp. 2509–2514.

[9] Bradski, G., and Kaebler, A. *Learning OpenCV*, 1 ed. O'Reilly, 2008.

[10] Brennan, S. E. *The Grounding Problem in Conversations With and Through Computers.* 1991, pp. 201–225.

[11] Brown, D. C. Close-range camera calibration. *Photogrammetric Engineering 37*, 8 (1971), 855–866.

[12] Browning, B., and Veloso, M. Real-time adaptive color-based robot vision. In *proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS'05).* (Augest 2005), pp. 3871–3876.

[13] Bruce, J., Balch, T., and Veloso, M. Fast and inexpensive color image segmentation for interactive robots. In *proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00)* (Octobar 2000), vol. 3, pp. 2061–2066.

[14] Bruce, J., and Veloso, M. Real-time multi-robot motion planning with safe dynamics. *Multi-Robot Systems: From Swarms to Intelligent Automata III* (2005).

[15] Bugmann, G., Klein, E., Lauria, S., and Kyriacou, T. Corpus-based robotics: A route instruction example. In *proceedings of the conference on Intelligent Autonomous System* (Amsterdam, March 2004), pp. 96–103.

[16] Burns, B., and Brock, O. Sampling-based motion planning using predictive models. In *proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA'05)* (April 2005), pp. 3120–3125.

[17] Calinon, S., and Billard, A. Teaching a humanoid robot to recognize and reproduce social cues. In *proceedings of the IEEE international Symposium on Robot and Human Interactive Communication (ROMAN'06)* (2006), pp. 346–351.

[18] Carpin, S. Randomized motion planning - a tutorial. *International Journal of Robotics and Automation 21*, 3 (2006), 184–196.

[19] Censi, A., Calisi, D., De Luca, A., and Oriolo, G. A bayesian framework for optimal motion planning with uncertanity. In *proceedings of the IEEE International Conference on Robotics and Automation (ICRA'08)* (Pasadena, CA, May 2008).

[20] Chalodhorn, R., Grimes, D. B., Maganis, G., and Rao, R. P. N. Learning dynamic humanoid motion using predictive control in low dimensional subspaces. In *proceedings of the 5th IEEE-RAS International Conference on Humanoid Robots* (2005), IEEE Xplore, pp. 214–219.

[21] Chella, A., Coradeschi, S., Frixione, M., and Saffiotti, A. Perceptual anchoring via conceptual spaces. In *proceedings of the AAAI-04 Workshop on Anchoring Symbols to Sensor Data* (2004), AAAI Press, Menlo Park, California.

[22] Chella, A., Frixione, M., and Garlio, S. Conceptual spaces for anchoring. *Robotics and Autonomous Systems 43*, 2-3 (2003), 193–195.

[23] Chestnutt, J., Kuffner, J., Nishiwaki, K., and Kagami, S. Planning biped navigation strategies in complex environments. In *processdings of the IEEE International Conference on Robotics and Automation (ICRA'03)* (2003).

[24] Chestnutt, J., Lau, M., Cheung, G., Kuffner, J., Hodgins, J., and Kanade, T. Footstep planning for the honda asimo humanoid. In *processing of the 2005 IEEE International Conference on Robotics and Automation* (Barcelona, Spain, 2005), pp. 629–633.

[25] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., and Thrun, S. *Principles of Robot Motion-Theory, Algorithms, and Implementations.* The MIT Press, June 2005.

[26] Clark, H. H. *Arenas of language use.* University of Chicago Press, 1992.

[27] Clark, H. H., and Brennan, S. E. *Grounding in Communication.* American Psychological Association, Washington, DC, USA, 1991, ch. 7, pp. 127–149.

[28] Coradeschi, S., and Saffiotti, A. Anchoring symbols to sensor data: Preliminary report. In *proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00)* (Austin, 2000), pp. 129–135.

[29] Coradeschi, S., and Saffiotti, A. An introduction to the anchoring problem. *Robotics and Autonomous Systems 43* (2003), 85–96.

[30] Drysdale, J. D., and Lyons, D. Learning image-based landmarks for wayfinding using neural network. In *Artificial Neural Networks in Engineering* (St. Louis, MO., 2004).

[31] Ekvall, S., Hoffmann, F., and Kragic, D. Object recognition and pose estimation for robotic manipulation using color cooccurrence histograms. In *proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)* (2003), vol. 2, pp. 1284–1289.

[32] Ekvall, S., Jensfelt, P., and Kragic, D. Integrating active mobile robot object recognition and slam in natural environments. In *proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)* (2006), pp. 5792–5797.

[33] Ekvall, S., Kragic, D., and Jensfelt, P. Object detection and mapping for service robot tasks. *Robotica 25*, 2 (March 2007), 175–187.

[34] Ekvalla, S., Kragicb, D., and Hoffmannc, F. Object recognition and pose estimation using color cooccurrence histograms and geometric modeling. *Image and Vision Computing 23*, 11 (2005), 943–955.

[35] Ersson, T., and Hu, X. Path planning and navigation of mobile robots in unknown environments. In *proceedings of the IEEE International Conference of Intelligent Robots and Systems* (2001).

[36] Eschenbach, C., Tschander, L., Habel, C., and Kulik, L. *Lexical Specifications of Paths*, vol. 1849/2000 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2000, pp. 127–144.

[37] Fasola, J., and Veloso, M. Real-time object detection using segmented and grayscale images. In *proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA'06)* (May 2006), pp. 4088–4093.

[38] Ferguson, D., Kalra, N., and Stentz, A. Replanning with rrts. In *proceedings of the IEEE International Conference on Robotics and Automation (ICRA'06)* (May 2006), pp. 1243–1248.

[39] Florczyk, S. *Robot Vision: Video-based Indoor Exploration with Autonomous and Mobile Robots*, 1st ed. WILEY-VCH Verlag GmbH and Co. KGaA, Weinheim, 2005.

[40] Fong, T., Thorpe, C., and Baur, C. *Collaboration, Dialogue, and Human-Robot Interaction*, vol. 6 of *Springer Tracts in Advanced Robotics*. Springer Berlin/Heidelberg, 2003, pp. 255–266.

[41] Fujitsu Automation Company Ltd., . *HOAP-2 Instruction Manual*, 3rd ed., 2004.

[42] Fusiello, A., Trucco, E., and Verri, A. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications 12*, 1 (2000), 16–22.

[43] Geraerts, R., and Overmars, M. H. Sampling techniques for probabilistic roadmap planners. In *proceedings of the Conference on Intelligent Autonomous Systems (IAS-8)* (2004), pp. 600–609.

[44] Goodrich, M. A., and Schultz, A. C. Humanrobot interaction: A survey. *Foundation and Trends in Human-Computer Interaction 1*, 3 (2007), 203–275.

[45] Gopalakrishnan, A., Greene, S., and Sekmen, A. Vision-based mobile robot learning and navigation. In *proceedings of the IEEE International Workshop on Robot and Human Interactive Communication (ROMAN'05)* (2005).

[46] Gutmann, J.-S., Fukuchi, M., and Fujita, M. Real-time path planning for humanoid robot navigation. In *proceedings of the International joint conference on artificial intelligence (IJCAI-05)* (2005), pp. 1232–1237.

[47] Guzel, M. S. Mobile robot navigation using a vision based approach. In *processdings of the Newcastle University Postgraduate Conference* (2009).

[48] Grdenfors, P. *Conceptual Spaces: The Geometry of Thought.* MIT Press, 2000.

[49] Habel, C. Incremental generation of multimodal route instructions. In *proceedings of the AAAI Spring Symposium on Natural language generation in spoken and written dialogue* (Palo Alto, CA, March 2003), pp. 44–51.

[50] Harnad, S. The symbol grounding problem. *Physica D 42*, 1-3 (June 1990), 335 – 346.

[51] Harnad, S. *The Symbol Grounding Problem.* Nature Publishing Group/Macmillan, 2003.

[52] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 2 (July 1968), 100–107.

[53] Hartley, R. I., and Sturm, P. Triangulation. *Computer Vision and image understanding 68*, 2 (1997), 146–157.

[54] Hillman, P. White paper: Camera calibration and stereo vision. Tech. rep., Square Eyes Software, UK, 2005.

[55] Hirai, K., Hirose, M., Haikawa, Y., and Takenaka, T. The development of honda humanoid robot. In *proceedings of the International Conference on Robotics and Automation* (1998), pp. 1321–1326.

[56] Hirose, M., and Ogawa, K. Honda humanoid robots development. *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences 365*, 1850 (January 2007), 11–19.

[57] Honda ASIMO Web Site, . http://world.honda.com/ASIMO/.

[58] Hsu, D., Jiang, T., Reif, J., and Sun, Z. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *proceedings of the IEEE International Conference on Robotics and Automation* (2003), pp. 4420–4426.

[59] Hussein, A. M., and Elnagar, A. A fast path planning algorithm for robot navigation with limited visibilty. In *proceedings of the IEEE International Conference on Systems, Man and Cybernetics* (Oct 2003), vol. 1, pp. 373–377.

[60] INourbakhsh, l. R., Bobenage, J., Grange, S., Lutz, R., Meyer, R., and Soto, A. An affective mobile robot educator with a full-time job. *Artificial Intelligence 114*, 1-2 (October 1999), 95–124.

[61] Iocchi, L., and Konolige, K. Multiresolution stereo vision system for mobile robots. In *processding of the AI&IA Workshop on New Trends in Robotics* (1998).

[62] Jan, G. E., Chang, K. Y., and Parberry, I. Optimal path planning for mobile robot navigation. *IEEE/ASME Transactions on Mechatronics 13*, 4 (Aug 2008), 451–460.

[63] Jeanne, J. M. Developing adjustable walking patterns for natural walking in humanoid robots. *Jet Propulsion Laboratory, Princeton University* (19 August 2004).

[64] Kanda, T., Ishiguro, H., Ono, T., Imai, M., and Nakatsu, R. Development and evaluation of an interactive humanoid robot robovie. In *proceeding the IEEE International Conference Robotics and Automation (ICRA'02)* (2002), vol. 2, pp. 1848–1855.

[65] Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., and Isozumi, T. Humanoid robot hrp-2. In *proceedings of the 2004 IEEE International*

Conference on Robotics & Automation (New Orleans, LA, April 2004), pp. 1083–1090.

[66] Karlsson, L., Bouguerra, A., Broxvall, M., Coradeschi, S., and Saffiotti, A. To secure an anchor - a recovery planning approach to ambiguity in perceptual anchoring. *AI Communications 21*, 1 (2008), 1–14.

[67] Khosla, P., and Volpe, R. Superquadratic artificial potentials for obstacle avoidance and approach. In *proceedings of the IEEE Conference on Robotics and Automation (ICRA'88)* (April 1988).

[68] Kidd, P. T. *Design of human-centred robotic systems.* Taylor and Francis: London, 1992, ch. 12, pp. 225–241.

[69] Kiesler, S. Fostering common ground in human-robot interaction. In *processdings of the IEEE International Workshop on Robot and Human Interactive Communication (ROMAN'05)* (2005), pp. 729–734.

[70] Kim, S., Lee, S., Kim, S., and Lee, J. Object tracking of mobile robot using moving color and shape information for the aged walking. *International Journal of Advanced Science and Technology 3* (2009), 59–68.

[71] Kobilarov, M., and Sukhatme, G. S. Near time-optimal constrained trajectory planning on outdoor terrain. In *proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA'05)* (Barcelona,Spain, April 2005), pp. 1821–1828.

[72] Koenig, S., and Likhachev, M. D* lite. In *proceedings of the AAAI Conference of Artificial Intelligence (AAAI'02)* (2002), pp. 476–483.

[73] Koenig, S., and Likhachev, M. Improved fast replanning for robot navigation in unknown terrain. In *proceedings of the IEEE International Conference on Robotics and Automation (ICRA'02)* (2002), vol. 1, pp. 968– 975.

[74] Koenig, S., and Likhachev, M. Incremental a*. *Advances in Neural Information Processing Systems (NIPS)* (2002), 1539–1546.

[75] Kollar, T., Tellex, S., Roy, D., and Roy, N. Toward understanding natural language directions. In *proceeding of the 5th ACM/IEEE international conference on Human-robot interaction (HRI'10)* (Osaka, Japan, 2010), pp. 259–266.

[76] Kuffner, J., Kagami, S., Nishiwaki, K., Inaba, M., and Inoue, H. Online footstep planning for humanoid robots. In *proceedings of the IEEE International Conference on Robotics and Automation (ICRA'03)* (September 2003), IEEE.

[77] Kuffner, J. J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *proceedings of the IEEE International Conference on Robotics and Automation (ICRA'01)* (Seoul, Korea, May 2001), pp. 692–698.

[78] Kuffner, J. J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. Motion planning for humanoid robots. In *proceedings of the 20th International Symposium Robotics Research (ISRR'03)* (Italy, October 2003).

[79] Kuipers, B. J., and Levit, T. S. Navigation and mapping in large scale space. *AI Magazine 9* (1988), 25–43.

[80] Kuipers, B. J., Tecuci, D. G., and Stankiewicz, B. J. The skeleton in the cognitive map : A computational and empirical exploration. *Environment & Behavior 35* (2003), 80–106.

[81] Kwon, E., and Kim, G. J. Humanoid robot vs. projector robot: exploring an indirect approach to human robot interaction. In *proceedings of the 5th ACM/IEEE international conference on Human-robot interaction (HRI'10)* (Osaka, Japan, 2010), pp. 157–158.

[82] Kyriacou, T., Bugmann, G., and Lauria, S. Vision-based urban navigation procedures for verbally instructed robots. *Robotics and Autonomous Systems 51* (2002), 1326–1331.

[83] Landau, B., and Jackendoff, R. what and where in spatial language and spatial cognition. *Behavioral and Brain Sciences 16* (1993), 217–265.

[84] Lang, S., Kleinehagenbrock, M., Hohenner, S., Fritsch, J., Fink, G. A., and Sagerer, G. Providing the basis for human-robot-interaction: A multi-modal attraction system for a mobile robot. In *proceedings of the International Conference on Multimodal Interfaces* (Vancouver, Canada, November 2003), pp. 28–35.

[85] Latombe, J.-C. *Robot Motion Planning*, 1st ed. Kluwer Academic, 1991.

[86] Lau, M., and Kuffner, J. Behavior planning for character animation. In *proceedings of the 2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, CA, August 2005), pp. 271–280.

[87] Lauria, S., Bugmann, G., Kyriacou, T., Bos, J., and Klein, E. Training personal robots using natural language instruction. *IEEE Intelligent Systems 16* (2001), 38–45.

[88] LaValle, S. M. *Planning Algorithms.* Cambridge University Press, May 2006.

[89] Lazebnik, S., Schmid, C., and Ponce, J. Beyond bag-of-features: Spatial pyramid matching for recognizing natural scene categories. In *proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006), vol. 2, pp. 2169–2178.

[90] Lenz, R. K., and Tsai, R. Y. Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 5 (1988), 713–720.

[91] Levit, M., and Roy, D. Interpretation of spatial language in a map navigation task. *IEEE Transactions on Systems, Man, and Cybernetics Part B 37*, 3 (2007), 667–679.

[92] Lindemann, S. R., and LaValle, S. M. *Current Issues in Sampling-Based Motion Planning*, vol. 15 of *Springer Tracts in Advanced Robotics.* Springer Berlin/Heidelberg, 2005, pp. 36–54.

[93] Litvintseva, L., Tanaka, T., Yamafuji, K., and Ulyanov, V. Intelligence computing for direct human-robot communication using natural language and cognitive graphics. In *proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '97)* (Jul 1997), pp. 332–337.

[94] Lorch, O., Albert, A., Denk, J., Gerecke, M., Cupec, R., Seara, J. F., Gerth, W., and Schmidt, G. Experiments in vision guided biped walking. In *proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and System (IROS'02)* (2002), vol. 3, pp. 2484–2490.

[95] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[96] Lozano-Perez, T. Spatial planning: A configuration space approach. *IEEE Transactions on Computers C-32* (1983), 108–120.

[97] Mackay, D. Path planning with d* lite: Implementation and adaptation of the d* lite algorithm. *DRDC Suffield TM 2005-242: Defence RD Canada Suffield* (2005).

[98] MacMahon, M. Marco: A modular architecture for following route instructions. In *proceedings of the AAAI Workshop on Modular Con-*

struction of Human-Like Intelligence (Pittsburgh, PA, July 2005), pp. 48–55.

[99] MacMahon, M., and Stankiewicz, B. Human and automated indoor route instruction following. In *proceedings of the 28th Annual Conference of the Cognitive Science Society* (Vancouver, BC, July 2006), pp. 1759–1764.

[100] Macmahon, M., Stankiewicz, B., and Kuipers, B. Walk the talk: Connecting language, knowledge, and action in route instructions. In *proceedings of the 21st National Conference on Artificial Intelligence (AAAI-2006)* (Boston, USA, 2006), pp. 1475–1482.

[101] Matas, J., Galambos, C., and Kittler, J. Progressive probabilistic hough transform. In *British Machine Vision Conference* (1998), pp. 256–265.

[102] Michel, P., Chestnutt, J., Kagami, S., Nishiwaki, K., Kuffner, J. J., and Kanade, T. Online environment reconstruction for biped navigation. In *proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA'06)* (Orlando, Florida, May 2006), pp. 3089–3094.

[103] Michel, P., Chestnutt, J., Kuffner, J. J., and Kanade, T. Vision-guided humanoid footstep planning for dynamic environments. In *proceedings of the 2005 5th IEEE-RAS International Conference on Humanoid Robots* (2005), pp. 13–18.

[104] Mizuuchi, I. *A Musculoskeletal Flexible-Spine Humanoid Kotaro Aiming at the Future in 15 years' time.* Pro literatur Verlag, 2007, ch. 3, pp. 45–56.

[105] Montello, D. R. Navigation. In *The Cambridge handbook of visuospatial thinking.*, P. Shah & A. Miyake, Ed. Cambridge University Press, 2005, pp. 257–294.

[106] Murdock, K. L. *Google SketchUp and SketchUp Pro 7 Bible.* Wiley, 2009.

[107] Nielsen, C. W., Bruemmer, D. J., Few, D. A., and Gertman, D. I. Framing and evaluating human-robot interactions. In *proceedings of the Workshop on Metrics for Human-Robot Interaction* (Amsterdam, March 2008), pp. 29–36.

[108] Nieuwenhuisen, M., Stuckler, J., and Behnke, S. Intuitive multimodal interaction for service robots. In *proceedings of the 5th ACM/IEEE international conference on Human-robot interaction (HRI'10)* (Osaka, Japan, 2010), pp. 177–178.

[109] Nishiwaki, K., Kuffner, J., Kagami, S., Inaba, M., and Inoue, H. The experimental humanoid robot h7: a research platform for autonomous behaviour. *Philosophical Transactions of The Royal Society A 365* (2007), 79–107.

[110] Nister, D., and Stewenius, H. Scalable recognition with a vocabulary tree. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2006), vol. 2, pp. 2161–2168.

[111] Okada, K., Inaba, M., and Inoue, H. Walking navigation system of humanoid robot using stereo vision based floor recognition and path planning with multi-layered body image. In *proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)* (Las Vegas, Nevada, October 2003), pp. 2155–2160.

[112] PAL Robotics Web Site, . http://www.pal-robotics.com/.

[113] Perzanowski, D., Schultz, A. C., Adams, W., Marsh, E., and Bugajska, M. Building a multimodal human-robot interface. *Intelligent Systems 16*, 1 (2001), 16–21.

[114] Pires, G., and Nunes, U. A wheelchair steered through voice commands and assisted by a reactive fuzzy-logic controller. *Journal of Intelligent and Robotic Systems 34* (2002), 301–314.

[115] Plaku, E., Bekris, K., and Kavraki, L. E. Oops for motion planning: An online open-source programming system. In *proceedings of the International Conference on Robotics and Automation (ICRA'07)* (Rome, Italy, 2007), p. 37113716.

[116] Pradel, G., and Hoppenot, P. Symbolic trajectory description in mobile robotics. *Journal of Intelligent and Robotic Systems 45*, 2 (February 2006), 157–180.

[117] Ramisa, A., Vasudevan, S., Aldavert, D., Toledo, R., and Lopez de Mantaras, R. Evaluation of the sift object recognition method in mobile robots. In *proceedings of the 12th International Conference of Artificial Intelligence and Applications* (2009), vol. 202, pp. 9–18.

[118] Ribes, A., Ramisa, A., Lopez De Mantaras, R., and Toledo, R. Object-based place recognition for mobile robots using panoramas. In *proceeding of the 2008 conference on Artificial Intelligence Research and Development* (Amsterdam, The Netherlands, 2008), pp. 388–397.

[119] Roth, P. M., and Winter, M. Survey of appearance-based methods for object recognition. Tech. rep., Institute for Computer Graphics and Vision, Graz University of Technology, Austria, 2008.

[120] Rouanet, P., Oudeyer, P.-Y., and Filliat, D. A study of three interfaces allowing non-expert users to teach new visual objects to a robot and their impact on learning efficiency. In *proceedings of the 5th ACM/IEEE international conference on Human-robot interaction (HRI'10)* (Osaka, Japan, 2010), pp. 185–186.

[121] Roy, D. Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence 167* (2005), 170–205.

[122] Rubio, J. P. B., Zhou, C., and Hernndez, F. S. *Vision-Based Walking Parameter Estimation for Biped Locomotion Imitation*, vol. 3512/2005 of *Perception and Robotics-Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2005, pp. 677–684.

[123] Sabe, K., Fukuchi, M., Gutmann, J., Ohashi, T., Kawamoto, K., and Yoshigahara, T. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)* (2004), vol. 1, pp. 592–597.

[124] Salter, T., Dautenhahn, K., and Boekhorst, R. Learning about natural human-robot interaction styles. *Journal of Robotics and Autonomous Systems 52*, 2 (2006), 127–134.

[125] Sanchez L, A., Zapata, R., , and Osorio L, M. A. Sampling-based motion planning: A survey. *Computacion y Sistemas 12*, 1 (2008), 5–24.

[126] Scholtz, J. Theory and evaluation of human robot interactions. In *proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)* (Washington, DC, USA, 2003), vol. 5, pp. 125–134.

[127] Schulz, R., Stockwell, P., Wakabayashi, M., and Wiles, J. Towards a spatial language for mobile robots. In *proceedings of the 6th International Conference on the Evolution of Language* (2006), pp. 291–298.

[128] Shapiro, S. C., and Ismail, H. O. Anchoring in a grounded layered architecture with integrated reasoning. *Robotics and Autonomous Systems 43* (2003), 97–108.

[129] Shiller, Z., Yamane, K., and Nakamura, Y. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. In *proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA'01)* (2001), pp. 1–8.

[130] Shinozaki, K., Iwatani, A., and Nakatsu, R. *Study of Dance Entertainment Using Robots.* Pro literatur Verlag, 2007, ch. 27, pp. 535–544.

[131] Sian, N. E., Yokoi, K., Kajita, S., Kanehiro, F., and Tanie, K. Whole body teleoperation of a humanoid robot: development of a simple master device using joysticks. In *processings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Lausanne, Switzerland, October 2002), pp. 2569–2574.

[132] Siegwart, R., and Nourbakhsh, I. R. *Introduction to Autonomous Mobile Robots*, 1st ed. Massachusetts Institute of Technology, 2004.

[133] Simpson, R. C., and Levine, S. P. Adaptive shared control of a smart wheelchair operated by voice control. In *proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97)* (September 1997), vol. 2, pp. 622–626.

[134] Sjo, K., Galvez Lopez, D., Paul, C., Jensfelt, P., and Kragic, D. Object search and localization for an indoor mobile robot. *Journal of Computing and Information Technology 17*, 1 (2009), 67–80.

[135] Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., and Brock, D. Spatial language for human-robot dialogs. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions 34*, 2 (May 2004), 154–167.

[136] Solutions, A. K. *Robotics*, 1st ed. Jones & Bartlett Publishers, April 2007.

[137] Sonka, M., Halvac, V., and Boyle, R. *Image Processing Analysis and computer vision*, 3rd ed. Thomson, 2007.

[138] Sony QRIO Web Site, . http://www.sonyaibo.net/aboutqrio.htm.

[139] Steels, L. *The Symbol Grounding Problem has been Solved. So What's Next?* Academic Press, New Haven, 2007.

[140] Steels, L., and Baillie, J.-C. Shared grounding of event descriptions by autonomous robots. *Robotics and Autonomous Systems 43* (2003), 163–173.

[141] Stentz, A. The focussed d* algorithm for real-time replanning. In *proceedings of the International Joint Conference on Articial Intelligence* (1995), pp. 1652–1659.

[142] Stoica, A. Humanoids for lunar and planetary surface operations. In *proceedings for the 5th IEEE-RAS International Conference on Humanoid Robots* (2005), pp. 345–350.

[143] Taha, T., Valls Miro, J., and Dissanayake, G. Sampling based time efficient path planning algorithm for mobile platforms. In *proceeding of the 2006 IEE International Conference on Man-Machine Systems (ICoMMS 2006)* (Langkawi, Malaysia, Sep 2006).

[144] Takeda, H., Facchinetti, C., and Latombe, J.-C. Planning the motions of a mobile robot in a sensory uncertainty field. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 10 (October 1994), 1002–1017.

[145] Tanie, K. Humanoid robot and its application possibility. In *proceedings of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems* (2003), pp. 213–214.

[146] Tedder, M., and Hall, E. L. Symbolic processing methods for 3d visual processing. *Intelligent Robots and computer vision: Algorithms, techniques, and active vision 4572* (2001), 93–104.

[147] Tellex, S., and Roy, D. Spatial routines for a simulated speech-controlled vehicle. In *proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction* (Salt Lake City, Utah, USA, 2006), pp. 156–163.

[148] Tellex, S., and Roy, D. Grounding language in spatial routines. In *processings of the AAAI Spring Symposium on Control Mechanisms for Spatial Knowledge Processing in Cognitive/Intelligent Systems* (2007).

[149] Thrun, S. Toward a framework for human-robot interaction. *Human-Computer Interaction 19*, 1 (June 2004), 9–24.

[150] Torrance, M. C. *Natural Communication with Mobile Robots.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, Junuary 1994.

[151] Trullier, O., Wiener, S. I., Berthoz, A., and Meyer, J.-A. Biologically based artificial navigation systems: review and prospects. *Progress in Neurobiology 51*, 5 (April 1997), 483–544.

[152] Tsai, R. Y. A versatile camera calibration technique for high-accuracy 3-d machine vision metrology using off-the-self ty cameras and lenses. *IEEE Journal of Robotics and Automation 3*, 4 (1987), 323–344.

[153] Tschander, L. B., Schmidtke, H., Habel, C., Eschenbach, C., and Kulik, L. *A Geometric Agent Following Route Instructions*, vol. 2685/2003 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2003, pp. 89–111.

[154] Tsianos, K. I., Sucan, I. A., and Kavraki, L. E. Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review 1*, 1 (2007), 2–11.

[155] Vukobratovic, M., and Borovac, B. Zero-moment point -thirty five years of its life. *International Journal of Humanoid Robotics 1*, 1 (2004), 154–173.

[156] Vukobratovic, M., Borovac, B., and Babkovic, K. Contribution to the study of anthropomorphism of humanoid robots. *Journal Humanoids Robotics 2*, 3 (2005), 361–387.

[157] Weiss, A., Buchner, R., Scherndl, T., and Tscheligi, M. I would choose the other card: humanoid robot gives an advice. In *proceedings of the 4th ACM/IEEE international conference on Human robot interaction (HRI'09)* (La Jolla, California, USA, 2009), pp. 259–260.

[158] Werner, S., Krieg-Brckner, B., and Herrmann, T. *Modelling navigational knowledge by route graphs.* Lecture Notes in Computer Science. Springer Berlin/Heidelberg, January 2000, pp. 295–316.

[159] Yussof, H., Yamano, M., Nasu, Y., and Ohka, M. *Humanoid Robot Navigation Based on Groping Locomotion Algorithm to Avoid an Obstacle.* Pro literatur Verlag, 2007, ch. 1, pp. 01–26.

[160] Zavlangas, P. G., and Tzafestas, S. G. *Integration of Topological and Metric Maps for Indoor Mobile Robot Path Planning and Navigation.* Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2002, pp. 121–130.

[161] Zhang, J., Baier, T., and Hueser, M. Instructing an assembly robot in situated natural language and gestures. In *proceedings of the 10th International Conference on Human-Computer Interaction* (Heraklion, Kreta, 2003).

[162] Zhang, J., Baier, T., and Hueser, M. A multimodal interface to situated assembly robot systems. In *proceedings of the IEEE International Conference on Robotics, Intelligent Systems and Signal Processing* (Changsha, China, 2003).

[163] Zhang, J., and Knoll, A. *A General Learning Approach to Visually Guided 3D Positioning and Pose Control of Robot Arms.* Springer Verlag, 2003, ch. 15, pp. 417–438.

[164] Zhang, J., and Quoy, M. Advances in robot skill learning. *Robotics and Autonomous Systems 38*, 3-4 (2002), 135–136.

[165] Zhang, J., and Roessler, B. Self-valuing learning and generalization with application in visually guided grasping of complex objects. *Robotics and Autonomous Systems 47* (2004), 117–127.

[166] Zhang, Z. A flexiable new technique for camera calibration. *IEEE Transactionson Pattern Analysis and Machine Intelligence 22*, 11 (2000), 1330–1334.

INDEX

Eidesstattliche Erklärung

Hiertmit erkläre ich an Eides statt, dass ich die vorliegende Dissertation selbst verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Hamburg, den 18.08.2010

Mohammed Elmogy