eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

Advances towards behaviour-based indoor robotic exploration

Ekaitz Jauregi Iztueta

Supervisors:
Elena Lazkano Ortega & Basilio Sierra Araujo

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the Basque Country University as partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science

Donostia, June 7, 2011

ISBN: 978-84-9860-611-9

To Silvia, Udane, Jone and Joxe

Acknowledgements

I have always loved robots, even before having any knowledge about them. Certain persons gave me the opportunity to be introduced into the world of robots, to“play” and “fight” with them, and above all the opportunity to learn and acquire wisdom about these brilliant machines. These wonderful people are Elena Lazkano and Basilio Sierra. Without them I would have never embarked on this journey. Thank you two for giving me the confidence and the opportunity to develop this work. A million thanks also to Itziar Irigoien who supported and helped me specially during the last phase of the research, and to Arantxa Azurmendi who checked the english grammar and made this dissertation readable.

While developing this work I had good moments but also difficult periods in which my family and my girlfriend have always been close. My grandparents Joxepa Arriola and Joxe Ramon Iztueta deserve an special mention. I warmly appreciate their love, support, understanding and kind affection.

I would also like to thank my colleagues, those that helped me to develop this work, to carry out the experiments with the robot and who shared fun moments with me at work and elsewhere.

I should not forget my friends who supported me in good times and hard times, making me forget about work when it was necessary and always sharing great times with me. I am sure I will forget someone if I try to mention them all. Thus, thank you very much to all of you, old friends and new ones made during my trips abroad.

Thank you all.

This thesis would not have been possible without the help of the Basque Government, who awarded me a grant to develop this research.

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

The main topic of this research work belongs to the area of navigating mobile robots, and more specifically to the behaviour-based indoor navigation approach. The starting point is the control architecture integrated in Galtxagorri, a behaviour-based system whose control architecture follows a taxonomy of biological navigation strategies and which is capable of topological navigation. The work detailed in the present dissertation contributes to the improvement of the navigation capabilities in three main aspects:

- Capability to recognise doors: this behaviour is of primary importance in order to enhance the knowledge of the world that the robots will use to achieve goals. Door handle recognition helps the robot to identify doors in places where the camera is not able to view the whole door; a situation typical in office-like environments, which are full of narrow corridors and junctions, and where the distance to doors is too short. In the present research, new approaches to object recognition are presented, applying machine learning paradigms for color based image segmentation and feature extraction algorithms.

- Capability of global localisation: a probabilistic localisation method is applied in a distributed form in order to provide the robot with the ability of self-localisation even if it ignores its starting position or to in case it gets lost.

- Capability of automatically acquiring the topology of the environment for terrain inspection: a new technique to construct a topological map during the exploration phase on an unknown office-like environment is presented. The new approach is based on a typicality test to perform the so called loop-closing action. This test indicates for each new localisation detected by the robot whether it is a new location – and, therefore, it should be added to the map the robot is building – or an already visited one – and therefore, the loop is closed. The developed system proved to be useful also for localisation purposes. A behaviour-based control architecture is used to perform all the experiments.

The main contributions of this research work belong to the areas of computer vision, more precisely to object recognition in images, and to robot localisation and mapping within the behaviour-based frame. The aim, successfully achieved, was to partially increase the level of autonomy of the robots.

The final goal of the European robotics research community could be to let the robot explore, learn and localise new environments – for instance, a house –, and have it perform specific tasks. The work presented here attempts to make a modest contribution in this direction.

Part I

Introduction

Chapter 1

Introduction

Contents

Since the early age of robotics the development of robots has undergone many changes. Starting from what a robot is, what a robot should do, how should it be controlled, up to what the behaviour of robots should be. For the time being, there are not answers to all these questions yet.

The aim of this chapter is to shortly review the evolution of robotics over the years, referring to the different stages robotics has gone through, and to name the main inventors and researchers that have influenced this field.

1.1 Short history of robotics

Robotics is the field – art, science or technique – that develops robots. But what is the meaning of *"Robot"*? The first time the word was used was in 1921. The word "robot" stems from a play *Rossum's Universal Robots* directed by the Czech playwright Karel Capek (1890-1938). Capek derived the word "robot" from the Czech word "robota", which means "forced labor" and the Czech word "robotnik", which means "serf". The theme of the play was about the dehumanisation of men in the technological civilisation. It should be noted that Capek's robots were not mechanical devices but the result of chemical processes (see Figure 1.1).

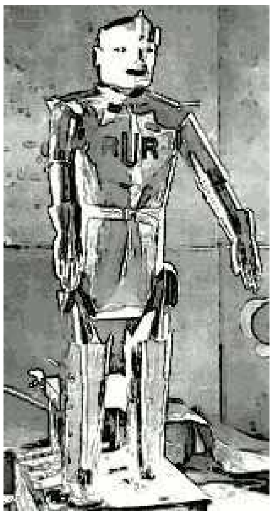


Figure 1.1: Rossum's universal robot

In 1942, the American scientist and writer Isaac Asimov wrote a novel titled "Run-around" where the word "Robotics" was used for the first time. He tried to describe the technology of robots, and also to protect human beings and limit the impact that robots would cause in the future society. To do so, Asimov defined three laws:

1. A robot may not injure a human, or allow a human being to be injured.
2. A robot must follow any order given by a human being that does not conflict with the First Law.
3. A robot must protect itself unless such protection conflicts with the First or Second Laws.

What are robots then? Today there is still no consolidated definition although we are able to recognise a robot when we see one. According to the *Robot institute of America (1979)*, a robot is defined as a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialised devices through various programmed motions for the performance of a variety of tasks.

For the *Japanese Industrial Robot Association (JIRA)*, robots are divided into the following six classes:

Class 1: *manual handling devices.* Devices with several degrees of freedom actuated by an operator.

Class 2: *fixed sequence robots.* Handling devices which perform the successive stages of a task according to a predetermined, unchanging method, which is difficult to modify.

Class 3: *variable sequence robots.* The same type of handling device as in class 2, but the stages can be modified easily.

Class 4: *playback robots.* The human operator performs the task manually by leading or controlling the robot, which records the trajectories. This information is recalled when necessary, and the robot can perform the task in automatic mode.

Class 5: *numerical control robots.* The human operator supplies the robot with a movement program rather than teaching it the task manually.

Class 6: *intelligent robots.* Robots with the means to understand their environment, and the ability to successfully complete a task despite changes in the surrounding conditions under which it is to be performed.

According to the Webster's dictionary, a robot is an automatic device that performs functions normally ascribed to human beings or a machine in the form of a human.

Nowadays most robots are created to perform obligatory work, for example in industry, doing repetitive and fixed tasks. These type of robots are manipulators that operate within a bounded workspace, and cannot move. But robotics is about much more than obligatory labor or manipulators. For most scientists working in this area, robots are *autonomous* systems which exist in the *physical dynamic* world, can *sense the environment*, and can *act on it* to achieve some goals. Hence, robots:

- Need a physical body.
- Need sensors and actuators to interact with the environment.
- Need to be able to take decisions autonomously in the sense that they should have long-term ability to operate without the aid of external operators.

It is not possible to pinpoint where the idea of a “robot” or some type of machine that could help people originated. The first machines of this type we know about were the organs and water clocks with movable figures that the Greek engineer Ctesibius made in the III century BC (Mayr, 1975). However, this inventions are not like the “machine” that we know nowadays, but rather clever mechanical devices.

The first animated machines can be found at the beginning of the eighteenth century. Among them the best known one is Vaucanson’s digesting duck (see figure 1.2). The idea of this French engineer and inventor was to build an automaton that could eat, digest, metabolise and defecate grain. The duck had over 400 moving parts, could flap its wings, lengthen the neck and mimic the footwork of a real duck. This machine followed the principles of Descartes, that is, the belief that animals were mere automata. The same idea would later on dominate the development of robots for years, during the gold age of artificial intelligence.

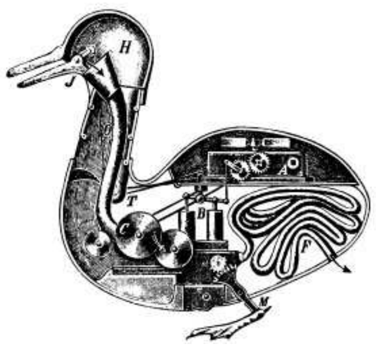


Figure 1.2: The digesting duck

The mathematician Norbert Wiener was the pioneer in the study of stochastic and noise processes, and he worked on electronic engineering, electronic communication, and control systems. In 1947, he made significant contributions to a number of areas of mathematics including harmonic analysis and Fourier transforms, but his most important contribution to robotics was the theory of cybernetics.

1.1.1 Cybernetics

The term cybernetics comes from the Greek word *kybernetes* which means *the art and science of guidance*. Its object of study are the biological systems, from the level of neurons to the level of behaviour. Cybernetics combined theories and principles from neuroscience and biology with those from engineering, with the goal of finding common properties and principles in animals and machines. In cybernetics, the main goal was to produce sophisticated behaviour similar to that found in nature, coupling the mechanism and its environment.

In 1950, the neurophysiologist and robotician Grey Walter, tried to implement the studies of cybernetics in simple robots. Walter was an innovative neurophysiologist interested in the way the brain functions and he developed the first robots of modern robotics, nowadays known as Walter's *turtles* or *tortoises* (see Figure 1.3). Walter gave his turtles Latin names to describe their behaviours, such as *Machina Speculatrix* and *Machina Docilis*. *Speculatrix* means "machine that thinks" and *docilis* means "machine that can be tamed/taught", by which he meant a machine that could learn, because it could be trained with whistles (Walter, 1953).

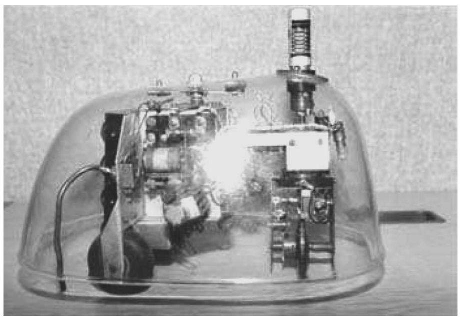


Figure 1.3: Walter's tortoise, the first behaviour-based robot

Walter's idea was to develop a machine with simple reflex behaviours, that would be in constant movement except when recharging, that would be motivated to move towards some environmental object, and would move away from certain negative stimuli and with the ability to distinguish between productive and unproductive behaviour. With this idea in mind, Walter's tortoise consisted of one analogue electronic circuit with two vacuum tubes, one photo cell to detect light levels, one bump sensor, one rechargeable battery, two motors and three wheels in a tricycle-like design. All this things were covered with a clear plastic shell.

The tortoises were endowed with the following behaviours or capabilities:

- Find the light: while the robot was moving exploring the environment, the photocell sensor rotated looking for a weak light source because that was where the recharging station was to be found.
- Head towards the light: once a weak light source was detected, the robot orientation changed heading to the light.
- Back away from bright light: when the light was bright, the tortoise had to repel the light because it was assumed that the robot did not need recharging.
- Turn and push: to avoid obstacles.
- Recharge the battery: when the robot had low battery the light became strong so that the robot could perceive and move towards the recharging station.

But the most impressive behaviour these turtles showed was not *programmed*. The battery level changed the behaviour of the light sensor. When the battery charge level went down, high intensity light sources were perceived as weak and hence, the robot was attracted by the light source with which the charge station was signalled. This side effect produced what is named as *emergent behaviour*, the robot was able to decide when it needed to recharge and when to avoid the bright light just by interacting with the environment.

To control these different behaviours a decision must be made about the rules that should be executed at each stage. Robots built by Walter, acted on the highest priority behaviour applicable, a method that is nowadays named *reactive control*. When properly combined, these simple rules resulted in animal-like behaviour.

Another important neuroscientist and cyberneticist who was inspired by Grey Walter's work is Valentino Braitenberg. He revived the tradition of Walter, proposing behaviour-based robotic systems three decades after him. In 1984, Braitenberg wrote a book titled *Vehicles: Experiments in Synthetic Psychology* (Braitenberg, 1984) describing a series of ideas to develop machines with animal-like behaviour. He used his knowledge about how the brain and neurons work to show how to design simple robots. Based on how neurons operate, he tried to develop systems using inhibitory and excitatory connections directly coupling sensors to actuators.

These robots/vehicles were simple agents built with motors and sensors. Connecting directly both components, it is possible to achieve excitatory or inhibitory behaviours. For example, using an *excitatory connection*, the stronger the light, the faster the robot will move showing a phototaxis behaviour. On the contrary, an *inhibitory connection* would result on slow movements upon strong input values and photofobic behaviour. Increasing the number (and type) of sensors and actuators and combining the connections, more sophisticated robots could be developed (see Figure 1.4).

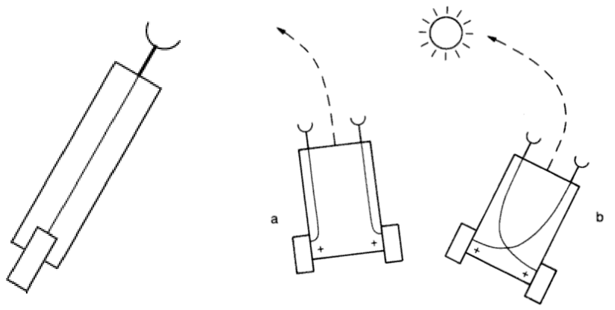


Figure 1.4: Braitenberg's vehicles

Although Braitenberg never gave expression to his ideas on physically embodied agents in the sense that he only proposed ideas and descriptions to develop machines, his book has been a source of inspiration for many roboticians. For instance, in 1991, scientists at MIT's Media Lab (Hogg et al., 1991) developed some of the *Braitenberg Vehicles* using LEGO technology.

For the cybernetics field, the environment and the organism were linked in the sense that the field covered all about control theory, information science and biology. But at the beginning of the 50s, these topics were split into different areas, giving rise to a new age of robotics.

1.1.2 Artificial Intelligence

In the age of cybernetics, the scientists tried to combine what were considered the building bricks of robots, i.e. *sensing*, *thinking* and *acting* processes, and the interaction with the environment. But after the 50s, this started to change.

In 1950 Alan Turing wrote an article titled *Computing machinery and intelligence* (Turing, 1950) presenting a test to determine the intelligence of a machine: *the Turing test*. The test proposed a game named *The imitation* *game*, that worked as follows: The game was set up in an environment with two rooms. In one of them, a person and a computer were placed and in the other one, just a person. The person who was alone had to determine if he/she was talking with the person or the machine in the other room. The conversation should happen using natural language and the computer had to use syntactic rules to manipulate symbol strings to communicate. The capability of the person to differentiate among the person and the machine in the other room would determine the intelligent behaviour of the machine. Turing ideas produced different reactions among the research community. Probably, his best known detractor was Jon Searle who argued that being able to write Chinese sentences does not imply to understand Chinese language (Searle, 1980). The *Chinese room argument* claimed that someone who knows only English sitting alone in a room and following English instructions for manipulating strings of Chinese characters may be able to make someone else believe that he understands Chinese. The argument was intended to show that, while suitably programmed, computers may appear to converse in natural language although they are not capable of understanding language. Searle argued that Turing's experiment underscored the fact that computers merely use syntactic rules to manipulate symbol strings, but have no understanding of meaning or semantics.

Anyhow, more and more scientists from different fields began to research in the field trying to develop intelligent machines. The goal of *intelligence* took more and more strength and become the highest priority target. In 1955 the most prominent researchers of the date, John McCarthy, Marvin Minsky, Allan Newell and Herbert Simon, met at a conference held at Dartmouth University, in Hanover. The goal of the meeting was to discuss the development of intelligence in machines. In that meeting the field of *Artificial Intelligence* (AI) was born and the decline of cybernetics started. The conclusion of the meeting, which was that *reasoning* was needed, would mark the future of robotics for the following thirty years.

Until then, cybernetics had effectively combined "thinking", "acting" and the "interaction" with the environment, but in the Dartmouth conference this entire field split between the fields of AI and robotics. From then on AI intended to provide robots with a learning brain and intelligence, whereas the scope of robotics was relegated to manipulation. Natural language, machine learning and a symbolic representation of knowledge became the AI main topics, topics always associated to higher levels of intelligence, and which do not have a dependency on physical bodies or on the environment.

The researcher of AI had to create systems that could efficiently solve problems and that could learn by themselves. The main areas of AI were neural nets, complexity theory, self-improvement, abstractions and creativity. Dartmouth conference conclusions summarised what intelligent machines need to use (Matarić, 2009):

- Internal models.
- Search the solution space.
- Planning and reasoning to solve problems.
- Hierarchical system organisation.
- Sequential program execution.

Robotics researchers developed robots based on these ideas for the next thirty years. One of the most popular AI-inspired robots was *Shakey* (see Figure 1.5). It was built at the Stanford Research Institute, California in 1966. *Shakey* was made up of a TV camera, a triangulating range finder, and bump sensors. This robot communicated with two computers where the perceptions and the solutions were explored and analysed. *Shakey* lived in a very special indoor world specifically made for it. The environment consisted of a white floor and some large black objects, such as balls and pyramids. *Shakey*'s task was to navigate from one room to another, avoiding obstacles autonomously and pushing boxes from one side to another. It used programs for perception and the computers created plans to move in that special world, communicating with *Shakey* via logic predicates.

Since AI focused on reasoning, the robots were not based on reactive control. AI-inspired robots used what later on was named *deliberative control*, a type of control focused on *planning* actions, which was derived from the decomposition of intelligence made by AI.

Another instance of AI-inspired robot was *Cart* (see Figure 1.6), developed at Standford University and entrusted by the National Aeronautics and Space Administration (NASA) in 1966 with the goal of studying the possibility of sending it to the moon controlled by radio signals. The project was carried out and became the research tool of graduate students during the 70s. John McCarthy and Rodney Schmidt started to think about unmanned guided vehicles and some years later *Cart* was able to follow a white line painted on the road for about 6 meters using a TV camera mounted on its top. In 1977 *Cart* was inherited by Hans Moravec and he focused on the development of a precise 3D representation of the environment using images.

An image can contain a lot of information, but the process of extracting useful information from vision sensors is computationally expensive and even more so with the computers available at the time. Given the technology of

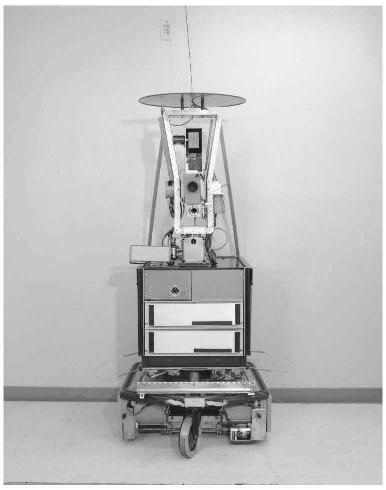


Figure 1.5: The robot *Shakey*

that time, *Cart* took about 15 minutes to process each image. The robot took nine analogue pictures to obtain information and learn about the environment. First the images had to be digitalised, which took five minutes. Then, five more minutes were needed to examine the images, and finally another five minutes were required for the maintenance of the environment model and to plan the appropriate path.

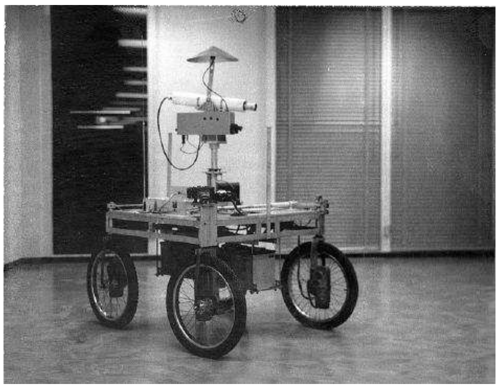


Figure 1.6: *Cart*, the car-like robot on bicycle wheels

The *Cart* and *Shakey* examples showed that the separation and treatment of each component individually, i.e. reasoning and manipulation, hindered the progress of robotics in both fields. The few real robots developed in the age of AI thought too hard but acted slowly and failed to move effectively and responsively. This fact brought some roboticians to think that intelligence on robots was not the exclusive feature to focus on. Something was missing in the approach taken by AI.

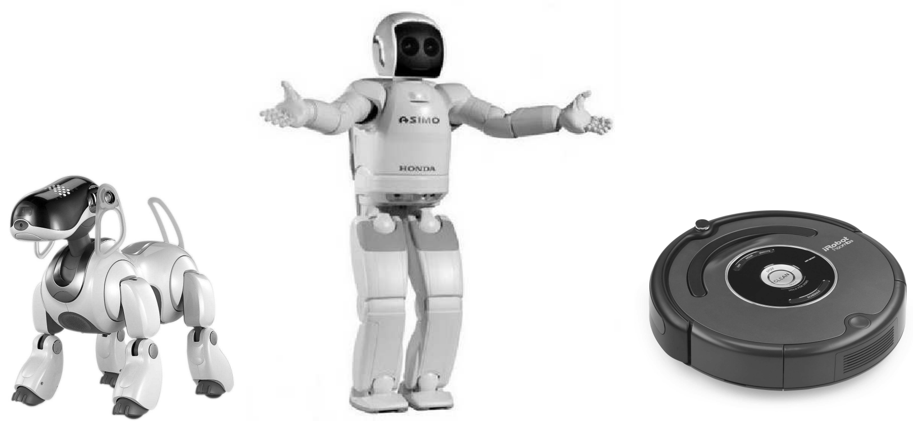
The rationale and focus of AI started to have an increasing number of detractors. It was at this point that the new era of behaviour-based (BB) machines emerged. In 1986 R. A. Brooks, nowadays considered one of the founders of modern robotics, proposed a new way of developing intelligent creatures together with a new control architecture for robots. Instead of a top-down decomposition, he proposed a bottom-up methodology for building robots. Agents with sensing and acting capabilities should be developed and behaviours should be integrated incrementally. Intelligent behaviour should emerge from the interaction between the agent and the environment. Brooks founded *behaviour-based robotics* and built a great number of robots. As the name suggests, BB systems were the technical counterpart of the philosophy of the Psychology school born in the early twentieth century named *behaviourism*. According to this school, behaviours as such can be described scientifically without recourse neither to internal psychological events nor to hypothetical constructs such as the mind. The first contribution of behaviourist researchers was to consider animals as intelligent creatures (Jennings, 1906), what led them to carry out experiments with simple animals as a means to better understand human intelligence.

Brooks noted that to develop humanoids, physical robots capable of interacting in real environments were to be developed. These machines should first show basic abilities. As behaviourism did, Brooks also claimed that intelligent behaviour was generated by main coupling perceptions and actions without the need of hard reasoning.

In 1997, the *NASA* agency, responsible for the space program, aeronautics and aerospace research, sent the first robot, named *Mars pathfinder Sojouner*, to Mars with the idea of conducting experiments on the Martian surface. At the same time, the Japanese company HONDA developed the first humanoid called *P3*, and a few years after, it presented *Asimo*, an evolution of the *P3* humanoid (see Figure 1.7(b)). This robot could run, walk on uneven slopes and surfaces, turn smoothly and climb stairs, but was mainly teleoperated and showed a low level of autonomy.

In 1999, Sony presented the first robot for entertainment called *Aibo* (Figure 1.7(a)). This dog-shaped robot imitated dog behaviour and was able to interact with human beings. With the passing of time, Sony introduced new features and enhancements in this robot. Finally, in 2003, the company presented the third version of this dog like robot.

During this first decade of the XXIth century, more and more robots are being developed. In 2002, the iRobot corporation – Co-founded by Brooks – presented the intelligent vacuum cleaner called *Roomba* (see Figure 1.7(c)), one of the most famous robots of today. Although created for everyday use, this intelligent robot is just now becoming a common artifact at home.



(a) Aibo (b) Asimo (c) Roomba

Figure 1.7: Modern robots

Robot contests are becoming popular and they contribute to the speeding up of the evolution of robots. Among them it is worth mentioning the DARPA challenge, the first autonomous vehicle long distance competition celebrated for the first time in 2004. In this competition autonomous vehicles must travel a long distance with the only help of the sensors they are equipped with. The first competition was held in the Mojave Desert and the third edition took place in George Air Force Base in 2007. This competition is open, anyone can develop their car and take part in it. Basically, the idea is to develop autonomous vehicles to introduce advances in real life reliably. Thanks to the achievements in this race there are now vehicles that park or slow down by themselves.

Gradually, more robots are being designed. From 2004 to date several humanoid robots have been developed capable of interacting with human beings using human-like ways of interaction. There are many companies that are dedicated to building more realistic humanoids with behaviours such as talking, singing, cooking, and much more. The robot *Sacarino*, developed by *Cartif* and University of Vigo is already working as a bellboy in Valladolid.

1.2 Applications of robots

As mentioned, many robots have been developed with very different functionalities. All these robots are different in shape, some of them are walking machines and most of them wheeled vehicles. Although they show different capabilities they are still far from being like the robots depicted in futurist movies and books. With the advance of technology, the world of robotics is growing and may well develop robots that were once unthinkable.

One of the first areas to introduce robotics was industry. As stated earlier, industrial robots are generally manipulators. These mechanical arms, thanks to their high precision and speed of movements, are used in almost all industries, mainly in those engaged in assembly lines. In this sector, robots can be deployed in applications such as manipulation of parts, application of pieces or material, quality control, welding, etc. They are also very useful in industrial cleaning or in agriculture. However, the usefulness of robots is growing and more and more applications are being developed in which robots will help human beings in our everyday life. Some of these applications are listed bellow:

- Interplanetary exploration: human beings always have had interest in the space like G. Galilei, N. Copérnico, J. Kepler etc, but there was not sufficient technology to explore it. In 1969 man landed on the moon, but since then humans have not travelled any further. The conditions of other planets, such as extreme temperatures, radiation, gravity and inaccessibility has made it impossible. Ideas to assemble space structures by a flying robot in orbit or to conduct servicing missions to existing satellites have been discussed. The latest achievements of the NASA agency was to reach Mars. Obviously, it was not reached by human beings but by two robots named *Spirit* and *Opportunity* which landed on Mars in 2004 to obtain information on this planet. Thanks to them something more about this planet is known (Aeronautics and Administration, 2004).

- Undersea exploration: in the same vein but with very different challenges, yet the sea is little explored by human beings. Nearly 70% of the planet's surface is covered by water, but the knowledge of the deep sea is very limited. For example, it is known that the flow of water has a lot in common with the weather on planet earth and the causes of climate change, increasingly is being given more and more importance to knowledge of the deep-ocean. One of the latest robots developed for this purpose is called *Benthic Rover*. Thanks to this robot new insights into the deep-ocean have been obtained (Henthorn et al., 2010).

- Demining: robots are being increasingly used to identify and remove mines which are left behind after wars end, but which do not loose their explosive force and remain a real threat for populations around the world. There is an estimate of 110 million mines scattered across the land, mostly in Africa. Their identification, deactivation and removal is extremely important, but very dangerous for human beings. For this purpose several robots, such as *iRobot's PackBot* (Kaneko et al., 2010) have been developed.

- Health care/Medical assistance: the use of robots has also reached the area of medicine, for example in surgery. Thanks to the precision of robotic arms long-distance surgery is being made possible. In 2001 took place the first long-distance operation with the robot *ZEUS*, developed by *Computer Motion*. The patient was operated in France while the team of surgeons was in New York. Currently there are several types of surgical robots, as robots servo-workers, assistants, coordinators, semi-autonomous and remote control (Sánchez Martín et al., 2007).

- Elderly people assistance: as life expectancy increases, this becomes probably one of the most interesting applications. On the one hand, societies are getting older and young people do not look after the elderly. On the other hand, people prefer to live at home as long as possible instead of being institutionalised in sheltered accommodation when problems related to ageing appear. That is why the idea of developing assistive robotics devices like autonomous wheel chairs or companion robots has become attractive. *Paro seal* and *Huggable bear* are two examples of companion robots.

- Education: robotics has received a great deal of attention as a way of motivating students to learn science and engineering (Druid and Hendler, 2000). Based on J. Piaget and S. Papert theories about constructivism, several educational robotics products have been created and are nowadays widely used in schools and universities, such as LEGO Minstorms, Fischertechnik, Logo etc. In the education field, robots make easier the learning and the strengthening of cognitive skills (González and Jiménez, 2009).

- Entertainment: many robots have been created for entertainment, but possibly the most famous of them is the dog robot AIBO, able to imitate actions and movements similar to real pets. There are more entertainment robots like *Pleo*[1], a robot with the appearance of a small Camarasaurus originally designed by Caleb Chung.

[1]Innvo Labs, http://www.pleoworld.com/Home.aspx

Chapter 2

Structure of this dissertation

The brief history of robotics described in the previous chapter was meant to introduce the reader to the state of the art of robots in general and in particular to the area of mobile robotics. Before continuing, it is worth to shortly review the structure of the dissertation to facilitate the comprehension of the text.

The dissertation is structured in five parts that can be read independently:

- Part I, *Introduction*.
- Part II, *Putting the reader in the picture* aims to contextualise the developed research work.

  It is composed of three chapters:

  – Chapter 3 describes Galtxagorri and Tartalo, the two robot platforms used for the empirical validation of the different methods proposed. Together with the characteristics of the sensors the robots are equipped with and their morphological structure and computational capabilities, the software tools used for reading and commanding actions to the hardware elements and for developing higher level behavioural actuation are explained.

  – Chapter 4 reviews the different paradigms for developing robot control architectures and emphasises on behaviour-based systems that are the philosophy adopted for the navigating control architecture being developed by the author.

  – Chapter 5 summarises the behaviour-based navigation system of Galtxagorri together with its predecessor *Toto*. The procedural topological description of the environment used in these two robots form the starting point of the consequent behavioural layers developed within this research work. The three main shortcomings of Galtxagorri's architecture that are faced in the present research work are identified.

- Part III, *Approaches to door handle identification*, describes the several approaches developed for obtaining vision based door identification behaviour. Three chapters are deployed:
  - Chapter 6 introduces the problem of door identification, justifies the need of such behavioural module and reviews the literature.
  - Chapter 7 reviews the image processing methods used for developing the different approaches.
  - Chapter 8 describes the proposed methods that are validated through a strong empirical phase. These approaches are also compared among them to select the most appropriate one to be integrated in the future in the navigating control architecture. Experiments performed in Tartalo validate the proposed approaches.
- Part IV, *Behaviour-based localisation and mapping*, describes the steps given for further develop the navigating behaviour-based architecture integrated in Galtxagorri.

  This part is composed of two chapters:
  - Chapter 9 describes the effort to integrate a probabilistic localisation system to the behaviour-based navigating control architecture. Experiments are executed in simulation as well as using a the robot Galtxagorri.
  - Chapter 10 explains how an automatic mapping behaviour is implemented as a typicality approach. Again, experiments using a simulator serve for validating the approach and afterwards, the mapping system is tested within the real robot environment system using the robot Tartalo.
- Part V: Conclusions and open research lines.
  - Chapter 11 summarises the developed work, giving the advantages, shortcomings and obtained conclusions and outlining the open research lines that remain open to continue working on.

Part II

Putting the reader in the picture

Chapter 3

Robot platforms and software

Contents

As discussed in Chapter 1, robots are autonomous systems which live in the real world, they can get information about the world, and are able to interact with it to reach goals. Holding to this definition, autonomous robots need a physical body, sensors and actuators to interact with the environment.

The tasks that can be developed or the goals that can be reached by robots depend on the set of information or stimuli they can perceive and the actuators they are provided with. The morphology of the robot and the type of sensors and actuators determine the interaction of the robot with the environment.

This chapter describes the mobile robot platforms that have been used in this research work. The different sensors available and their functionality are described together with the software used to access the different devices and to develop the control architecture.

3.1 Robot platforms

There are many types of mobile platforms or robots, which are usually specifically designed for the environments where the robot is supposed to work in. There are many different robots with very peculiar morphologies such as animal-shaped robots, humanoids or dollies. Apart from the morphology, robots differ specially in the type of motion system they use. That is to say, robots that move underwater or robots that fly use different actuators than those that move on the ground. At the same time, robots designed to live in indoors use simpler motion systems than those that move outdoors.

The *Pioneer* platform was introduced by *ActiveMedia Robotics* – now *MobileRobots*[1]– in 1995, and since then it has undergone several changes. With the advance of technology, *MobileRobots* has made improvements in several components and has developed different types of robots to adapt to all types of terrain. Within the *Pioneers* platforms family, the 3-DX and 3-AT types can be found. The former has two differential wheels and one caster, being suitable for indoor environments. The latter has four wheel steer, being suitable for outdoor environments where the surface is not flat. Given the limited space inside these robots, the on board computers are usually special and not as powerful as those available on the market. For this reason, depending on how the robot has to process data, many researchers choose to treat the information in an external computer and to send it to the robot after doing the necessary calculations. Thereby, the *Pioneers* are equipped with mechanisms to communicate with an external computer such as a wifi antenna and an Ethernet cable connection.

[1]http://www.mobilerobots.com/Mobile_Robots.aspx

The Robotics and Autonomous System group of the Basque Country University owns several robots (Figure 3.1), all of them adapted for indoor environments with flat floors and regular structures. The two platforms that have been used in the present research work were *Pioneer* robots provided by *MobileRobots*: a *Pioneer 3-DX* named Galtxagorri and a *PeopleBot* named Tartalo.

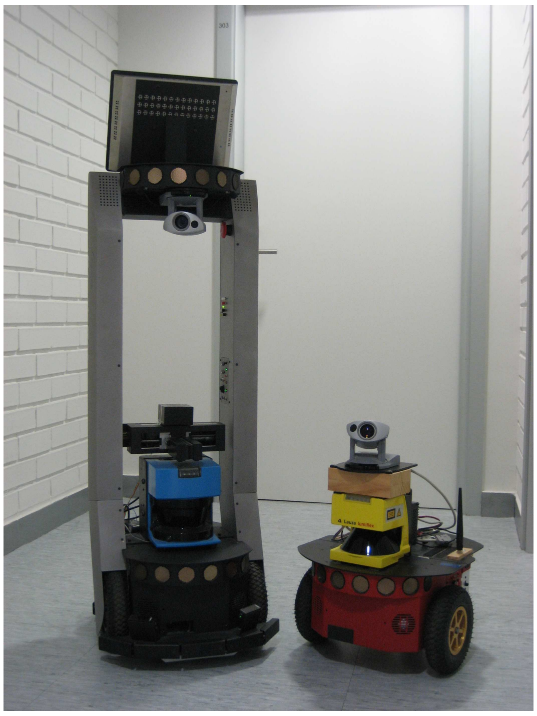
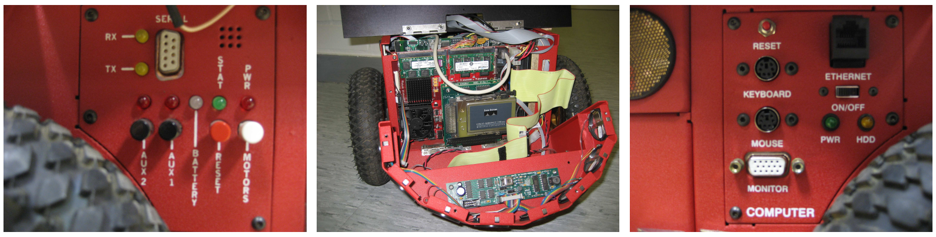


Figure 3.1: Tartalo and Galtxagorri

Galtxagorri is a *Pioneer* robot, model 3-DX, a very common platform within the mobile robotics research area. This *Pioneer* 3-DX platform is fully assembled with motors with 500-tick encoders, 19 cm diameter wheels and tough aluminium body. With a height of 23.7 cm, a width of 38.1 cm, a length of 48.5 cm and 9 Kg weight without accessories, it can reach speeds of about 1.4 m/s and carry a payload of up to 25 Kg. Inside the platform there are three batteries of 12V that provide energy to the various electronic components. On one of the outer sides of the platform there is a panel that provides status information of the robot through bright LEDs, such as the charge level of the batteries and the hard disk status (see Figure 3.2(a)). On the opposite side, there is a similar panel with connectors for a keyboard, a mouse, a monitor, and a 10/100Base-T Ethernet connection to the internal PC (see Figure 3.2(c)).



(a) P3-DX left panel (b) *Pioneer* 3-DX insides (c) P3-DX right panel

Figure 3.2: *Pioneer* 3-DX robot's views

This *Pioneer* 3-DX robot uses a Hitachi H8S micro-controller. Using ARCOS (Advanced Robotics Control and Operations) control system software, the H8S manages all the low level details of the mobile robot, sending commands and reporting the state of the devices attached to it to the client applications. To this end, the controller has a 32K RAM memory and an additional 128K flash memory which stores the parameters of the robot. The H8S controller contains several input/output ports to connect additional sensors or devices and also a serial plug to connect the controller to the on board computer (see Figure 3.2(b)). Galtxagorri was originally provided with a Pentium III 850 MHz computer. This computer was replaced with a 1.6 GHz Intel Pentium M on board computer with 1GB RAM memory to increase the on board computational capability of the robot.

Galtxagorri is equipped with 16 ultrasound sensors placed in two sensor rings at the front and at the rear, a Canon PTZ colour camera, a Leuze RS4 laser sensor and a TCM2 electronic compass.

Tartalo is a *PeopleBot* robot built on the P3-DX base. More specifically, Tartalo is a *Pioneer* robot with a chest-level extension that makes it taller and adequate for human-robot interaction. With a height of 112 cm, a width 38 cm, a length of 47 cm and 12 Kg weight without accessories, it can reach speeds of about 0.8 m/s and carry a payload of 13 Kg. *PeopleBot* robots have a SH2 micro-controller to communicate with the embedded sensors and the on board computer. The base of the *PeopleBot* has been extended with front and rear bumper rings and infrared sensors pointing upwards and configured to detect tables so that the robot can grasp objects if a gripper is also available. As mentioned earlier, the chest-level extension makes *PeopleBot* robots more suitable for interacting with people. At the top of the chest, it is possible to place a touchscreen as a device to command inputs to the robot. This screen is connected to the on board computer and its location is chosen to facilitate the communication between the user and the robot. For the same reason the robot is equipped with speakers and a microphone. Like the *Pioneer* platforms, *PeopleBot* robots have connectors to add different devices such as a gripper, laser scanners, cameras, an electronic compass, etc.

Tartalo has an on board computer, originally a Pentium III 850 MHz upgraded to an 1.8 GHz Intel Pentium M with 1 GB RAM memory and is additionally provided with front and rear sonar and bumper sensors, a gripper with two degrees of freedom, a SICK laser scanner, a Canon PTZ colour camera, a TCM2 compass sensor and a touchscreen on the top.

3.2 Robot sensors

Although robots can be used for different ends, they are all supplied with sensors.

Sensors are physical devices that measure physical quantities, and that allow the robot to perceive and measure environmental properties, such as distance, size, touch etc. This information, though, is sometimes confusing or noisy, in the sense that the measurements are not accurate. To obtain an improved accuracy of the gathered information robots are normally provided with multiple sensors.

The next subsections describe the main sensors available on the two robot platforms previously described and commonly used within the indoor mobile robot navigation area.

3.2.1 Bumpers

Bumpers are digital passive sensors that do not need to send any stimulus to be activated. Bumpers can be used to detect collisions that have already occurred, they can not anticipate contact with objects and hence, they are very useful when other alternative obstacle avoiding mechanisms fail. They are also used to detect limits in manipulators. Of the two robots used in this research, only Tartalo is equipped with bumpers. It has front and rear belts with five bumper sensors on each.

3.2.2 Infrared sensors

Infrared (IR) sensors are active sensors that emit a wave of infrared light and collect the intensity of the reflected light. The infrared light emitted by the IR sensor is usually modulated at a very low frequency (100Hz) in order to distinguish the emitted IR signal from the light emitted by a fluorescent lamp or by the sun (Nehmzow, 1999). For these sensors to give accurate measurements, the objects that are in the environment must have a uniform surface and colour. However, in real environments objects are of different colours and materials, and some are particularly difficult to be perceived. For example, in the case of dark objects the intensity of the reflected light is inversely proportional to the square of the distance, and they are very difficult to detect as it can be seen in Figure 3.3.

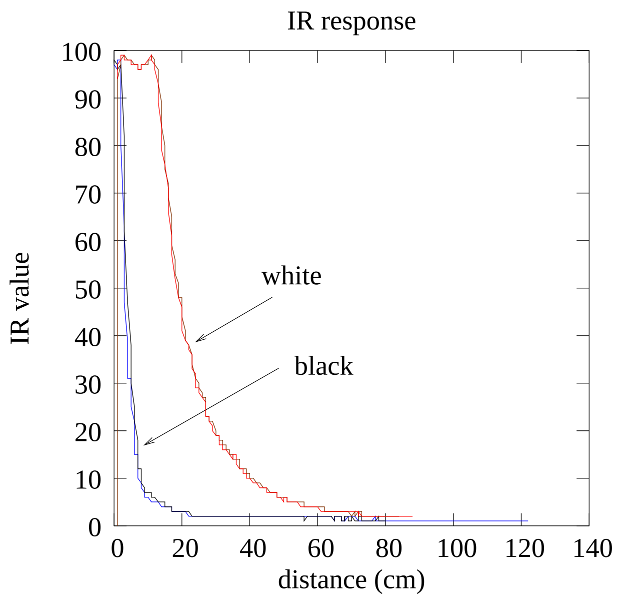


Figure 3.3: IR measurements in different surfaces

IR sensors are used as break-beam sensors in gripper like manipulators, to detect the presence of obstacles not perceived by larger range sensors and to detect irregularities such as steps.

Tartalo is equipped with IR sensors. Twin fixed-field IR point up and slightly forward from the front corners of the base to sense the underside of tables. The location and orientation of these IR sensors make it possible to detect tabletops or rope barriers. Besides, the gripper has infrareds located inside the fingers to detect that something is in between them ready to be grasped (Figure 3.4).

Drawbacks

Infrared sensors have a limited range of detection. They can detect the presence of obstacles at a distance of 5 cm and up to 100 cm.

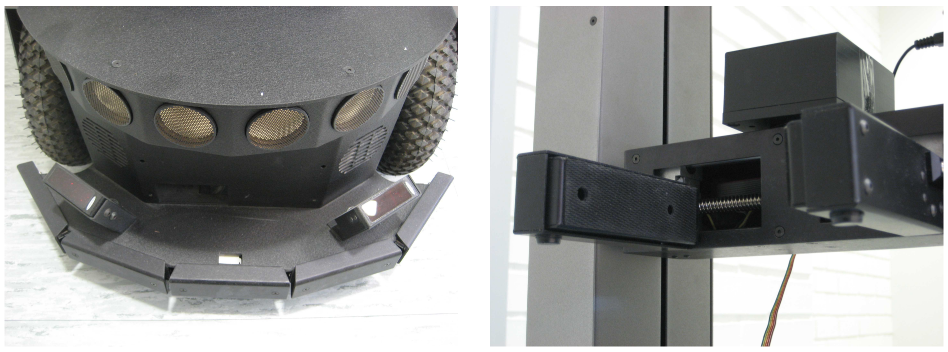

(a) Infrareds on the base (b) Infrareds in the gripper

Figure 3.4: IR sensors in Tartalo

Moreover, these sensors do not measure the distance to the object. The received signal is usually thresholded to decide the presence or absence of objects. Therefore, these sensors are rarely used alone for obstacle avoidance, they are normally combined with larger range sensors.

Infrared light is absorbed by dark colours and hence, dark surfaces are difficult to detect. Same problem arises with glass made objects or other reflecting surfaces.

3.2.3 Sonar sensors

The sonar device (*SOund NAvigation and Ranging*) is used to measure distances to elements in the environment. The ultrasonic sensors are time of flight (TOF) sensors. These sensors measure the time that a pulse of energy needs to travel to a reflecting object and return to the receiver. The distance is calculated by multiplying the speed of the wave energy (in the case of ultrasound, the velocity of sound, v_s) by the time required to travel. This value is divided by two, as the flight time of the signal corresponds to the time required to get there and return: $d = \frac{v_s \times t}{2}$.

Both robots, Galtxagorri and Tartalo are equipped with front and rear sonar rings at the base, with a total of eight ultrasonic sensors in each ring. Additionally, Tartalo includes eight more sonar sensors on the top of the deck. The sonar ranging acquisition rate is adjustable, and set to 25 Hz by default. These sensors are configured to detect obstacles at distances from 15 cm to 5 meters.

Drawbacks

Although the speed of sound is set to 341.000 km/s in an atmosphere of 20°C, depending on the humidity and temperature, the speed of sound changes thus affecting the measures obtained by ultrasounds. For instance, at 16°C of temperature, the error in the measurement is of the order of 3% (Nehmzow, 1999).

The sensitivity of a sonar sensor is cone shaped (see Figure 3.5), and it is not possible to know the exact position of an object detected at distance d, which could be anywhere within the sonar's cone, or on an arc of distance d from the robot (Everett, 1995).

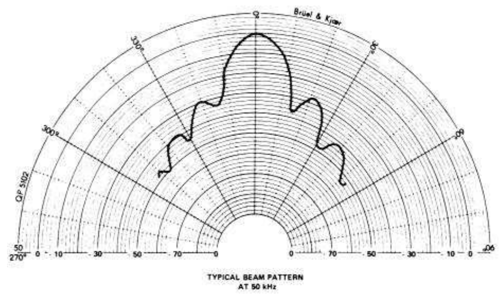


Figure 3.5: Approximation of cone shape measurements

Bouncing sound waves in the environment cause what is known as *specular reflections*. These imprecise measurements are caused by firings that hit objects with a smooth surface at shallow angles. The fired sound wave bounces off another object and returns to the sonar receiver, giving an erroneous distance measurement. Specular reflections produce overestimated distances and hence, the robot may leave potentially dangerous situations undetected. Figure 3.6 shows this effect, the measured distance to the object will approximately be $d_1 + d_2 + d_3$ instead of d_1.

3.2.4 Laser range finders

Laser (*Light Amplification by Stimulated Emission of Radiation*) sensors are TOF devices used to measure distances to objects in the environment. But instead of sending a sound pulse, these devices emit light through a process of optical amplification based on stimulated emission of photons. Laser energy is emitted in a rapid sequence of short bursts aimed directly at the object within range (Borenstein et al., 1996).

Laser sensors emit light beams at different angles covering a wide range. They make use of a rotating mirror to change the angle of the emitted light.

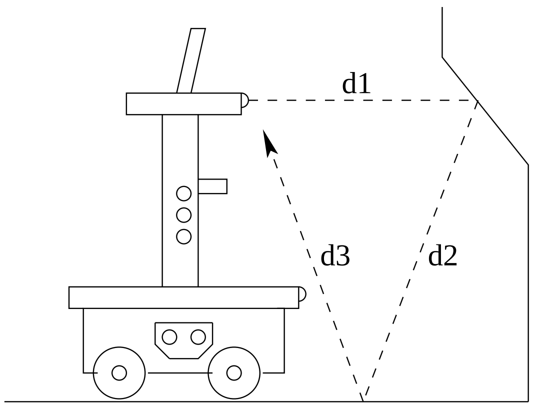


Figure 3.6: Specular reflections

In this way, instead of a single reading, laser devices give scans of readings (see Figure 3.7).

The high directionality of the laser light makes it a controlled beam and specular reflections are avoided. Moreover, the lasers' range is much larger than that of sonars although the maximum range is normally configured by software and set to shorter values because indoor environments do not require such long ranges as outdoor ones.

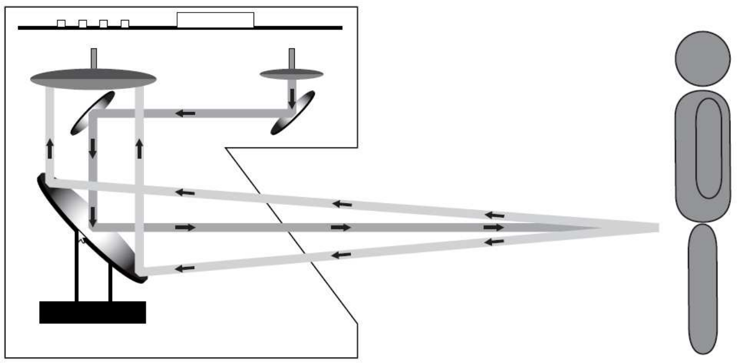


Figure 3.7: Functionality of the laser

Tartalo and Galtxagorri are equipped with lasers from different manufacturers. The *Pioneer* 3-DX has a *Leuze RS4* laser (Figure 3.8(a)) and the PeopleBoot a more common *Sick LMS* laser (Figure 3.8(b)).

The Leuze RS4 laser has an angular range of 190°, and an angular resolution of 0.36°. Its maximum registration of measurements values is 50 meters. The Leuze laser is fed by an additional 12V battery placed on the base plate via a 12V to 24V *Mascot 8862* converter.

(a) Leuze laser

(b) Sick laser

Figure 3.8: Lasers from different manufacturers

On the other hand, the Sick laser has an angular range of 180°, a maximum range of 80 meters and the angular resolution can be configured to 0.25°, 0.5° or 1°, firing a beam of light each 13.3 ms (75Hz). This laser is fed from the internal batteries of the robot.

Drawbacks

Laser sensors are nowadays becoming essential in navigating indoor and outdoor robots albeit they also present some flaws. Glasses and mirrored surfaces do not reflect the light and hence, can not be detected by these powerful sensors.

They are generally expensive although cheaper (and less accurate) models are being manufactured. Moreover, their energy consumption is high and this reduces the level of autonomy of the robot. Besides, their weight and size makes it difficult to place them on smaller robots.

3.2.5 Odometry

The odometry system is the set of sensors and hardware necessary for measuring the horizontal displacement of the drive wheels and thus, for measuring the horizontal displacement and change in orientation of the robot. There are two ways to measure it (Arkin, 1998):

1. Using *shaft encoders*: these are devices mounted on the rotating shaft of the wheels. Odometers consist of binary coded discs and optical sensors that calculate the shaft's position. Once the positions is calculated it is possible to measure the displacement of the robot.

2. Using inertial navigation systems: instead of calculating the displacement of the robot based on the rotation of the wheels, these systems measure the acceleration of the robot. This method is more precise but more expensive, and that is why shaft encoders are more widely used in indoor robots.

Both, Galtxagorri and Tartalo are equipped with two quadrature encoders with a resolution of 500ticks per revolution.

Drawbacks

Odometry is based on the assumption that wheel revolutions can be translated into linear displacement relative to the ground. However, the position obtained by the odometer is very unreliable because it is affected by two types of errors (Borenstein and Feng, 1996):

1. Systematic errors: these errors are due to inaccurate values of the robot parameters, such as wheel diameter or wheelbase length. These errors can be measured and calibrated.

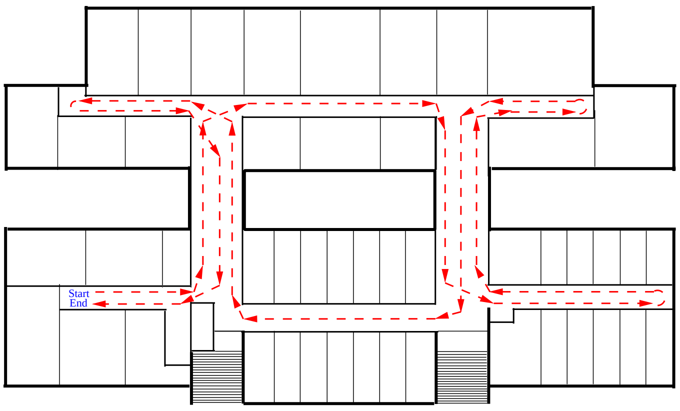
2. Non systematic errors: these errors are the result of wheel slippage produced by acceleration/deceleration of the robot, and by bounces produced by uneven floor. These effects occur randomly and hence, cannot be corrected.

Moreover, since the estimation of the robot position is based on the previous state, the odometry error is accumulative. Figure 3.9 shows the importance of this error. The robot path drawn appears completely out of phase so it is impossible to distinguish the path executed by the robot.

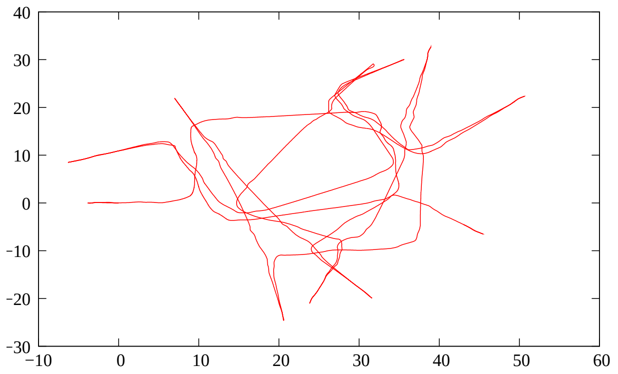
Odometry error can be said to be the main problem faced by mobile robotics nowadays, and it has deeply influenced the approaches taken to develop control techniques for robot navigation.

3.2.6 Electronic compass

There is a wide range of mechanisms for obtaining the angular orientation with respect to the earth's magnetic field, such as magnetoresistive compasses or magnetoelastic compasses (Borenstein et al., 1996). Tartalo is equipped with a TCM2 sensor which besides being a compass, it is also a inclinometer, a magnetometer and a thermometer. Each compass measurement consists of a number of samples of x and y pulses which are the horizontal components



(a) Nominal path



(b) Path drawn using raw odometry data

Figure 3.9: Cumulative odometry error

of earth's magnetic field. These measures are digitised and sent to the micro-processor, which uses algorithms to translate them into precise information about the orientation.

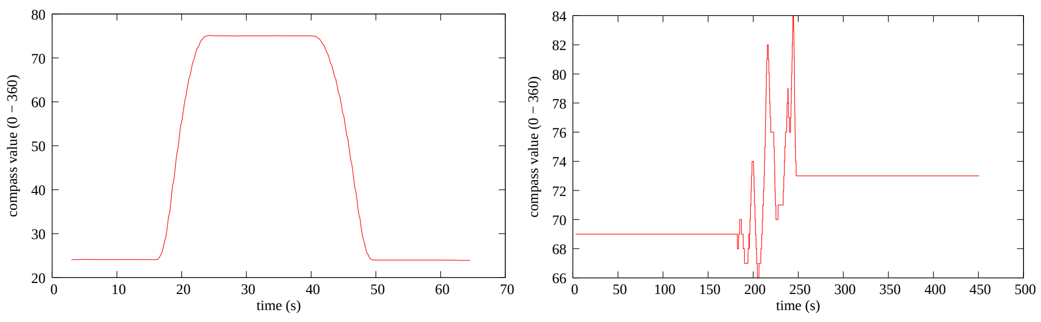
The TCM2 sensor is connected to one of the serial ports of the on board computer.

Drawbacks

Although compasses are designed to avoid the variations they suffer in their readings, caused by electrical fields or metal structures found in the environment, the errors in the measurements they provide are still significant.

The structure and components of the robot must be taken into account and it is necessary to place the compass in a proper position so that the readings are as accurate as possible.

Figure 3.10(a) shows an example of the error produced by the robot components. Plotted data readings were previously low filtered and converted to integers to remove noise. The compass sensor was hold by the gripper and the gripper was commanded to move upwards to its limit and downwards again. The microphone and speakers situated on the desk at the top of the chest produced a deviation of 50° in the compass reading. On the other hand, Figure 3.10(b) shows the compass reading during a forward path of the robot drove using a joystick. On its way, the robot passed by an electronic cupboard and although it maintained a constant heading, the readings oscillated considerably, and the initial and final orientations differed in about 15 degrees.



(a) Compass variation due the robots magnetic field (b) Compass variation due metal structures

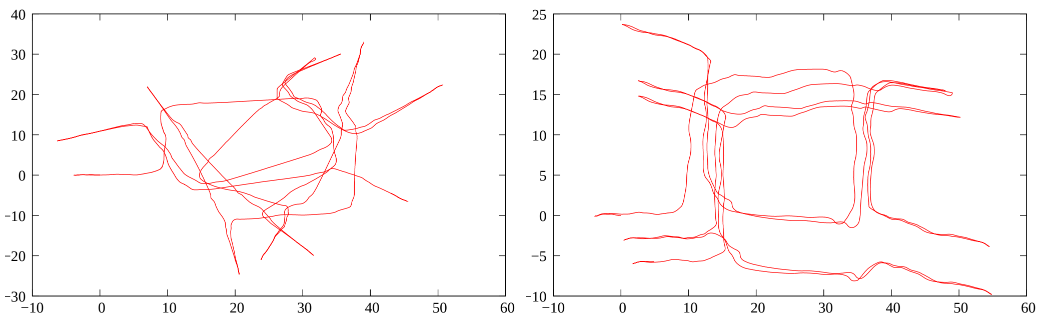
Figure 3.10: Compass readings variations

Despite these errors, the compass can be a valuable sensor for navigation. As discussed in Section 3.2.5, odometry error is mainly caused by the rotation of the robot. With this sensor the angular position of the robot can be obtained more reliably and hence, the heading value can be used to correct the robot position, thus, compensating for the accumulation of the error (Nehmzow and McGonigle, 1993) (see Figure 3.11).

3.2.7 PTZ camera

PTZ (Pan, Tilt, Zoom) devices are monocular sensors that capture video or images. Contrary to fixed cameras, they are provided with two degrees of freedom that allows the gaze to be adjusted to point to the object of interest. The “pan” angle allows the camera to rotate on its horizontal axle, and “tilt” allows the adjustment of the vertical angle of the camera. PTZ systems also offer the possibility to change the zoom value by software.

Both robots have a Canon VC-C50i camera, a sensor that has 26x optical zoom and 12x digital zoom, the pan angle range is ±100° and the tilt angle



(a) Raw odometry path (b) Compass based odometry path

Figure 3.11: Robot path

ranges from -30° to $+90^\circ$. It also has autofocus. Galtxagorri's camera is located on the base plate and given its short height, it offers a very limited vision. Tartalo's camera, on the other hand, is located upside down on the neck above the chest of the robot, a location that offers the option to grab higher objects and avoids distorting views.

3.3 Control software

The *Pioneer* platforms are provided with specific software from *MobileRobots* such as *Aria* and *MobileEyes*. Instead, Galtxagorri and Tartalo devices are accessed using the *Player/Stage* project software, and the control architecture is developed using SORGIN, a software framework specifically designed to develop behaviour-based control architectures (Astigarraga et al., 2003).

3.3.1 The *Player/Stage* project

The *Player/Stage* project provides a multithreaded robot control and simulation software for multiple robots (Gerkey et al., 2003). *Player* provides a network interface to a variety of robot and sensor hardware. The *Player*'s client/server model allows robot control programs to run on any computer with a network connection to the robot. The server must run on the same machine on which devices are connected, while the client establishes a connection through a TCP/IP socket with the server, and sends messages to access the needed devices. Once the connection is established, the server will feed the information or data continuously to the client or answer to its requests. *Player* supports multiple concurrent client connections to devices and a wide variety of mobile robots and accessories.

Player differentiates between *drivers* and *interfaces.* Drivers are pieces of code that physically access the hardware whereas interfaces specify the syntax and semantic with which *Player* and the drivers interact. This differentiation makes it possible to access several devices using the same abstraction level.

Its modular structure allows the robotics community to develop drivers for new devices. Moreover, it does not impose restrictions to the control architecture being developed and hence, the control architecture can be a multithreaded parallel program, a reactive control or just a sequential program.

Player client programs can be written in different programming languages such as C, C++, Java or Python. *Player* works in POSIX (Unix, Linux, MACOS) platforms.

On the other hand, *Stage* is a simulation environment for a single or multiple agents. *Stage* simulates a population of mobile robots moving in and sensing. It is possible to use two or three-dimensional bit mapped indoor environments. Various sensor models are provided, including sonar, laser, PTZ camera with colour blob detection, odometry, etc. *Stage* is a very useful tool to test new software and ensure that it operates smoothly on the robot.

3.3.2 SORGIN

To implement a control architecture it is essential to develop the tools needed to easily build and debug the different behaviours. The SORGIN software framework (Astigarraga et al., 2003) is a set of data structures and the library of functions associated to them which are specifically developed for behaviour-based control architecture engineering. From a generic point of view, global behaviour can be considered as a network of coupled concurrent active entities -threads- which interact asynchronously among themselves. SORGIN identifies these active entities with software components and defines their interaction, creating thus a principled method of modularity and behaviour libraries. SORGIN has been developed using the C programming language and Posix threads, both standard and portable elements.

SORGIN behaviours consist of:

- A set if inputs and outputs that are links to a specific class of data –see bellow for a description.

- A link to an initialisation function (*start*) that will perform everything needed before the main loop start. This function is specially important when the module deals directly with a physical device.

- A link to the function to be performed in the main loop, which will be executed as an independent process (*calculate*).
- A link to a function to be performed when the behaviour or module stops its execution (*stop*).

Compared with Van Breemen's framework (Van Breemen, 2001), there is not a central component that directs the overall controller. When behaviours are launched, they are fully responsible for their own execution; all of them run at their respective working frequency. After a behaviour brick is initialised and starts running, a thread is created that will execute until it stops or someone stops it.

Communication among the modules is performed by establishing connections between the modules' inputs and output links and the data. These data (**io_data**) are more than data-messages. Each **io_data** element contains all the information needed to safely read/write the data through the net when it is required, or to avoid race conditions when the information is required locally. In this way, during the implementation of an instance of a *calculate* function the designer of the control architecture does not need to worry about those details, only the offered standard functions to read/write the inputs/outputs must be called. Figure 3.12 shows graphically the backbone of SORGIN.

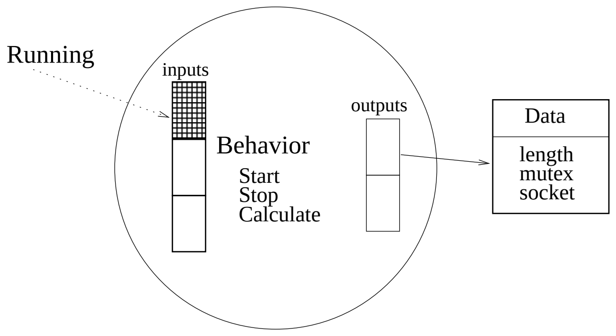


Figure 3.12: Backbone of SORGIN

The SORGIN software framework is inherently modular from a software design perspective. This enables a robotic system designer to expand the robot's competence by adding new skills without having to redesign or discard the old ones. This is very useful to construct increasingly complex robotic systems.

Behavioural abstraction is provided in this software framework with the **behaviour** and **io_data** components, allowing thus a uniform representation of computational objects. Code reusability is another positive aspect of SORGIN. It promotes the automatic reusability of behaviours across different tasks, and thus, the automatic generation of behaviour libraries.

Figure 3.13 shows how *Player* and SORGIN are combined. The robot devices are accessed through *Player* drivers that communicate with SORGIN behaviours that run as *Player* client programs.

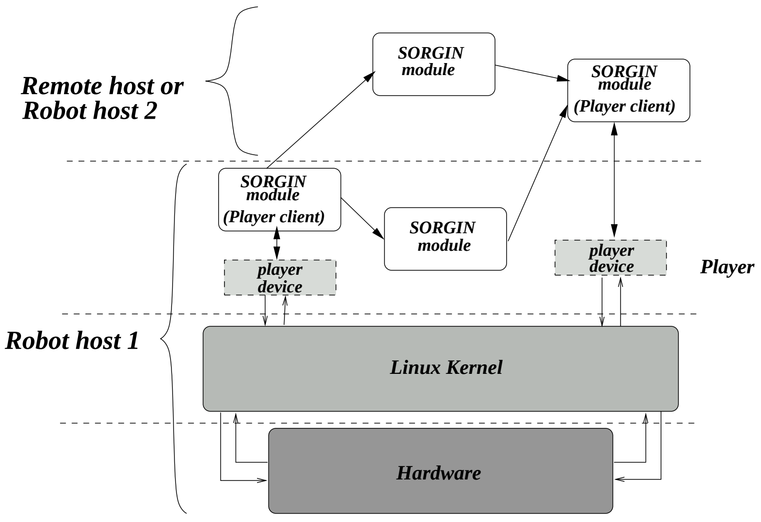


Figure 3.13: *Player* and SORGIN working together

Figure 3.14 shows an example of how behaviours can be implemented using the SORGIN framework.

```
void main()
{
  /*Data declaration*/
  io_data_t ir_readings;
  io_data_t avoid_output;

  /*Behaviour declaration*/
  behavior_t avoid_obstacles;

 /*Data initialisation*/
 init_data(ir_readings, 20);

 /*Behaviour initialisation*/
 avoid_obstacles =
 behavior_define(2, 1, avoid_obstacles_start,
                        avoid_obstacles_stop,
                        avoid_obstacles_calculate);

  /*Input/Output Connections*/
  behavior_set_input(avoid_obstacles, 0, running);
  behavior_set_input(avoid_obstacles, 1, ir_readings);
  behavior_set_output(avoid_obstacles, 0, avoid_output);

  /*initialise the behaviour*/
  behavior_start(avoid_obstacles);
  behavior_run(avoid_obstacles);
  while (1) sleep(1.0);
  /*finalise the behavior*/
  behavior_stop(avoid_obstacles);
}
```

Figure 3.14: Example of a SORGIN client program

Chapter 4

Behaviour-based systems

Contents

This chapter describes the different paradigms to develop robot control architectures and focuses on behaviour-based systems, which are the main scope of this research work.

4.1 Introduction

As mentioned in Chapter 1, when talking about robots manipulators and mobile robots can be distinguished.

Manipulators are machines that are fixed to a base, they minimally interact with the environment and they have reduced mobility. These machines are automaton and usually do not need to perceive information from the environment to plan the next movement. Therefore, they can not considered to be intelligent. On the other hand, mobile robots, as their name implies, are machines that can interact with the environment, perceiving the state of its properties with their sensor and changing its state. The way sensors and actuators are linked depends on the *control architecture* responsible for the behaviour of the robot under the environmental conditions.

As mentioned, in order to achieve good and safe movements or to carry out the correct actions, it is necessary to control the internal connections of the system between input sensors and robot actuators. The modules the control architecture is composed of, and the way these modules communicate, determine the type of control. Different lines or currents of control design have been developed which can be classified into four groups (Arkin, 1998):

- Deliberative or symbolic systems: associated to the intelligence concept of classical AI, the main goal is to develop the *brain* of the robot, leaving out the processing related to sensors and actuators.

- Reactive systems: reacting to the environment becomes the main goal instead of thinking.

- Hybrid systems: the combination of the first two systems attempts to overcome the drawbacks of deliberative systems adding the capability of interacting with the environment.

- Behaviour-based (BB) systems: they are based on reactive control, but robots with more complex capabilities are developed.

It is hard to decide which paradigm is better than the other because the adequate type of control depends on the type of robot, the task or goal to achieve and the niche of the robot. In the following sections these four types of control are better explained and analysed.

4.2 Deliberative or symbolic control

The deliberative control architecture scheme emerged when Artificial Intelligence (AI) dominated the world of robotics. The word *deliberation* refers to thinking hard and, as discussed in Chapter 1, the efforts of AI focused on developing aspects of human intelligence in machines.

This control architecture consists of three steps that need to be performed in sequence: *Sensing*, *Planning* and *Acting*. The sensing module, responsible for perception, has to obtain a highly reliable representation of the current state of the system based on readings from sensors and the internal model of the world. After extracting from the readings the current state of the environment, the task of the planning module is to find the sequence of actions that, given the current state and the goal, will bring the robot to the destination or the goal. Hence, the acting module is responsible for transforming the sequence of actions to signals that the actuators can understand to move towards the goal (see Figure 4.1). This cycle is repeated once and again.

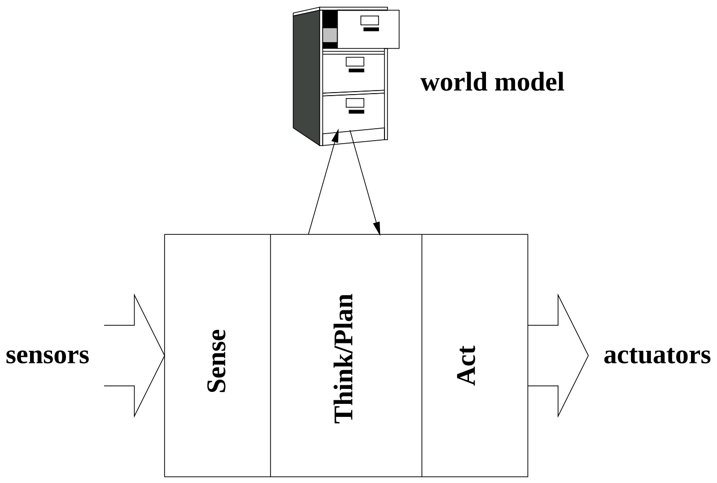


Figure 4.1: The deliberative control architecture scheme

AI concentrated on developing abstract representation methods to create models and reason about the environment. The perception module is the process of perceiving the environment through sensors and representing and modelling the current state through a set of symbols and predicates, so that a correspondence can be established between the current state and the internal representation. Therefore, all the reasoning is performed within the model and there is no direct link between the process of perception and the action module. The world model contains all the robot needs to know to carry out its task successfully.

Typically, planning tends to be more than finding a sequence of actions leading to the destination. Moreover, there can be more than a possible path to the goal and properties such as length and safety of the possible paths should be analysed in order to choose the most appropriate one. The planning process must take into account all the parameters in order to select the best sequence. The computational payload of the reasoning process is so high that the acting response is delayed unreasonably. As we saw in Chapter 1, both robots *Shakey* and *Cart* used this architecture and needed too much time to obtain a plan to reach the goal and execute a small movement.

The most important problems associated to this type of control architecture were:

- Delicate systems: one of the main problem of this control system is that the modules have to be executed in sequence. The output of one module is the input of the next one. The final response of the system is the result of the three modules, none of the modules produces a behaviour independently. If any of the modules fails, then the whole system will fail.

- Environment representation: given the sensory uncertainty and dynamism of the real world the representation of the environment is a very complex task. The model needs to contain only relevant information and all the relevant elements of the environment for the planner to reason properly and select the adequate plan. The design of the appropriate model has shown not to be trivial at all as described by the well known *Frame Problem* (see (Dennet, 1998)).

- Unable to cope with dynamic environments: the planning module is based on the model of the environment created by the system, and while planning the sequence of movements that the robot has to execute, the world may change and the action sequence proposed by the planner may not be appropriate any longer for the new state. Moreover, if the planning process is slow, the robot has to stop while making planning estimates in order to avoid moving blindly. The high latency of the planning process results in a slow feedback loop between sensing and acting. It must be mentioned that the computational power available nowadays relieves this problem.

In spite of these problems, AI researchers were confident with the Cartesian decomposition of intelligence and speculated that robot controllers should work at the same way. This control architecture is widely used nowadays in game-playing, specifically in board games like chess machines. In general, when the environment is static, when there are not dynamic objects and the robot does not need to update the internal model once and again, the planning module has enough time to calculate the desired sequence to reach the objective.

4.3 Reactive control

This control type is based on the tight connection between robot's sensors and actuators, as in the initial vehicles proposed by Braitenberg (1984). It has not a planning module, so it neither has to create nor to maintain the environment model, although it can maintain a small state information. It is simply based on the direct readings of the sensors to act as quickly as possible, hence the name *reactive* (see Figure 4.2). As a consequence, the main loop of this control paradigm is much faster than that of the deliberative one. The control usually consists of a collection of rules that do not require much thought; they act as simple reflexes such as jerking back one's hand after having being pricked with a needle.

The deliberative control was based on much thought. The action of the robot was the result of the reasoning made in the environment model once the current state was deduced from the sensory inputs received. On the contrary, reactive control systems are based on *reflexes*, and so the reasoning process of the reactive control is much faster. It places more emphasis on speeding up the feedback loop than on thought, avoiding the computational cost of reasoning and just executing a set of precompiled rules activated upon the sensory input received.

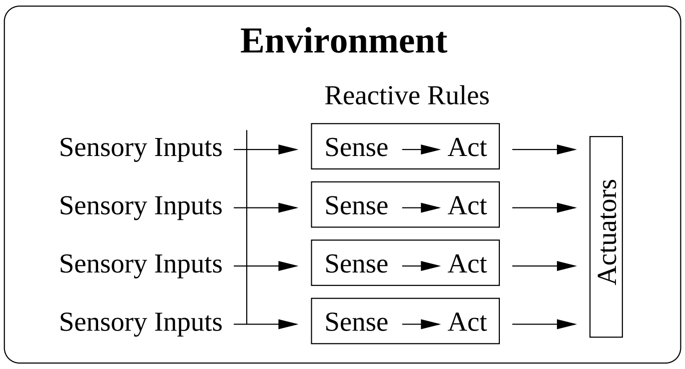


Figure 4.2: The reactive control architecture scheme

Reactive control then can be summarised as collection of conditions and actions, where conditions may be subject to the sensory readings, or to internal states. For example, the robot can turn to dodge an obstacle when it is detected by sensory readings or because the internal state of the system has indicated that it has to change direction. Of course, the complexity of the rules is beyond this example; it is possible to create very complex rules by combining the results of the sensor readings and internal state information.

The most important drawback of this type of control is that it is too rigid in the sense that it does not have any memory, it does not maintain any internal representation of the environment and as a result it can neither adapt nor learn. This fact represents a serious drawback on the level of complexity of the behaviours these kind of systems can show.

4.4 Hybrid control

Hybrid systems can be considered as the evolution of deliberative systems. As it has been commented, deliberative systems are complete respect to intelligence, but they are very slow. However, reactive systems are faster, but they do not hold any intelligence. The main idea of hybrid systems is to combine the best of these two systems, taking the intelligence of deliberative systems and enabling them to react upon dynamic environmental changes. Hybrid systems are also called *three-layer systems* (Kortenkamp and Bonasso, 1998; Arkin, 1998; Connell, 1992; Gat, 1991; Langley et al., 1991) because they typically consist of three components; a *reactive layer*, a *planner* and a *middle layer* that links the above two together (see Figure 4.3).

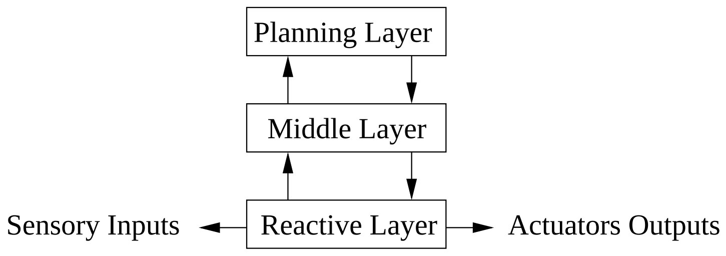


Figure 4.3: The hybrid control architecture scheme

The classical deliberative layer, with a model of the environment, is responsible for planning the actions the robot has to perform to reach its goal and the reactive layer enables the robot to survive while it is thinking out the path. The information flows in both directions, from the planning layer to the reactive one and vice versa. The planner sends action commands to the reactive layer and the reactive layer communicates the planner trapped situations and actions that can not be executed unexpectedly. The middle layer is used to resolve conflicts between the other two and it is responsible for compensating the limitations of both systems, reconciling their different time-scales, and dealing with their different representations and reaching consensus on any contradictory commands that they have sent to the robot.

The design of this type of control architectures has some complications (Matarić, 2009):

- These systems still show difficulties to cope with dynamic environments. Undoubtedly this architecture preforms well in static environments, but when introducing a deliberative layer, the problem of representing dynamic environments remains unsolved. The dynamic changes can not be foreseen and the requirement of keeping the correspondence between the world and the model requires a constant effort to keep the model updated and to plan the failed paths again.

- The design and implementation of the middle layer is very difficult and it cannot easily be reused for several robots/environments. This layer is severely tied to the specifications and the task of the robot, so it has to be developed for each new robot and task.

In spite of these problems, it must be said that this is the most popular type of control architecture used nowadays.

4.5 Behaviour-based robotics

Behaviour-based systems are distributed parallel control architectures in which each basic processing unit produces a behaviour on the robot. This control architecture is closer to reactive control and it is often confused with it. Behaviour-based systems are biologically inspired systems developed bottom up, like reactive systems. The difference lies in the level of representation allowed and in the processing units. Instead of precompiled rules, the basic processing units of behaviour-based systems, named *behaviours*, have the following characteristics (see Figure 4.4):

- Each one achieves its purpose: each behaviour only has one goal.

- They all run continuously: behaviours receive inputs and produce outputs in a continuous loop.

- Behaviour modules communicate among them: a behaviour is able to communicate with other behaviours, receiving inputs from other modules as well as from sensors.

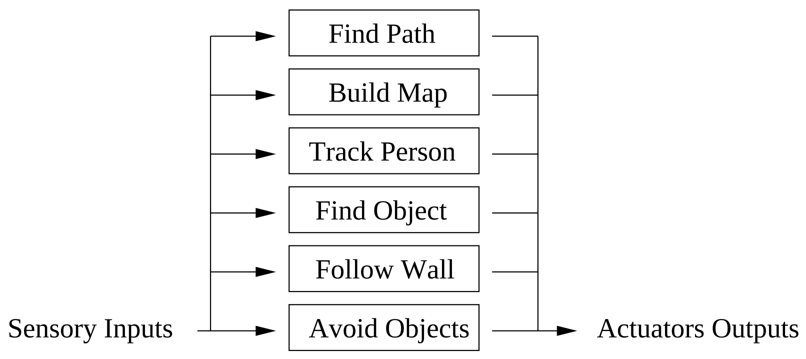


Figure 4.4: Generic behaviour-based control architecture scheme

- They are more complex than rules: the precompiled rules of reactive systems produce simple actions like *turn-left*, *go-straight* and joining the outputs of rules the robot achieves a simple behaviour.

What is a behaviour then? It is a very loose concept that could refer to the motion of a mobile robot, the trajectory of a manipulator, an ant following another, etc. A behaviour is composed of three fundamental components; *the task*, *the environment* and *the body* (see Figure 4.5). These three elements are tightly coupled and can not be considered independently (Nehmzow, 1999).

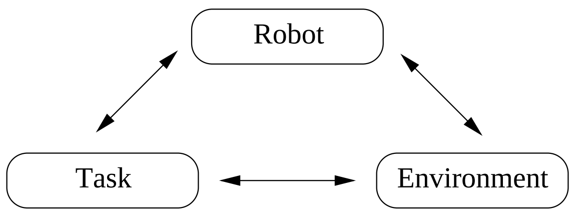
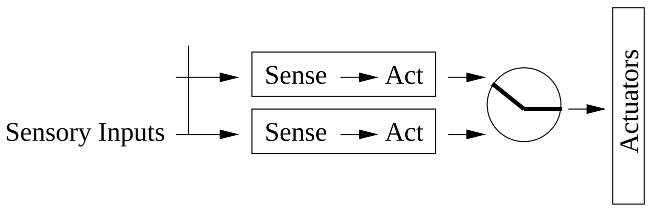


Figure 4.5: The triangle of fundamental components to be intelligent

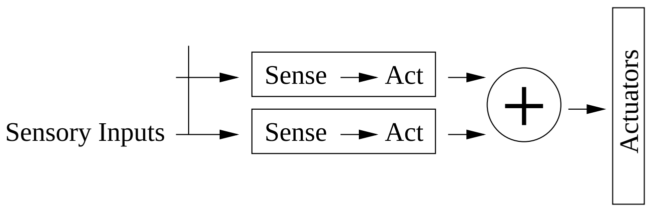
The behaviour *emerges* from the interaction between these three components, and if one of these components change, so does the behaviour. For example, a behaviour such as opening a door, has no meaning if the robot does not have arms, or if it is located undersea. Thus, a behaviour is an execution thread that has only one goal usually specific to the environment.

Like reactive control rules, the behaviours of this system are executed in parallel, getting a fast response. One of the biggest differences between reactive systems and behaviour-based systems is that it is impossible to acquire enough knowledge of the environment using exclusively rules. However it is possible to construct a representation of the environment using the appropriate network of behaviours.

Joining behaviours may eventually lead to create an intelligent machine, but control is needed to coordinate all the behaviours. When building a BB system it is very important to analyse well the design. The system is composed by behaviour modules that run in parallel and somehow, the sensor fusion problem of deliberative architectures is transferred to *action fusion* in BB architectures. Different modules output different actions but only one action at a time can be executed by the actuators and therefore, action proposals must be coordinated. Generally speaking, there are two approaches for *action selection*: *command arbitration* and *command fusion* (Figure 4.6).



(a) Arbitration



(b) Fusion

Figure 4.6: Approaches for action or behaviour selection

Command arbitration scheme is the process of selecting one action among all possible candidates, also called *competitive strategy.* The modules compete each other so that its action has an effect on the actuators, being the winner the one which is going to control the behaviour of the system. There are different ways of selecting the winner. The modules can compete on a priority scheme. Maes (1989) proposed a system in which each module defined the level of its activation, and the winner was the module with the higher activation. Another alternative is to use a voting system on the set of all the possible actions to take. The action that receives most votes will be run. This voting system was used by the DAMN architecture (Rosenblatt, 1995).

On the other hand, the *command fusion* or cooperative approach is the process of combining all the candidate outputs in a single output action. Thus, with cooperative strategies, different modules contribute to the overall response, usually as a weighted sum (Arkin, 1989). The final output is not the action specified by a single module, but the combination of the actions output by the active modules.

4.5.1 The subsumption architecture

The *subsumption architecture* proposed by Brooks (1986) is the most well known BB architecture with which many successful robots were developed. *Herbert* (Connell, 1990), *Polly* (Horswill, 1994) and *Toto* (Matarić, 1992) are three instances of robots developed at the MIT using the subsumption architecture, and with very different capabilities.

The subsumption architecture is composed by layers which are arranged in increasing order of complexity. Each layer implements one level of behaviour (Figure 4.4). A layer is composed by several behavioural modules working in parallel and achieves a task in the sense that, behavioural modules form a priority based control architecture, being the bottom layer the one with lower priority. Higher level layer modules can subsume/inhibit outputs/inputs of modules in lower levels. This communication messages include a time stamp to indicate the latency of the signals.

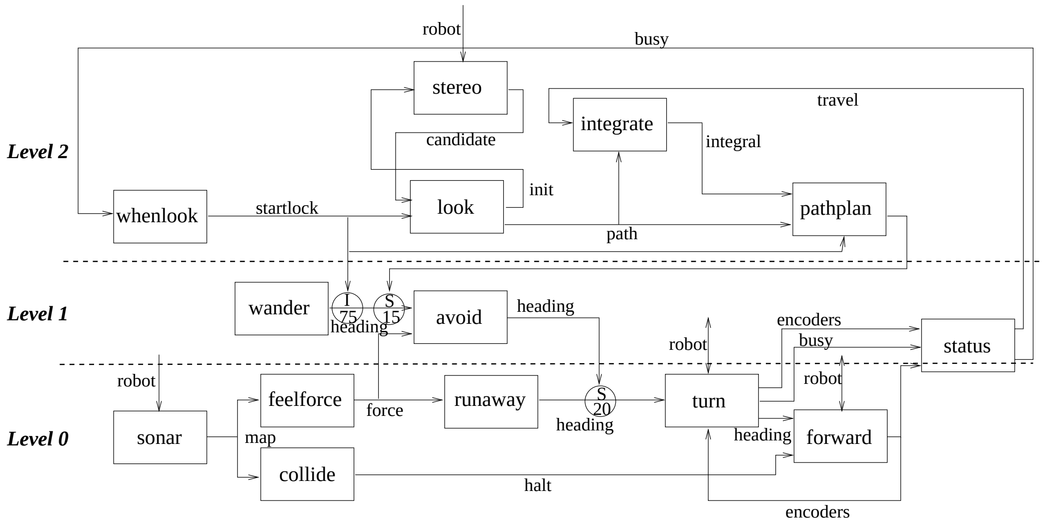


Figure 4.7: Communication among the modules in the subsumption architecture

In this way, layers of behaviours are developed incrementally and hence, the system shows robustness if a layer fails (see Figure 4.7).

Therefore, the subsumption architecture implements a priority based command arbitration action selection scheme and so the hard of its design. It is not obvious how the priority of layers should be established. Depending on the environmental conditions, a selected priority order could bring the robot to reach its goal or to fail to achieve it.

Chapter 5

Galtxagorri: a behaviour-based navigation system

Contents

This chapter introduces the concept of behaviour-based (BB) navigation and describes two instances of robots that use biologically inspired navigation techniques: *Toto* (Matarić, 1990), the best exponent of BB navigation systems; and Galtxagorri, (Lazkano, 2004), another BB system inspired by *Toto*. Galtxagorri's control architecture was the starting point of the work described in the present dissertation and thus, needs to be explained in detail in order to put the reader into the picture.

5.1 Introduction

Getting from one place to another is a considerable challenge for a robot. *Navigation* refers to the way a robot finds its way in the environment (Matarić, 2009). To face this is essential for its survival. Without such basic ability the robot would not be able to avoid dangerous obstacles, reach energy sources or return home after exploring its environment. Navigation is therefore a basic competence that all mobile robot must be equipped with.

The navigation task has been approached in different ways by the main paradigms of control architectures previously described in Chapter 4.

From the point of view of the classic or symbolic trend, navigation is the result of answering three questions (Levitt and Lawton, 1990):

- *Where am I?*
- *Where are the other places relative to me?*
- *How can I reach them?*

These questions give rise to the three main areas of robot navigation: environment *mapping*, *localisation* and *path planning*. Navigation then can be defined as the combination of those three fundamental competences (Nehmzow, 1999); map-building, self-localisation and path planning.

The fact that simple animals show excellent navigation capabilities prove that a system, equipped with appropriate sensors and actuators, together with an appropriate control architecture, can exhibit an adaptive behaviour that allows it to survive in unpredictable environments (Trullier and Meyer, 1997).

Bioinspired behaviour-based navigation is then defined as the process of determining and maintaining a path or trajectory to a goal destination (Bekey, 2005; Mallot and Franz, 2000). The main question to be answered for navigation is not *Where am I?* but *How do I reach the goal?* and the answer does not always require knowing the initial position. Therefore, the main abilities the agent needs in order to navigate are to move around and to identify goals whereas neither a centralised world model nor the position of the robot relative to it need to be maintained.

Bioinspired robotics should not be confused with *Biorobotics*, an emerging field defined as the intersection between biology and robotics (Webb, 2001; Meyer, 1997; Bennett, 1997).

5.2 A note about landmarks

Landmarks play an important role in classical navigation and also in biological and bioinspired navigation systems. A characteristic landmark is any physical property that the robot can perceive with its sensors (Nehmzow and Owen, 2000). Landmarks then do not need to be physical objects but may be environment corners, pipes or line segments that can be associated with features such as corridors or open doors. They can also be parts of images or scenes (Trahanias et al., 1999).

Regardless of the nature of the landmarks, they can be classified according to different criteria:

- Local landmarks: these are characteristic of a particular location, but do not need a global reference system to be recognised, for instance doors or fire extinguishers.

- Global landmarks: these are related to a global reference system such as the Earth's magnetic South or the (x, y) coordinates of the robot with respect to its initial position.

If the navigation system is based purely on landmark recognition, then the landmarks should have certain characteristics (Nehmzow, 1999):

1. They have to be visible from various positions.

2. They have to be recognisable under different lightning conditions, viewing angles, etc.

3. They must be either stationary throughout the period of navigation, or their motion must be known to the navigation mechanism.

The selection of the appropriate set of landmarks is a difficult task which greatly depends on the sensors that the robot is equipped with and on the environment where the robot will be placed. Since not all the environments have the same properties one of the best options may be for the robot to learn the appropriate marks (Greiner and Isukapalli, 1994; Thrun, 1998a).

Typically, indoor environments are large spaces full of landmarks that the robot can use to locate itself in it. But these landmarks usually do not uniquely characterise a single location because many locations look alike. This problem is known as *perceptual aliasing*, which can be dealt with using sensory information from different sensor modalities. For example, it is possible to combine the information of landmarks with the odometry to disambiguate between locations with the same sensory pattern (Thrun, 1998b; Burgard et al., 1998).

5.3 Behaviour-based navigation

As mentioned earlier, behaviour-based systems are inspired by biology. Different authors (Trullier et al., 1997; Mallot and Franz, 2000) classify biomimetic navigation behaviours into two main groups:

1. *Local navigation* strategies are local control mechanisms that allow the agent to choose actions based only on its current sensory input, i.e. relying only on the characteristics that are within the agent's perceptual range. Four are the strategies that fall in that group:

    - Search or wander: move in the environment safely and avoiding all types of obstacles, without any purpose or objective.
    - Path integration: integrate or store the movement direction and distance to maintain the association between the source and the target.
    - Taxis: move towards a target that produces a certain stimulus. Taxis is certainly involved in most navigation behaviours (together with obstacle avoidance and course stabilisation), but taxis does not necessarily include the essential capabilities of locomotion and goal recognition.
    - Goal orientation: this is a navigation mechanism by which the agent finds the goal. To do so, it sets its self-centred relationship with respect to the elements surrounding the target, making unnecessary that the goal is marked with any characteristic.

2. *Way-finding* methods: they are responsible for driving the agent to goals outside the agent's perceptual range that require recognition of different places and relations among them. They rely on local strategies. Three are the main way-finding strategies and they are mentioned bellow in increasing order of complexity:

- Perception-triggered response (PTR): this strategy connects two locations via a local navigation mechanism. The recognition of the "position" launches a navigation mechanism that leads the robot to the target. In this context, a location is defined as a particular perception, a *landmark*. This strategy can be used to build paths. The paths or the sequences are independent of each other and direct the robot towards a unique goal (Trullier et al., 1997). But this strategy is not enough to choose a path, i.e. it can not be used for path planning since knowledge is limited to the next action to perform. Linking perceptions to a unique action is called PTR and limits the navigation system to always use the same sequence of locations. These locations are combined defining routes, but planning and terrain inspection capabilities require a wider knowledge about the space configuration of the known places.
- Topological navigation: this strategy requires abilities like path integration and planning. To plan a path between the robot's location and the goal, it is essential to know the spatial relationship among the connected locations in the environment, so this strategy is not valid to generate paths in unknown terrain.
- Terrain inspection: during this process, the spatial representation must be manipulated and accessible as a whole, so that the spatial relationship between any two of the represented places can be inferred. An agent using survey navigation is able to find novel paths over unknown terrain, since the embedding of the current location into the common frame of reference allows the agent to infer its spatial relation to the known places.

The behaviour-based approach to robot navigation relies on the idea that the control problem is better assessed by bottom-up design and incremental addition of light-weight processes, called behaviours, where each one is responsible for reading its own inputs and sensors, and for deciding the adequate motor actions. There is no centralised world model and data from multiple sensors do not need to be merged to match the current system state in the stored model. The motor responses of the several behavioural modules must be somehow coordinated in order to obtain valid intelligent behaviour. As mentioned before, way-finding methods rely on local navigation strategies. How these local strategies are coordinated is a matter of study known as *motor fusion* in BB robotics as mentioned earlier (Section 4.5).

The knowledge about the world differs from the world map used within the classic or symbolic approach in the sense that it is neither a central model nor a purely symbolic one. As proposed by Matarić (1992) and further explained in the following sections, it can be fully distributed and of a procedural nature.

5.4 Galtxagorri's predecessor: the robot *Toto*

Toto was a robot developed by Maja Matarić (1990) with the aim of implementing a complete navigation system. *Toto* was circular and omnidirectional. It was equipped with a ring of twelve Polaroid sonars and a low resolution compass of two bits. The robot's base diameter was 30.48 cm, and hided a three-wheeled driving system. The body, which had a diameter of 20.32 cm, stood on the base (see Figure 5.1).

Matarić wanted to demonstrate that behaviour-based systems, far from restricting the robot to purely reactive actions, provide a methodology for developing complex tasks like goal oriented navigation. The robot had to calculate the shortest path to a given target, and to do so it explored the environment, learning and storing it in an appropriate representation. *Toto* (Matarić, 1992) is, even today, the main exponent of how environment information should be expressed and managed in navigating BB systems. Matarić proposed a topological representation that instead of being a declarative model, was fully procedural in the sense that each node in the graph that formed the map was a completely functional unit responsible for guiding the robot when the current location matched the node.

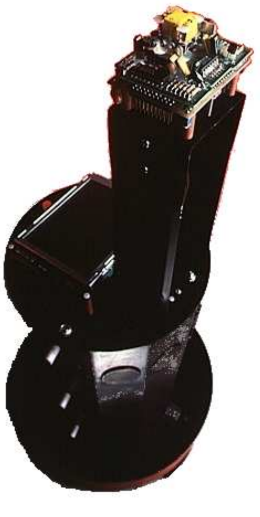


Figure 5.1: The robot *Toto*

Toto's control architecture consisted of various competences organised in three layers.

5.4.1 *Toto's* low level controller

Toto's bottom layer consisted of four mutually exclusive behaviours: stroll, avoid, align and correct. Therefore this layer did not require any coordination mechanism. The function of the low level of the controller was to move around safely. This level of the controller was in fact a reactive system. Getting the information from its sonar ring, *Toto* was endowed with a biomimetic behaviour found in rats that kept the robot near walls and objects while navigating. This behaviour allowed *Toto* to follow boundaries.

5.4.2 *Toto's* landmarks identification behaviours

As it has been said, landmark identification is necessary for goal oriented navigation. Any task beyond mere wandering requires knowledge of characteristic places of the environment, so *Toto* also had behaviours for that purpose. The functionality of the second layer was to identify landmarks so that *Toto* was able to recognise walls and to notice if it was moving straight in the same direction for a while, or if it was meandering. A straight movement indicated that it was next to a wall and a meandering one indicated a cluttered area. Once the wall was recognised, *Toto* was able to identify if the wall was on its left or on its right. *Toto* was also able to identify corridors. For that, if it recognised walls at both sides, then the landmark was labelled as a corridor. On the other hand, if no wall was detected, it recognised the place as a messy area. Landmarks contained also information about the compass heading and about their approximate length. Equivalent compass heading values were defined so that *Toto* recognised the locations regardless of its heading value and was able to traverse them in both directions.

5.4.3 *Toto's* mapping

The last layer was formed by the behaviours for the construction of the map and for goal oriented navigation. *Toto* constructed a topological map were the identified landmarks were stored in the nodes. *Toto* used a graph of behaviours which was a procedural representation in which each node was a process responsible for acting in a certain way when the assigned node landmark was confirmed. The result was a distributed map representation, where each of the landmarks that the robot discovered was stored in its own behaviour process.

When the time came to build the representation of the environment, i.e. the map, *Toto* began the exploration process with an empty graph of nodes. The nodes were empty in the sense that no mark was allocated or associated to any of them. Once a landmark was observed, a node was filled with the characteristics of the landmark. The first landmark identified was assigned to the first node, and the process went on with the addition of nodes that met the characteristics of the identified landmarks to the graph, and the connections between adjacent nodes.

Every time that the landmark detectors confirmed the identification of a new landmark, all nodes verified if the characteristics of the landmark corresponded to the one they had stored. If any of them matched the same characteristics, the node became *active*. Only one node at a time was active and thus, the active node represented the localisation of the robot. If no node confirmed the characteristics of the landmark, it was associated with a new node adjacent to the currently active one, and a *wake up* signal was sent to the new one. When the landmark of the new created node was confirmed, it became the active node and an *inhibition* signal was sent to its predecessor.

Figure 5.2 shows an example of an environment and the topological map obtained using the mapping process described.

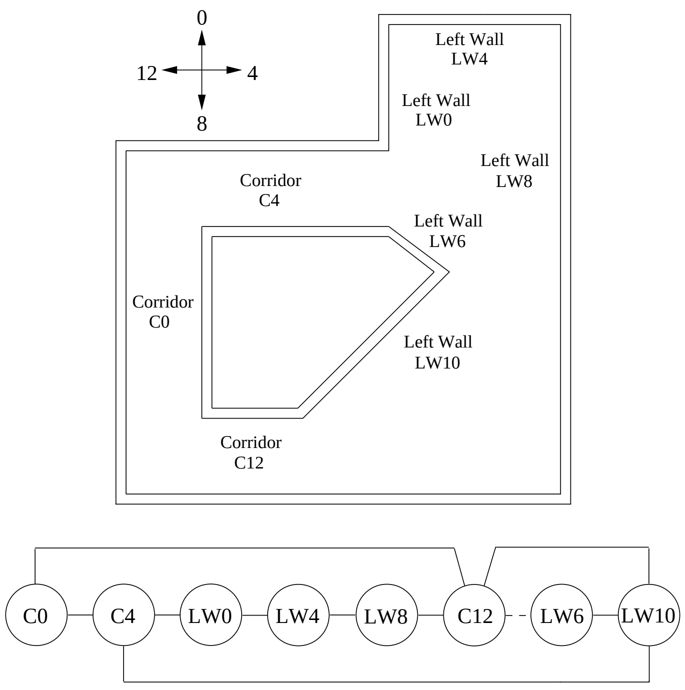


Figure 5.2: Environment and the corresponding topological representation

5.4.4 *Toto's* path planning

Toto's possible goals were the locations represented by the nodes on the acquired topological map. These goals were input to the system pressing some in-built buttons. Once a target was received, the path planning process was performed by a *spread of the activation* mechanism. The node identified as "objective" sent *calls* continuously to its neighbouring nodes until the active node matched the target node. When an intermediate node received a *message*, the message was propagated in the direction of the goal node. In this manner *Toto* knew the direction it had to take from one node to the next. To ensure the shortest path, the length of the node was added to the distance information in the message while the spread was propagated. The spreading of the activation mechanism avoided the need for replanning because the goal node continued sending calls until the robot reached its objective.

Matarić demonstrated the correct functionality of this type of system in small environments, showing also that behaviour-based systems allow the development of complex tasks such as exploration and mapping, and that they even endow the robot with the capacity for planning the shortest path.

The procedural topological representation together with the process of localisation and path planning had significant advantages over traditional methods of navigation:

- Procedural representation by contrast with the declarative model. The "map" is not a symbolic model, but a series of connections between processes that are executed at run time.
- Lower computational cost, since nodes are very simple processes.

5.5 Galtxagorri

Lazkano et al. (2006a) developed a BB control architecture for navigation that attempted to extend Matarić's proposal and overcome some of *Toto*'s shortcomings:

- In (Lazkano et al., 2006a) the topology of the graph was extended to an a priory unlimited number of predecessor/successor links. Moreover, Galtxagorri's environment's description incorporates information about the spatial relationships among the nodes that helps to identify the direction to follow in order to reach the goal. The links to neighbouring nodes include information about the direction θ in which the j-th succession occurs: $succ(\theta)_j$, $pred(\theta)_j$.

The number of successors of a location (a node) is only limited by the cardinality of the set of identifiable compass orientations. In that way, each successor of a node is a successor in a fixed orientation distribution.

- Function specificity: each node executes its own specific function when it becomes active. This function specialization allows the breaking up of the planning task, making it possible to assign the corresponding local planning strategy to each node.

The developed architecture was integrated in the robot Galtxagorri (see Chapter 3, Section 3.1). The proposed environment representation is completely functional and can be fully distributed. Moreover, the architecture was tested in a much larger environment where there were cyclic paths.

5.5.1 *Galtxagorri's* local navigation strategies

Galtxagorri's control architecture followed the taxonomy of biological navigating systems described in Section 5.3 from local navigation strategies to the topological navigation level.

The BB control architecture consisted on the following local navigation strategies:

1. Balance free space around the robot.
2. Maintain a privileged compass orientation.

Each of these two modules output translational and rotational velocity components. The compass follower behaviour affects the rotational velocity, implementing a proportional control over the difference between the current orientation and the desired one, and the translational velocity proposed is inversely proportional to the rotational one. This mechanism enables the robot to turn on the spot by allowing this behaviour to act independently when the robot moves forward in the opposite direction from what it is required. The overall robot velocities result from a cooperative control strategy, obtained by a weighted sum of the individual proposals.

It must be pointed out that no laser sensor was mounted on Galtxagorri at the time and hence, sonar sensors where the only available range sensors either for obstacle avoidance or for landmark identification.

5.5.2 *Galtxagorri's* landmark identification behaviours

Four are the landmark identification subsystems:

1. Corridor identification: this module outputs a confidence level of being in a corridor as a result of calculating a weighted sum of the two left most sonars and the two right most sonars over time.

2. Mean compass orientation over time (t): the belief of the compass orientation being maintained, ($\overline{\theta(t)}$) is the mean value of the compass readings stored in a buffer.

3. Emergency exit panel (eep) recognition: it is needed to somehow identify crossroads; and it is crucial to help choose the correct action when different ways can be taken. But the lack of a laser sensor made it difficult to robustly identify the junctions by using patterns of range sensors.

   Taking advantage of the regulation for the location of emergency exit panels in public buildings, crossroads are anticipated by recognising these panels using vision. To train the identification module to recognise the panels from images a multi-layer perceptron (MLP) combined with a genetic algorithm was used to train the identification module, and the tilt and zoom parameters of the camera were actively controlled to make the recognition module less sensitive to the robot's viewpoint. The module output a confidence level (cl_{eep}) that affected the robot's global velocity (v) to avoid noisy identifications (see (Lazkano et al., 2004a) for more details):

   $$v' = (1 - cl_{epp}).v$$

4. Dead-ends: some corridors are dead-end ways so the only way out is to turn around and retracing one's steps. Dead-ends were identified using the four front sonars of the sonar ring and measuring the distance to the walls that surrounded the robot.

   To get the desired behaviour it was enough to set the opposite compass orientation and to let the compass following module act alone. The output of this behaviour module was then used to subsume the outputs of the local navigation strategies.

The proper coordination of all the modules was enough to perform perception-triggered responses (Lazkano et al., 2004b). But space information was still needed for the robot to be able to navigate in different routes.

5.5.3 *Galtxagorri's* environment representation

Within the behaviour-based approach, topological maps should be composed of tightly coupled behaviours specific to the representative locations. The overall "map" is then composed of sets of behaviours, each launched on a different thread, with an associated functionality and a specific node signature that makes them distinguishable.

Formally, a map is a directed and connected graph, where each node consists of:

1. A set of inputs (from landmark identification subsystems) and outputs. These outputs should serve to reduce the distance between the current state and the goal.

2. A set of predecessors and successors. These links to neighbouring nodes include information about the direction θ in which the i-th succession occurs: $succ(\theta)_i$, $pred(\theta)_i$. The purpose of integrating knowledge about significant locations is to help the robot to decide the action required to reach the goal. The number of successors of a location (a node) is only limited by the cardinality of the set of identifiable compass orientations. In that way, each successor of a node is a successor in a fixed orientation distribution.

3. A signature that uniquely identifies the node from its immediate predecessors and successors: $signature_i$. Locations are identified by the recognition of characteristics that makes that place singular, and therefore the signature reflects the state of a set of specific landmarks.

4. A function α_i to be executed when the node is active and that will output the action to be performed at the node specific current state. The behaviour the robot shows can differ depending on its location, and so can the associated function of several nodes.

5. Each node has also assigned a location identifier or pose information in terms of coordinates (x_i, y_i) that gives the spatial relationship among the nodes and helps to extract information about the direction to be followed to reach the goal. This location identifier does not give information about distance. This information is just a virtual coordinate frame exclusively used for planing purposes.

Every node will follow the finite state automata (FSA) in Figure 5.3, where four states are identified:

- State 0 (*sleep*): none of neither its predecessors nor itself is currently active and therefore, nodes in state 0 are simply inactive.

- State 1 (*waiting*): one of its predecessors is the current active node and it sends a *wakeup* signal to its successors to indicate them that they may be the next to become active. A node in state 1 is continuously checking the state of the landmark identification modules, expecting they will match their signature to become active. But at the same time they can receive a *dewakeup* signal to put them back to sleep again.

- State 2 (*active*): is the active state. The landmarks match the node signature, becoming the node that should guide the robot through the execution of the α_i function associated to the node. The node that reaches this state first is responsible for sending *wakeup* signals to its successors.

- State 3 (*transition*): although the node remains active, it has lost the signature match. Two reasons can make this happen: the robot is out of the nominal trajectory due to some kind of obstacles, but it will recover it when the obstacle is out of its perceptual range, and the node will return to state 2. Another possibility is that the robot is somewhere in between the locations represented by two consecutive nodes, and that the node will receive a *sleep* signal when the proper successor becomes active. But before changing to state 0, the node in this state is responsible for alerting its successors through *dewakeup* signals that they should not be alert any longer.

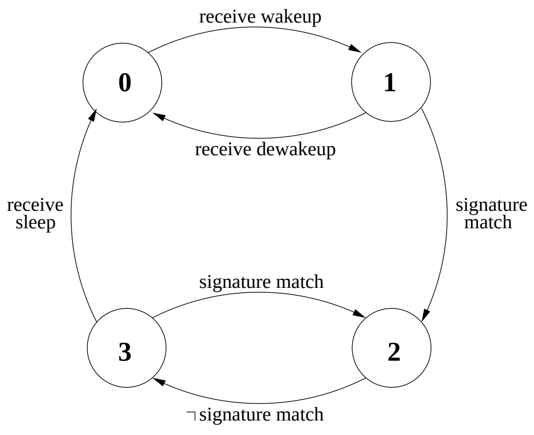


Figure 5.3: FSA associated to each node

5.5.4 *Galtxagorri's* path planning

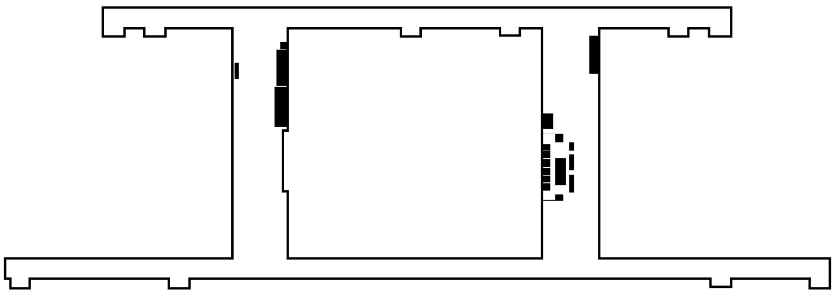
For the proposed approach, and due to the strong regular and symmetrical characteristics of the environment, planning is based in some metric encoding integrated in the nodes and in the explicit information about the orientation in which each successor occurs. Each node has a location identifier in terms of a pair of coordinates (x_i, y_i) that helps to extract information about the direction in which the goal can be reached from the current position. In the example described in the next section, the same location is represented by several nodes that share the same grid identifier but have different signature.

This metric encoding together with the information about the direction in which successions occur, allows each node to recognise its spatial relationship with the rest and to which of its successors the robot must go to in order to get closer to the goal node. This additional information about the direction in which a successor occurs, allows the recognition of spatial relationships among nodes, which is similar to the way surrounding cells in a metric map are related to each other by means of a coordinate system. Using of the simulated grid of rough granularity, the spatial relationships of nodes and goals are easily identifiable and therefore, it is rather simple for each node to decide the action to trigger in order to get closer to the goal.

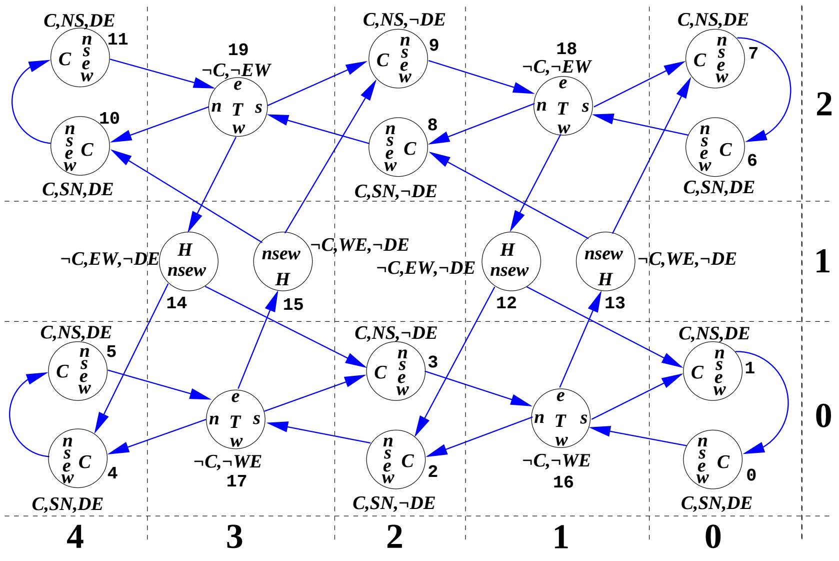
Although in the example given bellow planning is simple, more complex strategies could be used depending on the environmental requirements. Even more, each node could apply its own particular planning strategy, maintaining thus the distributed nature of the map.

5.5.5 An empirical example

Figure 5.4 shows the map defined for the environment in the top of the figure. The environment is represented by a total amount of 20 nodes (each node in the figure has a number associated that will be used to reflect the node sequence activation during robot trajectories in the experiments explained later on in this section. Notice that each successor of a node has encoded the orientation in which the succession occurs. The illustration also contains the signature of each node. The map structure is not isomorphic to environment morphology; though physically they are single locations, every corridor and hall is broken down into two nodes, each one belonging to one of the nominal orientations in which the location can be traversed. By doing so the planning algorithm is simplified because it can be used to keep track of the robot's path. Note the metric coordinates reflected in the figure; these coordinates will be used to define routes and goals during the performance evaluation phase.



(a) Second floor of the Faculty of Computer Science



(b) Environment map

Figure 5.4: Procedural map of nodes (C: corridor, H: Hall, DE: dead-end, NS, SN, EW, WE: orientations)

Node signature

The signature is simply the state of the landmark identification processes at the place represented by the node. The environment we deal with, together with the local navigation strategies, impose signatures (sig_i) composed of three characteristics: nominal compass orientation distribution, corridor belief and dead-end properties (Equation 5.1).

$$sig_i = \{\rho_{corr}(i), \rho_{compass}(i), \rho_{dead\_end}(i)\} \tag{5.1}$$

Let's analyse each signature component.

$\rho_{corr}(i) = \{0, 1\}$: this Boolean value should be thresholded with the confidence value of being in a corridor.

$\rho_{compass}(i) = (\hat{\theta}_i, \pm\sigma_i, csign)$: the magnetic orientation is the main reference of the developed control architecture. Local navigation strategies allow the robot to accommodate its position in the free environment while following a compass orientation. Each location is traversed in a nominal orientation and thereby, the nominal orientation is a local property that can be exploited.

The signature of a node will then have a compass value range allowed in the location. For the example here described we assume a normal distribution in the orientations and describe the range by the mean compass value ($\hat{\theta}_i$) together with the standard deviation (σ_i) allowed. The belief or confidence level of matching the compass signature is then the Gaussian function described in (Equation 5.2). Eventually, a node can require a negative correspondence in the compass range, for example, any orientation but not east to west (EW). To cope with such situations, a third element is added to the compass signature ($csign$) that represents the sign, positive or negative, of the correspondence that is expressed as in (Equation 5.3).

$$compass\_bl_i(\overline{\theta(t)}) = e^{-\frac{(\overline{\theta(t)}-\hat{\theta}_i)^2}{2*\sigma_i^2}} \tag{5.2}$$

$$compass\_bl_i(\neg\overline{\theta(t)}) = 1 - e^{-\frac{(\overline{\theta(t)}-\hat{\theta}_i)^2}{2*\sigma_i^2}} \tag{5.3}$$

$\rho_{dead\_end}(i) = \{0,1\}$: this signature component just describes if a dead-end should be expected in the node.

Node behaviour

Each node has a number of predecessors and successors only limited by the magnetic orientations in which the different locations can be nominally followed. As mentioned before, the concrete environment treated in this experimentation has very strong regular properties and a high degree of symmetry, which allows the setting of only four nominal orientations: NS, SN, EW and WE.

Nodes in Figure 5.4 marked with C, H and T symbols belong to corridor, hall and transition locations respectively. These places are different not only with regards to the signatures of the nodes, but also in the kind of behaviour the robot should show in each type of location. As a result, nodes have distinct functionalities; it is sufficient to assign different α_i functions to the nodes. For the example described here, the three different functions described bellow are distinguished.

α_{corr}: the fact that most of the offices are placed in corridors makes these locations very significant places. When situated in a corridor, the robot will simply follow it to the end, no matter what kind of end, a dead-end or a non corridor place it finds. It will just follow the nominal orientation set by the node.

α_{trans}: they are hall locations where the nominal orientation to follow is perpendicular to the one needed to traverse halls. When the robot is in such places, it must decide whether it has to go towards the hall or continue to reach the next immediate corridor. The function to be executed in transitions must select the neighbouring node to direct the robot to. Therefore, it will choose the orientation to the appropriate neighbour, depending on the goal. Notice that there is no need of transitions to enter the corridors once the hall is traversed, because the successors have opposites nominal orientations that make them distinguishable.

α_{hall}: the decision to be made is reduced to selecting the corridor the robot should be directed to. Because the robot lacks the ability to identify crossroads based on geometric information, we need to make use of the emergency exit panels located approximately in the centre of both halls to anticipate them. To do so, the robot uses the camera mounted on its top. Before perceiving the panels, the robot will just follow the nominal orientation. Once the visual landmark is perceived, the robot's trajectory will draw a diagonal towards the correct corridor just by adding or subtracting the standard deviation associated to the compass signature of the node.

The proposed map, composed by twelve nodes of type C, four T and four H nodes, amounts to 20 processes (threads) launched with different parameters.

Figure 5.5 shows the type of locations defined and their corresponding functions according to the way the robot has to behave.

Galtxagorri's empirical evaluation

The goal of this experiment is to enable the robot to navigate autonomously in the described office-like semi-structured environment. To test the overall navigational system, three routes were defined (see Figure 5.6) and each one tested 8 times in 4 different days, for a total amount of about 8 hours of successful navigation and localisation. The experiments were performed in a real 65×25 meter environment and each path length exceeded 300 meters.

Table 5.1 reflects the standard deviation during each trial (mean velocity: 0.21m/s). As it can be seen, there are no significant differences in those mean values and the deviation can be considered small, what confirms behaviour repeatability.

Figure 5.6(a), 5.6(b) and 5.6(c) reflect robot's behaviour during routes

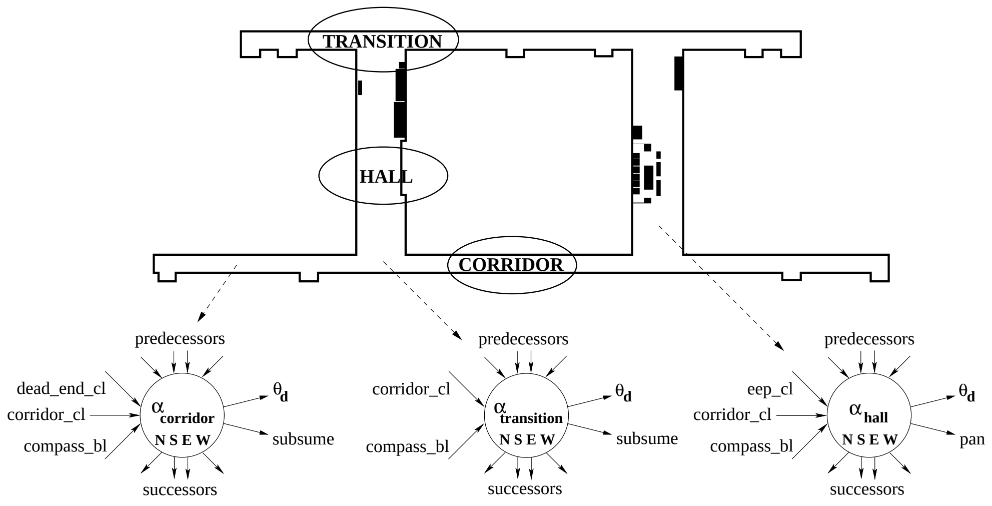


Figure 5.5: Type of locations with their associated functions

	day			
	#1	#2	#3	#4
Route R1	± 0.05	± 0.05	± 0.05	± 0.06
	± 0.05	± 0.05	± 0.07	± 0.07
Route R2	± 0.06	± 0.06	± 0.07	± 0.07
	± 0.06	± 0.05	± 0.07	± 0.07
Route R3	± 0.06	± 0.07	± 0.06	± 0.06
	± 0.06	± 0.06	± 0.06	± 0.07

Table 5.1: Standard deviation from mean velocity in each trial

R1, R2 and R3 respectively. Each of the routes is composed by a set of subgoals that the robot should achieve. For example, subgoals of R1 are: $(0,0),(4,0),(4,2),(0,2)$ and back to $(0,0)$.

More complex experiments where performed to measure the persistent behaviour of the robot, shown in Figures 5.7 to 5.9. Every figure contains the node activation sequence during the route on the left side, and the trajectory followed by the robot on the right, drawn using compass angle instead of odometry reference for rotational information. The node numbering corresponds to the numbers written down beside each node on the topological representation of the environment in Figure 5.4(b). In Figure 5.7, the goal is maintained and so it is the robot's position. Once the robot leaves the corridor and enters the transition, it is reoriented to the goal corridor through the "subsume" output of the T node. Notice the node sequence iteration from second 1100 and ahead; chain 0, 16, 1 is repeated five times. Figure 5.8 illustrates a R3 route iterated over one hour.

On the other hand, Figure 5.9 reflects the robot ability to reach the objective even when its natural way is blocked. In this case, during route R1 the entry to location $(2, 0)$ (the way from T-node 16 to C-node 2) has been blocked using a screen, which does not let the robot to go into the middle corridor. Local navigation strategies act directing the robot to the hall and, once the corresponding node becomes active, it directs the robot to the upper corridor to be able to reach the goal situated in the lower corridor $(4, 0)$.

It should be pointed out that the overall control architecture, composed of 34 threads each one responsible for its own particular task, together with the *Player* server involves only a 13% of the CPU available in the Pentium III on-board computer of the robot. An 85% remains unused while the rest is required by the OS.

5.6 Galtxagorri's shortcomings

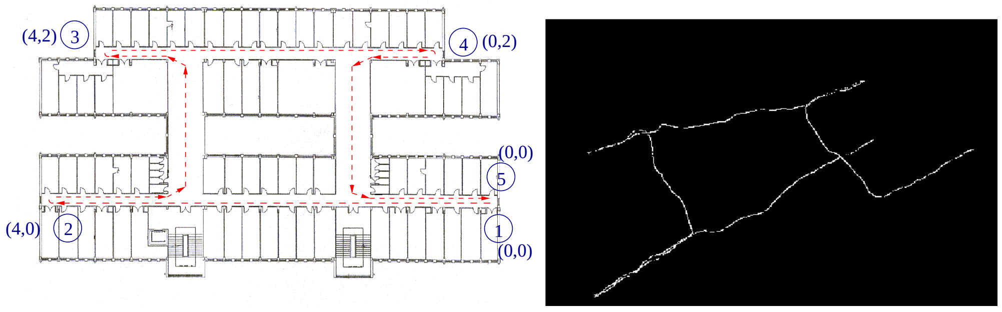
As stated earlier, the control architecture of Galtxagorri was the starting point of the development of the present research work. Thus, first the limitations of the architecture were identified to increase its functionalities and improve its performance. The main shortcomings of the topological representation implemented in Galtxagorri were the following ones:

1. The low resolution of the topological map is due to the lack of landmark identification processes that would allow the robot to access the rooms and labs.

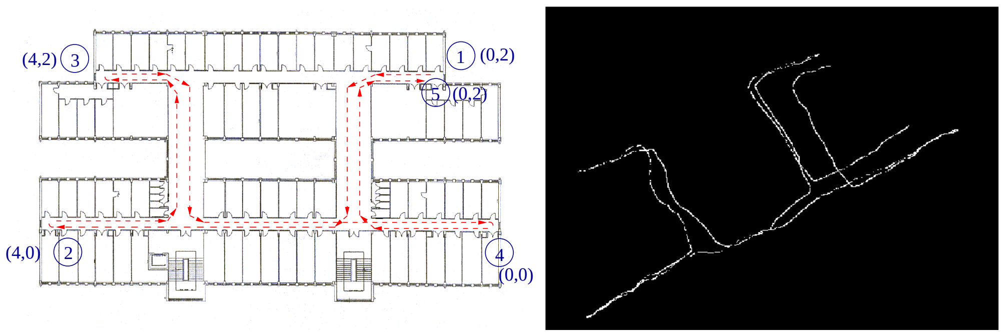
2. The deterministic nature of the FSA that controls the localisation of the robot. Only one node stays active at a time and only its immediate successors are alerted by the wake-up signals. This property makes the robot both unable to recover its location when for an unpredictable reason none of the successors landmarks is recognised and the robot gets lost, and unable to cope with the global localisation problem.

3. Galtxagorri has not an automatic mapping mechanism. The topological map is given to the robot. This is a big burden for the designer of the map because robots' perceptions differs highly from humans' perceptions. An automatic mapping method should be developed for the robot to acquire the node information and the spatial relationship among the nodes.

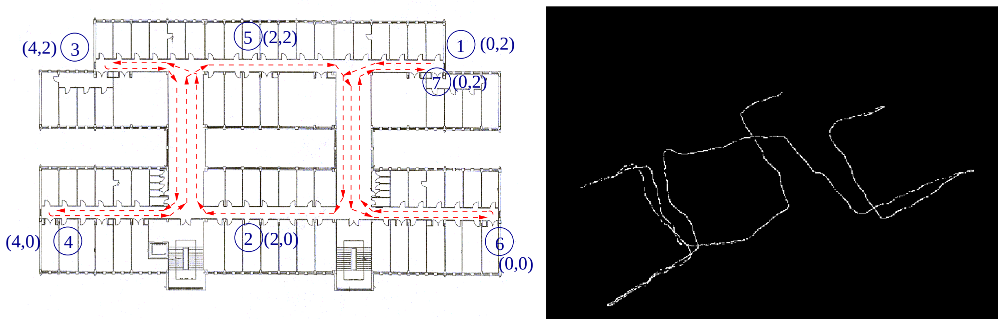
The following chapters explain how these limitations have been approached in this research work. In one side, Chapters 6 to 8 describe the work done to implement a door identification system that should serve for increasing the



(a) Route R1: (0, 0), (4, 0), (4, 2), (0, 2), (0, 0)



(b) Route R2: (0, 2), (4, 0), (4, 2), (0, 0), (0, 2)



(c) Route R3: (0, 2), (2, 0), (4, 2), (4, 0), (2, 2), (0, 0), (0, 2)

Figure 5.6: Three experimental routes

resolution of the topological map. On the other side, Chapter 9 describes how the Markov localisation method has been integrated in Galtxagorri's control architecture in order to cope with the global localisation problem, whereas in Chapter 10 a statistical method is applied to acquire the set of nodes that composes the topological map.

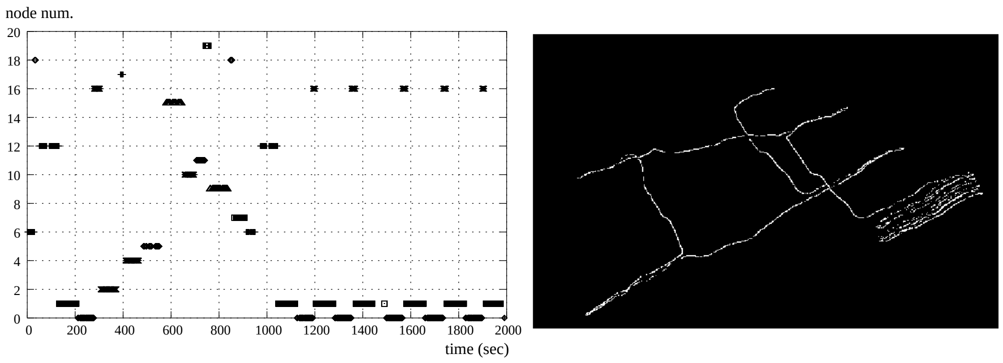


Figure 5.7: R1: persistence towards goal

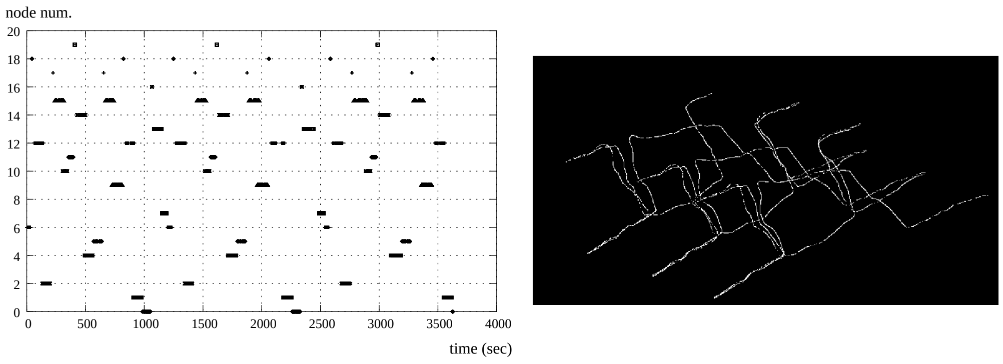


Figure 5.8: R3: persistence over time

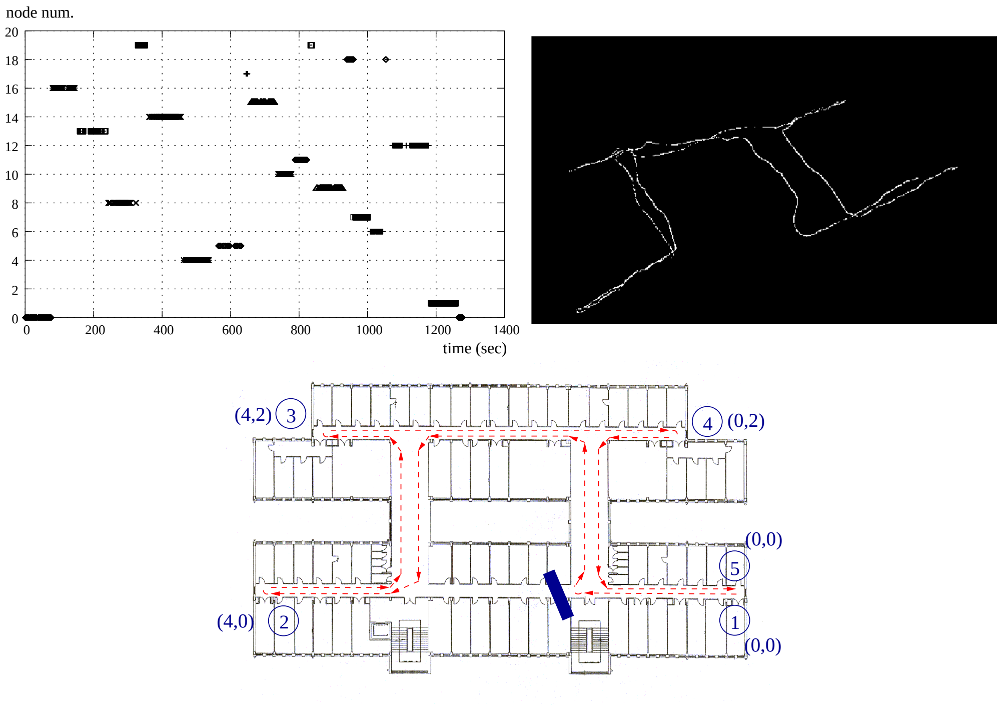


Figure 5.9: R1: persistence in spite of way blocking

Part III

Approaches to door handle identification

Chapter 6

Introduction to door identification

Contents

Indoor semi-structured environments are full of corridors that connect different offices and laboratories where doors give access to the locations that are defined as goals for the robot. Since navigation tasks can be accomplished with point to point navigation, door identification and door crossing (Li et al., 2004), endowing the robot with door identification ability would undoubtedly increase the navigating capabilities of the robot.

6.1 Literature review

There are several references in the literature that tackle the problem of door recognition and crossing. Most of them assume a pre-map where the location of the doors is represented. Stella et al. (1996) have developed SAURO, an autonomous robotic system designed to transport tasks in indoor environments. SAURO uses vision to locate the robot and verify that the path planning is done properly. The authors use an obstacle avoidance system based on ultrasonic sensor readings and the doors are equipped with several artificial landmarks. If, according to the planner, the robot must cross a door, the crossing doors subsystem is activated and the vision subsystem is used to align the orientation of the robot with the axis of the door. Once aligned, a straight motion allows the robot to safely cross the door. The vision module maintains the proper alignment of the robot to avoid collisions.

Eberset et al. (2000) present also a vision based system to identify gates and crossings. The door structure is extracted using a filtering method of parallel lines and then the robot is driven through the door *tracking* the segmented lines. The last step, which corresponds to the stretch where the lines can not be detected because they are outside the focus of the camera, is performed blindly, using *dead-reckoning* and the ultrasonic sensors to ensure that the robot does not collide with people or objects that may be blocking the door.

Stoeter and Papanikolopoulos (2000) use a "ranger" robot that is in charge of launching smaller robots in environmental conditions that could be harmful to human beings. This "ranger" needs to recognise its position in the environment to decide where to throw the "scouts". To do this, the doors are identified using vision. Firstly the image is thresholded and then an edge closure is applied, with a generalised dilation followed by a generalised erosion. Finally segments of vertical orientation are detected and marked. Afterwards, the status of the doors (open or closed) is evaluated using ultrasonic sensors.

Monasterio et al. (2002) present a system for identifying and crossing open doors. The textureless surfaces of the door blades allow the robot to approach the location of the aperture of an open door applying vertical line detectors to images. Once the aperture is approached, a neural classifier is applied to confirm the existence of a door opposite the robot. This approach is prone to give false positives when the environment is not textureless.

The system proposed by Kragic et al. (2002) differs from previous ones in the sense that the doors are located on a map and do not need to be recognised. To find the handles in order to open the doors, they are identified using an integration of features by consensus: for each pixel, the probability of belonging to a handle is calculated by combining the gradient and the intensity of each pixel and a degree of membership is obtained. Then, the authors use a sample model to obtain consensus in a region.

In (Seo et al., 2005) a Principal Component Analysis (PCA) is performed to detect patterns of doors in the images obtained from the robot. A fuzzy controller based on the ultrasound readings ensures that the robot avoids obstacles and crosses the doors. This fuzzy controller selects the direction (heading) of the robot and the linear velocity.

Muñoz-Salinas et al. (2005) mention that the detection of doors can be useful for the mapping and localisation processes of the robot. They present a visual detection system based on the lines of the door with a Canny edge detector and the Hough transform to extract line segments from images. Using characteristics of the extracted lines, a fuzzy logic system is applied to analyse the relationship between the size, the direction or the distance among lines to establish the existence of a door.

Lazkano et al. (2006b) propose a method to obtain the structure of a Bayesian network that allows the robot to cross doors, but the assumption is that the doors are within the perceptual range of the robot and opened. The obtained behaviour allows the robot to cross the door smoothly without colliding with the panels of the door and keeping the trajectory to the target.

More recently Rusu et al. (2009) present a laser-based approach for door and handle identification. The laser sensor is mounted on a tilting platform that is tilted up and down continuously to create a 3D view of the area ahead of the robot. Geometric attributes such as door planes, their boundaries and locations with respect to the world coordinate system are used to select the current closest door in the map. The handle detector is invoked in the proximity of a door which has been detected as a candidate. Geometric information together with surface reflectivity information provided by the laser scans is used to select the actual handle.

6.2 The door identification task

Doors in indoor environments are not necessarily uniform; door panels can be of different texture and colour and handles can vary in shape. Specific methods can be designed for each type of door. But although feature-based methods are easily applicable to different object recognition tasks, they are computationally expensive and therefore less attractive to be applied for real-time problems such as for mobile robot navigation.

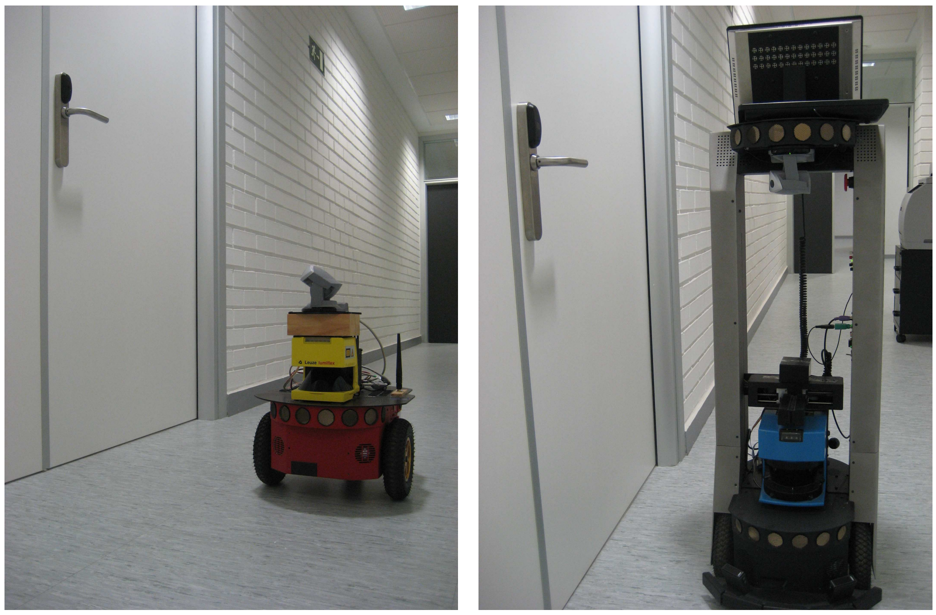
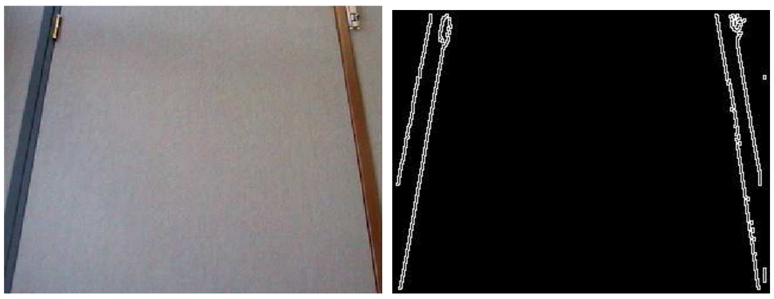

(a) Galtxagorri's viewpoint (b) Tartalo's viewpoint

Figure 6.1: Position of the camera in the robots

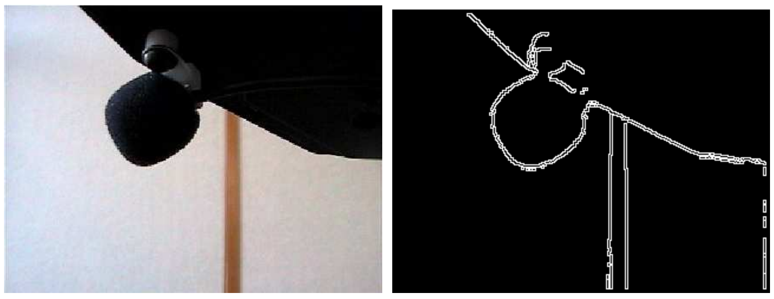
A common intuition to detect doors is to use line extraction processes to identify the door frame. But this approach is not applicable neither in Galtxagorri nor in Tartalo given that the placement of the cameras on the robots provide a limited point of view and the images obtained include partial views of the door frame (see Figure 6.1).

The extraction process is made more difficult in narrow corridors because of the inadequate camera focus that results from the secure distance the robot has to keep to the walls when using its basic navigation skills (Figure 6.2).

Alternatively, door handles can be used for the same task. Handles are naturally located to be easily reached by human beings to open the doors. This placement also has the advantage that is more easily captured by the vision systems our robots are equipped with. To be more precise, Tartalo's camera is located at approximately the same height as the handles and it can



(a) Galtxagorri can not see the complete door frame due to its height



(b) Tartalo can not see the door frame due to occlusions

Figure 6.2: Door frame extraction examples. Left: originals. Right: after applying the Canny edge detector

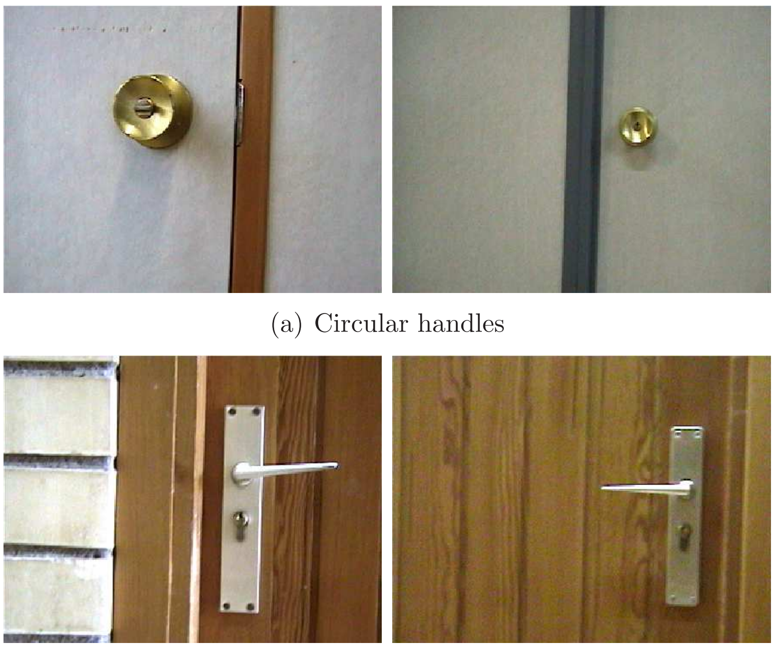
be assumed that when the camera captures an image of a door that contains an object, that object will with all probability be a handle.

The objective of the work described in this part of the essay is to develop the necessary behaviour modules that will allow the robot to extract the necessary characteristics of the different types of handles and therefore, to robustly identify doors.

6.2.1 Handles at the Faculty of Computer Science

Before its complete restructuring, the Faculty of Computer Science had mainly two types of handles. Circular handles which were fitted on pladour doors painted in light grey; and rectangular handles, which were placed on wooden doors (see Figure 6.3).

The door identification methods explained in Chapter 8 were developed for the identification of these two types of handles. Some time later, the inside of the Faculty of Computer Science was completely rebuilt. The wooden and grey pladour doors were removed and new ones were installed. All the new doors are now uniform and have the same type of handle (see Figure 6.4). These new doors have been used afterwards to test the generalisation


(a) Circular handles

(b) Rectangular handles

Figure 6.3: Handles at the Faculty of Computer Science before 2009

capability of a new two-step algorithm that combines region extraction and feature extraction explained in section 8.3.

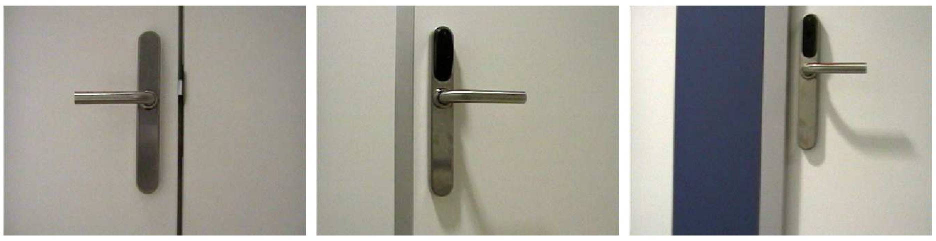


Figure 6.4: Handles at the Faculty of Computer Science after 2009

Chapter 7

Review of the image processing methods used

Contents

The most important challenge for a mobile robot to be fully autonomous is to be integrated into humans life, to perform naturally in it and to comfortably interact with humans. Vision sensors are meant to provide robots with the fundamental capability to identify people and objects to allow them to carry out useful tasks and to reach the desired human-robot interaction level dreamt.

Computer Vision can be defined as the science that develops the theoretical and algorithmic basis which is used to automatically extract and analise information about the real world from an observed image (Haralick and Shapiro, 1992). In the early 70s, computer vision was viewed as the visual input to mimic human intelligence and endow robots with intelligent behaviour. Computer vision is being used today in a wide variety of real-world applications, such as optical character recognition, machine inspection, retailing, 3D model building, medical imaging, automotive safety, surveillance and of course, robotics.

During this research work several vision algorithms have been used. These algorithms can be grouped into two main sets: segmentation techniques and feature extraction techniques. This chapter shortly reviews these methods.

7.1 Image segmentation techniques

Image segmentation is a major research topic since the earliest days of computer vision and it is one of the most extensively studied problems. Segmentation subdivides an image into its constituent parts or objects, i.e. it is the task of finding groups of pixels that “go together” (Szelisky, 2010). The level to which this subdivision happens depends on the objective of the segmentation and is certainly one of the most difficult tasks to tackle.

Segmentation algorithms for monochrome images are based on two basic properties of grey values: discontinuities and similarities (Gonzalez and Woods, 1993).

1. Discontinuity based segmentation partitions the image according to abrupt changes in grey level. There are three basic types of discontinuities in digital images: points, lines and edges. The isolated points and lines are not frequent in most practical applications, so edge detection is by far the most common approach for the detection of discontinuities. The edges represent the boundaries between two regions with different properties of grey level. They are the most widely used segmentation technique in practise.

Due to the inherent noise in the images, the methods that detect discontinuities rarely localise an edge completely. Hence, the edge detection algorithms need additional methods to connect the pixels associated with the edges by means of local or global processing techniques, such as the Hough transform, graph theory methods, etc.

2. Similarity based segmentation attempts to group similar regions using thresholding, region growing, and region splitting and merging. *Thresholding* is one of the most important segmentation methods. Unfortunately, a single threshold is rarely sufficient to segment the whole image. Given the fundamental role of lighting in the segmentation result, optimal and adaptive methods have been developed that allow the adaptation of the threshold value depending on the intensity of the image. Another option consists on grouping pixels or subregions into larger regions. This is known as *region growing*.

   An alternative to region growing is the division of an image into a set of arbitrary and disjointed regions which are then merged and/or split more until they fulfil the desired criteria. A common way to do this is to subdivide the image successively into smaller quadrants until one of them fulfils a certain predicate. If none of the quadrants fulfil the predicate, the process is repeated for each quadrant. Since this approach can lead to a division in which there are adjacent regions with identical properties, it is necessary to allow the union of adjacent regions with the same properties, i.e. merging only adjacent regions that fulfil the certain predicate (Gonzalez and Woods, 1993).

Whereas there is a large variety of segmentation algorithms for monochrome images, the problem of segmenting colour images is more complex and requires more thorough information about the objects. Colour segmentation is defined as the process of extracting one or more connected regions that satisfy a uniformity criterion based on characteristics derived from spectral components (Skarbek and Koschan, 1994). The colour space chosen to represent the image plays a key role.

The colour segmentation techniques can be classified into three groups:

1. Histogram-based techniques: one or more peaks in the colour space are identified. These peaks are then used to classify the pixels.

2. Clustering techniques: the pixel values are grouped according to the representatives previously obtained.

3. Fuzzy clustering: membership fuzzy functions are evaluated for each pixel and for each of the defined clusters.

In the following subsections the main segmentation techniques used in this research work are described.

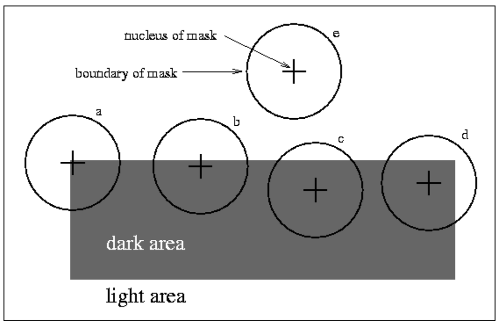
7.1.1 The SUSAN operator

SUSAN (*Smallest Univalue Segment Assimilating Nucleus*) is an operator that detects edges and corners, and for structure preserving noise reduction (Smith and Brady, 1997).

The most primitive methods to enhance edges in images use convolution masks to approximate the first derivative of the luminosity function of the image. But these filters give very little control over the location and smoothness of the edges.

The principle of SUSAN is based on the fact that each image point has an associated local area with the same brightness. This area is known as USAN (Univalue Segment Assimilating Nucleus) and contains a lot of information about the structure of the image. Using the size, the centroid and the second moment of USAN, 2D features and edges can be detected.

Figure 7.1(a) shows a binary image that contains a black rectangle in the lower part. Four circular masks located at different points are also shown. Comparing the value of the centroid of each mask with the rest of the points associated with it, the USAN region of each central point is obtained (the white areas inside the circles in Figure 7.1(b)).



(a) Original image with circular masks

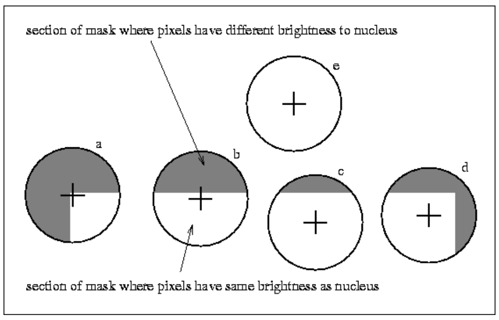

section of mask where pixels have different brightness to nucleus
e
a
b
c
d
section of mask where pixels have same brightness as nucleus
(b) USAN regions of each mask

Figure 7.1: Regions used for a simple binary image

The USAN area falls as it approaches an edge and it reaches the minimum value at the exact position of the edge. The same happens with the corners, produces a local minimum. Processing an image to obtain the inverted USAN area, would produce enhanced edges and with the 2D features more intensively enhanced than edges. This is precisely the SUSAN principle.

The algorithm to filter an image with the SUSAN operator works as follows:

- Place a circular mask in the pixel. Generally, a circular mask of 37 points is used.

- Calculate the USAN of the pixel, $n(\vec{r_0})$, according to Eq. 7.1.

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0}), \text{ where } c(\vec{r}, \vec{r_0}) = e^{-(\frac{I(\vec{r})-I(\vec{r_0})}{t})^6} \tag{7.1}$$

  where $\vec{r_0}$ is the position of the nucleus in the 2D image, $\vec{r}$ is the position of any other point and $I()$ is the image intensity value. The threshold t indicates the minimum contrast of edges to be drawn.

- Subtract the size of USAN to the geometric threshold to produce the initial response to the edges, as follows:

$$R(\vec{r_0}) = \begin{cases} g - n(\vec{r_0}) \text{ if } g > n(\vec{r_0}) \\ 0 \text{ if } g \leq n(\vec{r_0}) \end{cases}$$

  being $g = 3n_{max}/4$ the geometric threshold, n_{max} the maximum value which n (the number of pixels in the USAN) can take.

- Use the calculations of the applied moment of USAN to find the direction of the edges. Although the SUSAN principle does not require to find the direction, it can be computed from USAN.

- Apply non-maximum suppression, thinning and sub-pixel estimation, if required.

7.1.2 Hough transform

The Hough transform is a method originally developed to identify lines in images. It can be used to group edges into line segments even across gaps or occlusions. To do so, firstly a method to detect the edges or points such as *Canny* or *Susan* must be applied. The Hough transform is a well-known technique to have edges "vote" as plausible line locations (Ballard, 1981).

Let's assume that polar representation of lines is used: $x \cos\theta + y \sin\theta = \rho$. A horizontal line corresponds to $\theta = 0, \rho = x$ and a vertical with $\theta = \pm 90°, \rho = \pm y$. The parameter space is then a set of sinusoidal curves in the $\rho\theta$ plane. The Hough transform subdivides the parameter space in an array

Algorithm 7.1 The Hough transform

```
1. Clear the accumulator array.
2. For each detected edge at location (x,y) and orientation $\theta$, compute
   the value
       $\rho = x\cos\theta + y\sin\theta$
   and increment the accumulator corresponding to $(\rho, \theta)$. The range
   of angle $\theta$ is $\in -90°, \cdots, 90°$.
3. Find the peaks that correspond to lines in the accumulator.
4. Optionally re-fit the lines to the constituent edges.
```

of cells called *accumulator* and represented as $A(i, j)$ proceeding as shown in Algorithm 7.1.

Incrementing θ and solving for ρ (step 2) would produce M entries in accumulator $A(i, j)$ – associated with the cell (ρ_i, θ_j) – which would indicate that M collinear points lie on a line $x\cos\theta_j + y\sin\theta_j = \rho_i$.

The computational cost of applying the Hough transform is $O(n)$ linear, being n the number of edge points detected in the image.

7.1.3 Circular Hough transform

The Hough transform is applicable to any function of the form $g(v, c) = 0$ where v is a vector of coordinates and c is a vector of coefficients. A circle in the image is described as $(x - a)^2 + (y - b)^2 = r^2$ where (a, b) are the coordinates of the centre of the circle and r is the radius. Thus, an arbitrary point (x, y) is transformed into a circular cone in the space of parameters (a, b, r) (see Figure 7.2). Thus, the parameter space of a circle belongs to $\Re^3$.

As the number of parameters needed to describe the shape increases, also does the computational complexity of the Hough transform. To simplify the parametric representation, the radius can be considered constant or limited to a known range.

The process to find circles through Circular Hough Transform (CHT) is as follows:

1. Find the edges with a suitable operator (Canny, Sobel, or morphological operations).

2. For each point that belongs to the edges, draw a circle on its centre

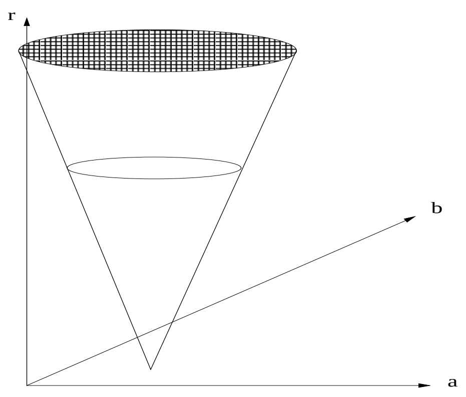


Figure 7.2: Space parameters of CHT

and of the desired radius, increasing all the entries of the accumulator that belong to the coordinates of the perimeter of the circle.

3. Find one or more maximum peaks in the accumulator. This step is extremely important. The preferred method is to apply a second step in which the maximum accumulator value voted by the edge point is identified.

It must be taken into account that if the radius is not fixed, the size of the accumulator can be very large. Depending on the size of the image and the number of the allowed radius, the computation time of the CHT can be excessive.

In recent years different methods to detect circles based on the Hough transform and several techniques to speed up its computation have been developed (Yuen et al., 1990). One of them is the *2-1 Hough Transform*, which reduces the requirements of the space splitting the circle detection problem in two phases. As the centre of the circle must fall into the normal of each point of the circle, the intersection of all normals will be the centre of the circle. Hence, a 2D accumulator array is needed to accumulate the votes of the normal of each point. To identify the radius, the distance of each point from a possible centre (candidate) is calculated and a histogram of radius is produced. Thus, the storage space required is small and only a 2D accumulator and a uni-dimensional histogram is needed.

7.1.4 Localisation of continuous regions: blob detection

A *blob* (Binary Large OBject) is an area of touching pixels with the same logical state. Blob extraction, also known as region detection or labelling, is an image segmentation technique that categorises the pixels of an image as belonging to one of many discrete regions. The terminology is somewhat inconsistent and the process is often referred to in ways as diverse as *labelling of regions*, *labelling related components*, *detection of blobs* or *extracting regions*.

Originally, the detection of blobs was used to get interesting regions to be processed a posteriori. These regions could indicate the presence of objects or parts in order to track down an interest area (*tracking*), to recognise specific items or to perform texture analysis. More recently, blob descriptors have gained interest. Local statistical measures obtained from the extracted regions are being used for appearance based object recognition.

There are multiple techniques that include sequential and recursive algorithms. Blob detection is generally performed on the resulting binary image from the thresholding or edge detection step. Once the interesting points are identified, pixels are labelled with an identifier which represents the region they belong to (see (Horn, 1986) for more details). To label the pixels, a region growing technique must be applied to group pixels that belong to the same region.

As mentioned before, region growing is a procedure that groups pixels or subregions into larger regions. The simplest technique known as *pixels aggregation* starts with a set of "seed" points. Then the regions are increased by appending to each seed point the neighbour pixels with similar properties, such as grey level, colour or texture (see Algorithm 7.2).

How to choose these seed points and the criteria of similarity are the keys to correctly determine the regions. The choice of the starting points depends on the application and type of image. Usually, the analysis of regions is performed on descriptors based on intensity or spatial properties such as moments or texture. The descriptors should take into account connectivity information or adjacency to avoid producing a meaningless set of regions. Another problem is to decide when the process is finished. The easiest way to decide would be to stop when there are no more pixels that fulfil the criterion. However, adding other criteria related to shape or size of the regions a better performance of the region growing algorithm may be achieved.

7.2 Image matching by scale invariant feature extraction

Visual similarity is a major problem in computer vision which requires the solving of problems such as recognition of objects or scenes, reconstruction of 3D structure from multiple images, matching of stereo images and mo-

Algorithm 7.2 Region growing by pixel aggregation

```
1. Initialisation: Select $N$ seed pixels $S_i$, $i = 1, 2, \cdots, N$, and a threshold $T$. Initialise $N$ regions $R_i^{(0)}$ as the seed pixels $S_i$. Initialise $N$ region mean values $M_i^{(0)}$ as the grey values of $S_i$.
2. At each iteration $k$, find border pixels of each region $R_i^{(k)}$.
3. Search the 8-neighbourhood of each border pixel of each region. Assign a pixel $p = f(x, y)$ to $R_i^{(k)}$ if:
     • $p$ is a 8-neighbour of a border pixel of $R_i^{(k)}$ and
     • $p$ has not been assigned yet and
     • grey value of $p$ is sufficiently close to region mean:
           $|f(x, y) - M_i^{(k)}| \leq T$
4. Stop if no region growing was possible. Otherwise, update $M_i^{(k)}$ and iterate by going to step 2.
```

tion detection. Looking for correspondences between images can be useful for different goals. If two images contain projections of the same element of a scene, then the correspondence will localise or identify these objects. On the other hand, if the images are temporally consecutive, then the correspondence determines the motion, and if they are spatially separated but are simultaneous, then the correspondence determines the stereo-disparity (Lan and Mohr, 1997).

Regardless whether the goal of the correlation it is necessary to define and represent the points of interest to be extracted. It is important that these points are invariant to image distortions, i.e. that changes in the point of view of the camera do not prevent the recognition of the objects. In short, the selected features need to be invariant to scale, rotation and occlusion. The features can be of different nature. The first type of features used were the edges. The Harris corner detector, popular in the 90s for feature extraction, is stable to illumination changes and rotation but unstable to more complicated changes. On the other hand, estimates based on the extraction of local properties can include colour and motion information, region descriptors, stereo-depth data, even combinations of them all.

This study's scope is limited to finding correspondences between two images in order to identify specific objects: the door handles. The following sections summarise the extraction techniques that were used in the experimental phase.

7.2.1 SIFT

SIFT (*Scale-Invariant Feature Transform*) is a vision algorithm proposed by D. Lowe (2004) to extract distinctive features of an image. SIFT can be used as a starting point to find correspondences between images and hence, for object recognition. SIFT descriptors are the most widely used nowadays because, in spite of being computationally expensive, the algorithm offers a good time/performance ratio.

The features extracted by the algorithm are invariant to image scaling and rotation and partially invariant to change in viewpoint and illumination. These invariant characteristics are obtained after a cascade filter, which minimises the cost of extracting them. As a result of this filtering, each image is converted into a set of descriptors.

The SIFT features are extracted following the four steps described in Algorithm 7.3 and explained in more detail bellow.

Algorithm 7.3 SIFT feature extraction

1. Detect the extrema in the scale space: each pixel of the image is compared to its eight neighbours in the current image and to its nine neighbours in the scale above and below, i.e. in the adjacent scales that result from the Gaussian filtering.

2. Locate the *keypoints*: the keypoints are selected from the extrema located in the space of scales.

3. Assign a magnitude and orientation to the keypoints: a gradient orientation histogram is calculated for each keypoint.

4. Obtain a descriptor for each keypoint: each keypoint is represented by a 128 dimensional vector.

Detection of the scale space extrema

This step aims to identify the locations and scales of the keypoints of the same object to make it identifiable from different viewpoints. The process is performed by analysing the *scale space* of the image. The scale space of an image is defined as a function $L(x, y, \sigma)$ resulting from the convolution of variable-scale Gaussian ($G(x, y, \sigma)$) and the image $I(x, y)$ (Eq. 7.2).

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y), \quad G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{7.2}$$

It is possible to use different techniques to detect the locations of the stable points in the space of scales. One such approach, proposed by Lowe (1999) is to extract the scale-space extrema from the result of convolving the DoG (*Difference-of-Gaussians*) function with the image. This function, $D(x,y,\sigma)$ can be computed from the difference of two nearby scales separated by a constant multiplicative factor k (Eq. 7.3).

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma) \tag{7.3}$$

The function D is easy to calculate since it is computed from the smoothed images L. Once we get the smoothed images, D is obtained by simply subtracting those two images. Moreover, the function D is an approximation of the normalised *LoG* (*Laplacian of Gaussian*) function.

That is to say, given the original image which is “blurred” or smudged using Gaussian filters that result in a series of Gaussian images (the scale space produced by cascade filtering), then, subtracting the images of their closest neighbours, a new series of images is produced: the approximated LoG. Figure 7.3 reflects this process. The original image is incrementally convolved to produce a series of images separated by a constant factor k in the scale space. Every octave of the scale space (equivalent to multiplying σ by two) is divided into a number of intervals s so that $k = 2^{\frac{1}{s}}$. At each interval, $(s+3)$ images must be generated to obtain the DoG images.

To detect the local maximum and minimum of $D(x,y,\sigma)$, each point is compared with its eight neighbours on the same scale, and with its nine neighbours on the scales above and below (Figure 7.4). Only the candidates with a higher or smaller value than those of all its neighbours are selected.

Given that the extrema that are close to each other showed to be quite unstable to small perturbations of the image. It is of core importance to establish an adequate sampling frequency to detect the extrema. Although it is possible to get a stable set of descriptors even with ordinary sampling scales (eg., three sampling scales per octave), experimental results (Lowe, 2004) showed that both, the number of keypoints and also the number of descriptors correctly matched increased proportionally to the number of scales sampled. On the other hand and related to the sampling frequency of the spatial domain, it is important to determine the prior smoothing σ which is applied to each image before constructing the representation of the scale space for

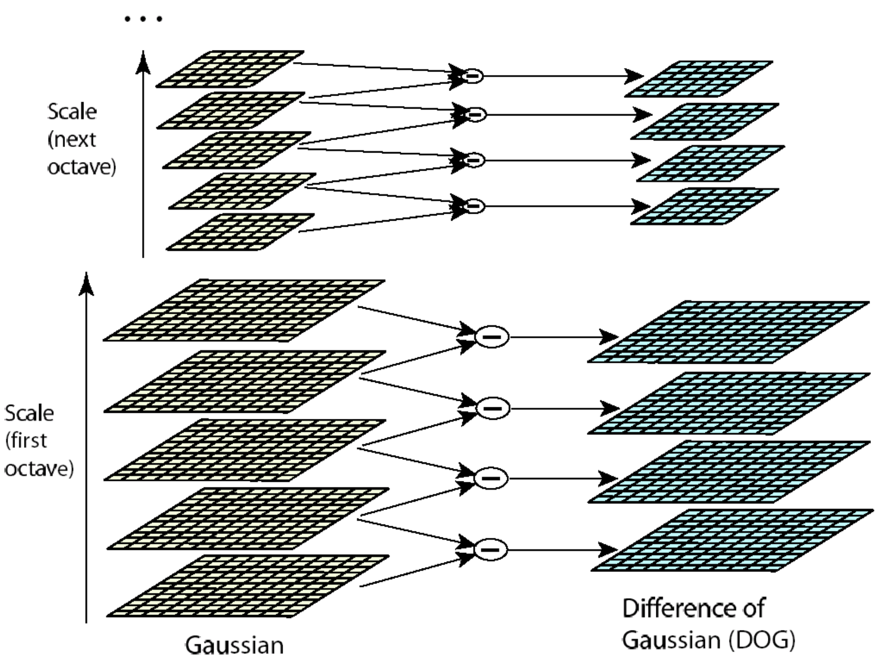


Figure 7.3: DoG function

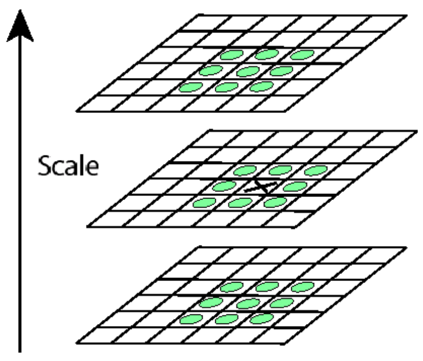


Figure 7.4: Detection of local extremes

each octave. Once more, the replicability of the keypoints increases when increasing σ.

It is important to note that if the image is smoothed prior to detecting the extrema, the higher spatial frequencies are eliminated. That is why usually the size of the original image is doubled using linear interpolation before building the pyramid. This preprocess increases the number of stable keypoints.

Location of the keypoints

The goal of this step is to reject the keypoints that have low contrast or are poorly localised along edges. To determine the interpolated location of the maximum, a Taylor expansion of the scale space function $D(x, y, \sigma) = D(\overline{x})$, shifted so that the origin is at the sample point, is used:

$$D(\overline{x}) = D_0 + \frac{\partial D^T}{\partial \overline{x}}\overline{x} + \frac{1}{2}\overline{x}^T\frac{\partial^2 D}{\partial \overline{x}^2}\overline{x} = D_0 + \nabla D^T\overline{x} + \frac{1}{2}\overline{x}^T H_D \overline{x}$$

where H is the Hessian and ∂D the derivative of D. Note that the function D is shifted so that the origin is at the sample point $\overline{x}$. The new location of the extremum $\hat{\overline{x}}$ is determined by taking the derivative of this function with respect to x and equalling it to zero:

$$\hat{\overline{x}} = -\frac{\partial^2 D^{-1}}{\partial \overline{x}^2}\frac{\partial D}{\partial \overline{x}} = H_D^{-1} \times \nabla D$$

Both the Hessian and the derivative of D can be approximated by using differences between neighbouring sample points. The resulting 3×3 system of equations can be solved with a minimal computational cost.

The function value at the extrema $D(\hat{\overline{x}})$ is used to reject unstable extrema with low contrast:

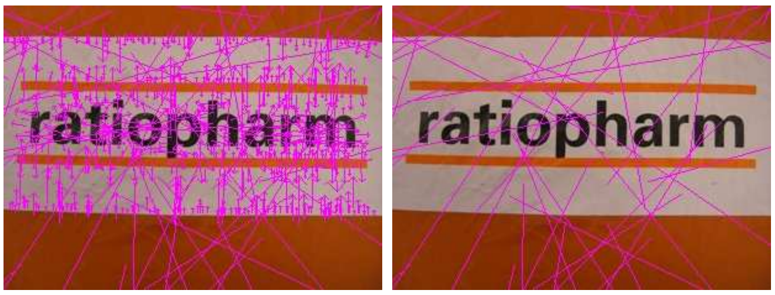
$$|D(\hat{\overline{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \overline{x}}\hat{\overline{x}}| < 0.03$$

On the other hand, the DoG function has a very strong response along the edges, where it identifies many keypoints in them. Hence, it is not sufficient to reject keypoints with low contrast. Taking into account that the eigenvalues of H are proportional to the principal curvatures of D, by simply verifying that the ratio of the principal curvatures is below a threshold ($r = 10$) some keypoints can be eliminated:

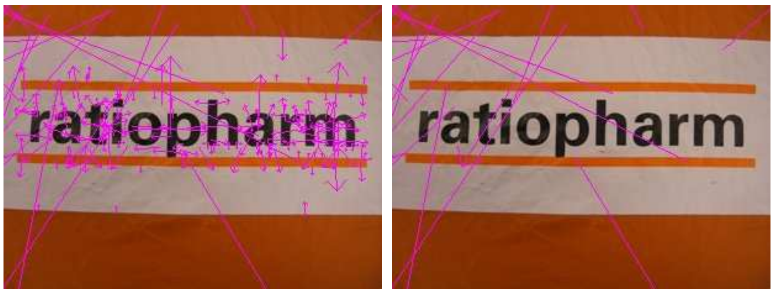
$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}, \begin{cases} Trace(H) = D_{xx} + D_{yy} \\ Det(H) = D_{xx}D_{yy} - D_{xy}^2 \end{cases}$$

$$\frac{Trace(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

Figure 7.5 shows an example of the effect of removing the extrema at the edges. Left hand side figures show the image with the keypoints and right hand side figures the scale-space extrema.



(a) Without eliminating the extrema edges



(b) After eliminating the extrema edges

Figure 7.5: Effect of removing the extrema at the edges

Assign magnitude and orientation to the keypoints

This process follows the next steps:

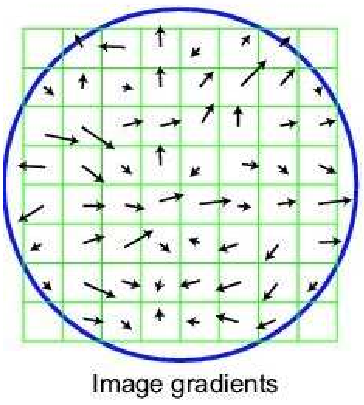
- Choose the smoothed image L based on the closest scale. Thus, all calculations are done keeping the scale invariance.

- For each image sample $L(x, y)$, calculate the magnitude of the gradient $m(x, y)$ and the orientation $\theta(x, y)$:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \arctan \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}$$

  The histogram is formed from the gradient orientations in the region around the keypoint (Figure 7.6). This histogram has 36 entries, covering 360°. Each entry that is added during the calculation of the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window so that σ is 1.5 times the value of the scale of the keypoint.



$$\hat{\vec{x}} = (x, y, s)$$
$$r = 3 \times 1.5 \times s$$
$$\sigma_h = 1.5 \times s$$
$$w = e^{-\frac{d^2}{\sigma_w}}$$
$$histo[bin] += m(i, j) \times w, \;\; i, j \in [-r, r]$$

Figure 7.6: Image gradients around the point

- Smooth the histogram (up to 4 times):

  $$smooth\_histo[i] = 0.25 \times histo[i-1] + 0.5 \times histo[i] + 0.25 \times histo[i+1]$$

- Detect the highest peak in the histogram of orientations: the peaks of this histogram represent the dominant directions of local gradients.
- Generate new keypoints with the orientations within the %80 from the highest peak. In this way, multiple new keypoints are created at the same location and scale but with different orientation. This contributes significantly to the stability in the matching process.
- Fit a parabola to the three closest values of the histogram peak to interpolate the peak position for better accuracy.

Obtain the descriptors

Through the above steps, each keypoint is assigned a position, scale and orientation. These parameters impose a repeatable local 2D coordinate system in which to describe the local image region. The next step generates a descriptor for the local region of the image, distinctive enough and still partially invariant to changes in illumination or 3D viewpoint.

Intuitively, this could be achieved by sampling the local image intensities around the keypoints at the appropriate scale level and then matching them with a proper correlation function. But it is known that the correlation between regions of images is very sensitive to changes that produce variations in the samples.

SIFT descriptors are obtained as follows:

- Sample the image gradient magnitudes and orientations around the keypoint location, using the scale of the keypoint to select the level of Gaussian blur for the image.

  A window of size $n \times n$ is used, which is centred at a circular neighbourhood with radius r,

$$r = \frac{n+1}{2} \times K_s * \sqrt{2} + 0.5$$

  where $K_s = 3 \times scale$.

- Rotate the coordinates of the descriptor and the gradient orientations to achieve orientation invariance.

$$x_p = (c \times \cos\theta - r \times \sin\theta) \times K_s$$
$$y_p = (c \times \sin\theta + r \times \cos\theta) \times K_s$$

- Thus,

$$mag' = \sqrt{(x - x_p)^2 + (y - y_p^2)}, \quad \theta' = \tfrac{y - y_p}{x - x_p}$$
$$\theta_{bin}(\theta' - \theta) \times BINS\_PER\_RADIAN \times \tfrac{bins}{2\Pi}$$

- To avoid sudden changes in the descriptor with respect to small changes in the position of the window and to give less emphasis to gradients that are far from the centre of the descriptor, apply a weight w to the magnitude of each sample point. This weight is equivalent to a Gaussian function with a σ equal to one half the width of the descriptor window.

$$\sigma = 0.5 \times n^2 w = e^{-\frac{x_{bin}^2 + y_{bin}^2}{\sigma}}$$

- After obtaining the gradient magnitudes and the orientations of the samples, four histograms of orientations on the samples of 4×4 regions are created and 8 orientation bins are assigned to each histogram (see Figure 7.7).

- To prevent the gradient samples in the boundary from affecting the descriptor, a trilinear interpolation is used to distribute the value of each gradient sample into adjacent entries of the histogram. Each value corresponding to an input is multiplied by a weight of $(1 - d)$ for each dimension, where d is the distance of the sample from the central value of the histogram.

$$h[x_{bin}][y_{bin}][\theta_{bin}] = tri\_interp(mag' \times w \times (1 - d))$$

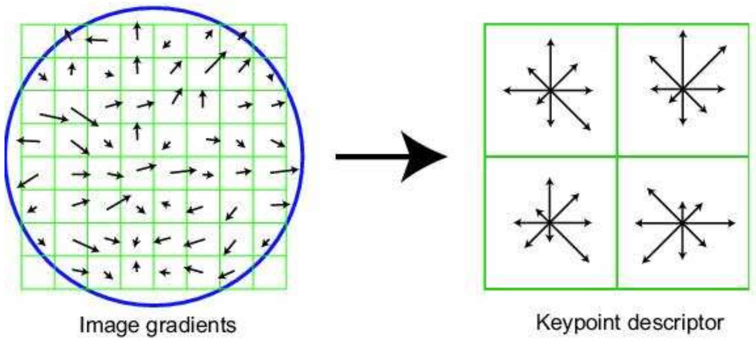


Figure 7.7: Transformation of points in descriptors

As a result, for each keypoint a descriptor vector of 128 elements that can be used to find correspondences between images is obtained for each point. In Lowe's proposal, the matching was done by nearest neighbour search, in addition to the second nearest neighbour from a different object.

7.2.2 SURF

SURF (Speeded Up Robust Features) is another type of robust image descriptor that detects scale and rotation invariant interesting points. It is inspired by the SIFT descriptor, but the standard version is faster than SIFT and apparently more robust to various transformations of images. The SURF descriptors are based on the sums of responses of 2D Haar-wavelets and makes an efficient use of integral images (Bay et al., 2006).

This is achieved by relying on integral images for image convolutions by building on the strengths of the leading existing detectors and descriptors, using a Hessian matrix-based measure for the detector, and a distribution-based descriptor. This leads to a combination of novel detection, description, and matching steps.

Like SIFT, to find matchings with images three stages are needed. First, the interest points at distinctive locations in the image are selected, such as corners, blobs, etc. Then, the neighbourhood of each interesting point is represented using a feature vector. Finally, the descriptor vectors of different images are compared to find correspondences between images.

Interest point detector step

An interest point is a point in an image which has a well-defined position and can be robustly detected, for example maximum local intensity points and corners. A simple approach to corner detection in images is to use correlation, but this is very expensive computationally and suboptimal. A frequently used alternative approach is based on a method proposed by Harris and Stephens (1988), which in turn is an improvement on a method by Moravec (1980).

Moravec's corner detector was based on the local auto-correlation function of a signal, which measures the local changes of the signal with patches shifted by a small amount in different directions.

As pointed out by Moravec, one of the main problems with this operator was that it is not isotropic. If an existing edge does not appear in the direction of the neighbours, then it will not be detected as an interest point. Harris and Stephens (1988) improved upon Moravec's corner detector by taking into account the differential of the corner score with respect to direction, instead of using shifted patches.

However, Harris corners are not scale invariant. Instead, SURF's detector is based on the Hessian matrix because it showed to be more stable and repeatable (Mikolajczyk and Schmid, 2005). Moreover, instead of using the original image to calculate the Hessian matrix, the Hessian of the integral image is calculated, thus reducing the computation time. This process is known as the Fast-Hessian detector. The descriptor of the interest point describes the distribution of Haar-wavelet responses within the interest point neighbourhood.

Fast-Hessian detector

To find the location of the interest points and the scale, the Fast-Hessian detector relies on the determinant of the Hessian. Given a point $X = (x, y)$ in an image I, the Hessian matrix $H(X, \sigma)$ in X at scale σ is defined as follows:

$$H(X,\sigma) = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix}$$

$L_{xx}(X, \sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in point X, and $L_{xy}(X, \sigma)$ and $L_{yy}(X, \sigma)$ are similarly defined. The Gaussians need to be discretised and cropped, so, instead of using this function, the Gaussian second order derivative is obtained by approximating it using a box filter (see Figure 7.8). The approximation is denoted as D_{xx}, D_{yy} and D_{xy}.

In Figure 7.8 a 9×9 box filter has been used with $\sigma = 1.2$, and the weights applied to the rectangular regions are kept simple for computational efficiency, but for the expression of the Hessian's determinant a relative weight

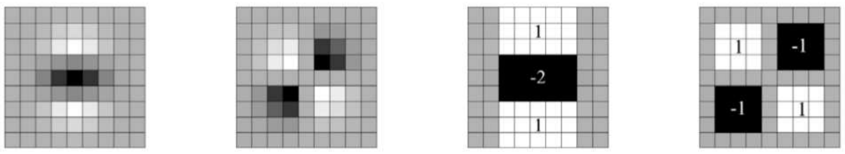


Figure 7.8: Left to right: the (discretised and cropped) Gaussian second order partial derivatives in y-direction and xy-direction, and the approximations thereof using box filters. The grey regions are equal to zero

is needed:

$$\frac{|L_{xy}(1.2)|_F |D_{xx}(9)|_F}{|L_{xx}(1.2)|_F |D_{xy}(9)|_F} = 0.912... \approx 0.9$$

where $|x|_F$ is the *Frobenius* norm. The Hessian's determinant then will be:

$$det(H_{approx}) = D_{xx} D_{yy} - (0.9 D_{xy})^2$$

Finally, to have a constant Frobenius norm for any filter size, the responses of the filter are normalised. The scale space is usually implemented as image pyramids, smoothing the images with Gaussians, but in the case presented here, where box filters and integral images were used, the scale space was analysed by up-scaling the filter size rather than iteratively reducing the image size. The output of the 9×9 filter is considered as the initial scale, with $\sigma = 1.2$, and the next scales layers will be 15×15, 21×21, 27×27, etc.

In order to localise interest points in the image and over the scales, a non-maximum suppression in a $3 \times 3 \times 3$ neighbourhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in the scale and image space.

SURF descriptor

SURF's descriptors are based on similar properties than those of SIFT but instead of being represented with a 128 dimensional vector, the descriptors are defined as 64 dimensional vectors. This reduced size allows the reduction of the matching time when looking for correspondences. The process follows two stages. Firstly, in order to be invariant to rotation, the orientation is fixed based on information from a circular region around the interest point. Afterwards, a square region is aligned to the selected orientation in order to extract the descriptor.

To calculate the orientation, the Haar-wavelet is applied in x and y directions inside a circular neighbourhood of radius $6s$ around each interest point, where s is the scale where the interest point was detected. The Haar-wavelet is scale dependent with $\sigma = 2.5s$ is used and the orientations are represented as vectors. The dominant orientation is estimated by calculating the sum of all responses.

Once the orientation is calculated, to extract the descriptor a square region centred around the interest point is defined. The size of this window is $20s$, and the region is split up regularly into smaller 4×4 square subregions. Inside the square, the Haar-wavelet is applied in horizontal (d_x) and vertical (d_y) directions relative to the selected interest point orientation. To compensate for the geometric deformations and localisation errors, before applying the Haar-wavelet to each subregion the responses of the filter is weighted with a Gaussian ($\sigma = 3.3s$) centred at the interest point. Finally, d_x and d_y are summed up in each subregion, and to include information about the intensity changes in the image, the absolute values $|d_x|$ and $|d_y|$ are also summed, being $v = (\Sigma d_x, \Sigma d_y, \Sigma|d_x|, \Sigma|d_y|)$ the four-dimensional descriptor vector of each 4×4 subregion. The complete descriptor vector is 64 long.

The matching is often based on distance between the vectors, e.g. the Mahalanobis or Euclidean distance. In the matching stage, only features that have the same type of contrast are compared.

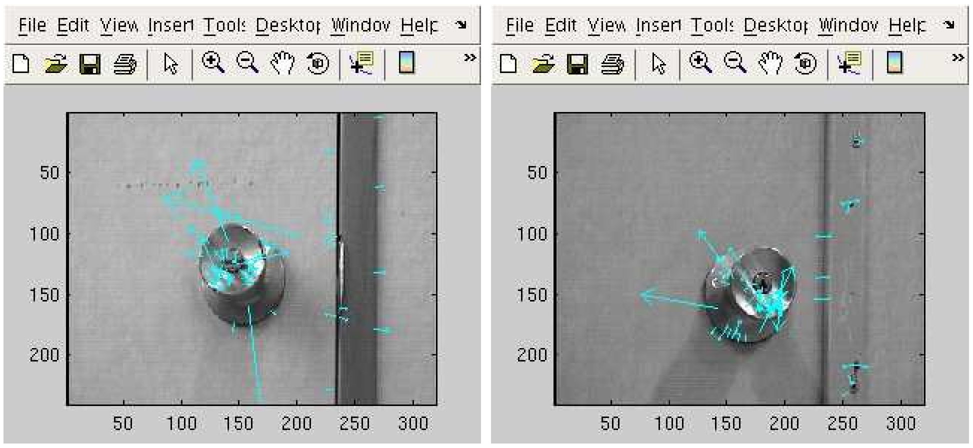
7.2.3 USURF

The SURF descriptors are invariant to scale and rotation, but in some cases, the rotation characteristic is not relevant. Being invariant to rotation means that the camera is going to rotate about its optical axle, but, in many applications, like mobile robot navigation, the camera only rotates about the vertical axis, and thus, the invariance to rotation is of no importance. For those applications where rotation invariance is not needed, a version of SURF named "Upright SURF" (*USURF*) has been developed by Bay et al. (2006). USURF avoids the calculation of the orientation of the keypoints and keypoint extraction process is speeded up.

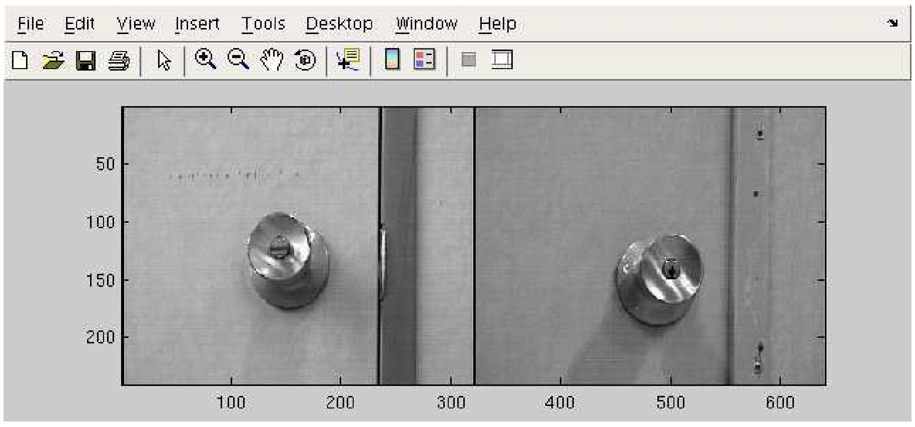
7.2.4 Examples

Figures 7.9 and 7.10 show some examples result of applying SIFT to handle images obtained by the camera of the robot at the Faculty of Computer Science.

It is important to note that the majority of the keypoints of the circular handle images appear on the handle itself due to the textureless surface of the door blades (Figure 7.9(a)). But it must be stressed that it is difficult to find correspondences between two images as shown in Figure 7.9(b).



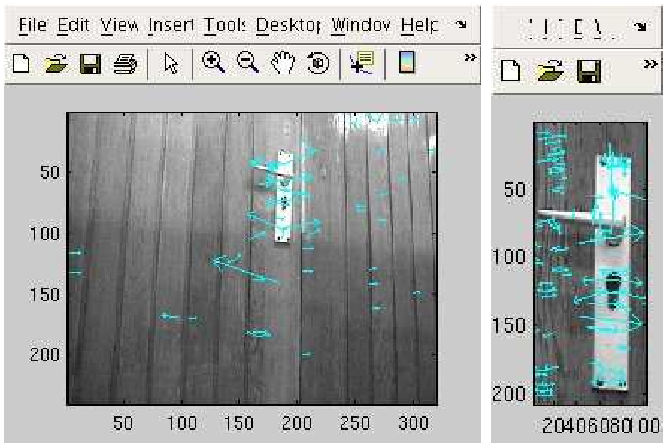
(a) Keypoints are on the handles



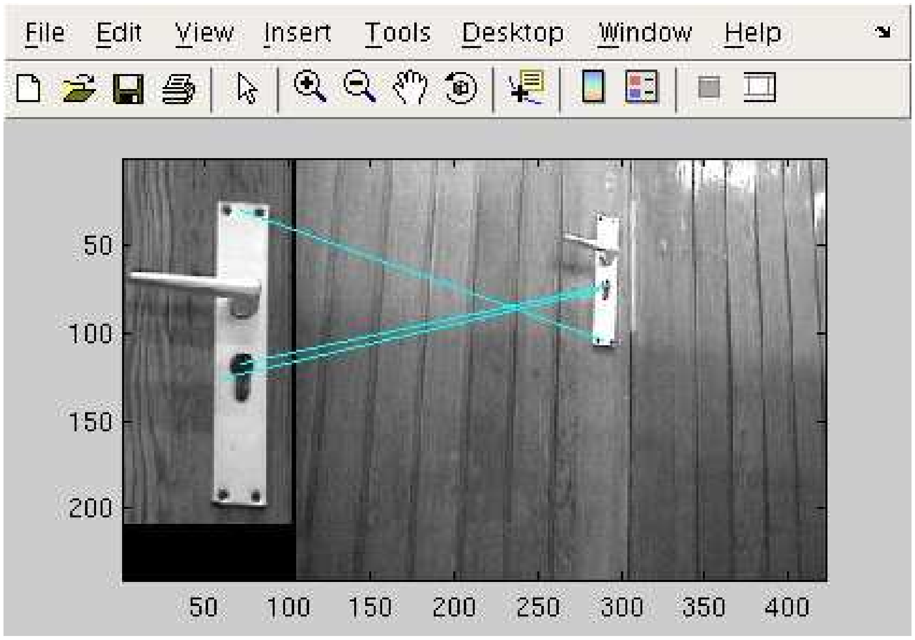
(b) No correspondence is found

Figure 7.9: Circular handles (SIFT)

Rectangular handles are placed in wooden doors, which means that more keypoints that do not belong to the handles are identified. But the correspondences are more frequent and occur mainly in the lock and screws.



(a) Keypoints are on and out of the handles



(b) Correspondences are found

Figure 7.10: Rectangular handles (SIFT)

Chapter 8

Approaches to door identification

Contents

This chapter describes and compares the different approaches that have been developed for door identification, ranging from specific methods based on colour segmentation techniques to more general ones that focus on feature extraction methods.

The first approach consists of a three-stage procedure that relies mainly on colour segmentation. Next, a more appealing feature extraction based identification is presented. Finally, a new two-step multiclassifier algorithm that combines region detection and feature extraction is presented.

8.1 Segmentation based door identification

Often, door panels are made of materials and colours that make them distinguishable from the walls. The segmentation method takes advantage of this environmental property and after selecting door candidates by extracting shapes that can match the proper handle, colour segmentation is used to confirm the presence of door handles in door candidates selected by the extraction of shapes that match possible handles.

8.1.1 Door identification process

In order to identify images containing a door handle a three-stage algorithm has been designed. This three-stage process proceeds as follows:

1. Region detection: this step is designed to approximate the handle area, if any, in the image. The most intuitive approach to detect circular handles seems the extraction of circular edges. The *Circular Hough Transform* (CHT) is an appropriate method to achieve this objective. But handles can also be non circular. A more general method such as *blob extraction* also known as region detection or labelling is needed.

2. Colour segmentation: using the position information of the previous step, the surroundings of the candidate object are segmented in order to see if the candidate handle is surrounded by the appropriate door blade.

3. Statistical validation: measures of the segmented image are then taken to classify an image as containing or not a handle.

Region detection

As mentioned before, some handles are assumed to be circular in shape. Therefore, circle detection can be performed to locate handles in the images taken by the robot. Although many circle extraction methods have been developed, probably the best well-known algorithm is the Circular Hough transform. Moreover, Hough transform methods have shown to be robust enough to deal with noisy images (Ayala-Ramírez et al., 2006).

The use of the Hough transform to detect circles was first outlined by Duda and Hart (1972) and then, Kimme et al. (1975) gave probably the first known application of the Hough transform to detect circles in real images. Later on, Yuen et al. (1990) investigated five circle detection methods which were based on variations of the Hough transform. One of those methods, the *Two stage Hough Transform*, is implemented in the OpenCV vision library (http://www.intel.com/research/opencv) used in the experiments described later on in this chapter.

The circle finding function can identify a huge number of circles depending on the image background. Due to the local navigation strategies of the robot the images will be obtained within the same distance range and therefore, it is possible to know in advance the approximate radius of the handles. Thus, only the identified circumferences with a radius that lies within the known range would be considered as handle candidates.

The CHT only detects circular shapes; hence, an alternative method is needed to first approximate the non circular handles. The method can be generalised by scanning the image for continuous connected regions or *blobs*.

Again, the blob extraction process can give many different regions for a single image. For a blob to be confirmed as a candidate, the result of the blob detection process should be filtered and false positives should be discarded. Different filtering techniques can be used for blob discrimination. For instance, in (Ye and Zhong, 2007) location-related pixel information is used for blob discrimination where the aim is to count people in images taken by a surveillance system. A similar approach is used in our proposal where blobs that are not consistent with the defined size restrictions are discarded. The size restrictions depend on the distance the images are taken from.

Colour segmentation

The region extraction step is not reliable enough neither to confirm nor to reject the presence of a handle in an image. Therefore, its performance needs to be improved with the addition of complementary processes. The approach used in the present research is to use the information provided by the colour which surrounds the handle candidate. Door blades pixels are segmented and then used as discarding criteria: candidate handles not surrounded by the defined blades are rejected as false positives.

Within the robot environment, a circular handle is always associated to pladour door and rectangular ones to wooden doors which present different colour shades according to lighting conditions – e.g. presence or absence of electric or natural lighting.

Segmenting pladour door blades

A supervised machine learning approach has been selected to segment pladour door blades. To build the classifier we chose *Oc1* (Oblique Classifier 1) (Murthy et al., 1994), a decision tree induction system well suited for applications where the instances have numeric feature values. *Oc1* builds decision trees containing linear combinations of one or more attributes at each internal node. The trees built in this way partition the instance space with both oblique and axis-parallel hyperplanes. Images taken by the robot are represented in RGB colour space and thereby, each pixel is a three component vector, where each component takes a value that ranges from 0 to 255.

In every classification problem, a training set is required to obtain a model to be later used when a new instance is presented to the model. To get the training set, we firstly constructed a database of positive instances (those associated to pladour doors), as well as negative instances (those not associated to pladour doors). The size of these databases was about two million pixels, obtained from about sixty images taken by the robot camera in a corridor. From these databases we extracted 80000 pixels randomly, 40000 of them labelled as *pladour* and the remaining 40000 as *non pladour*. Then, these 80000 pixels were input to the *Oc1* tree generation procedure to obtain the decision tree that would be used for the segmentation of the images taken by the robot camera. The obtained performance after applying a 10 fold crossvalidation to this database was 93.61%. Figure 8.1 shows several examples of this segmentation process. The first two rows contain images with handles, although the variable lighting conditions affect the pladour colour recognition process and therefore, the segmentation process. Notice for example the upper right corner of the segmented image in second row. The bottom rows show images without handles. In spite of the fact that the radius exceeds the preset threshold, the original images contain the circles detected by the Hough transform. It can be observed that the segmentation of these images does not imply the existence of any handle.

Segmenting wooden door blades

To segment wooden surfaces and classify their pixels a specific thresholding algorithm has been designed. First of all, using the normalised RGB colour space the following reference values are selected from an image containing

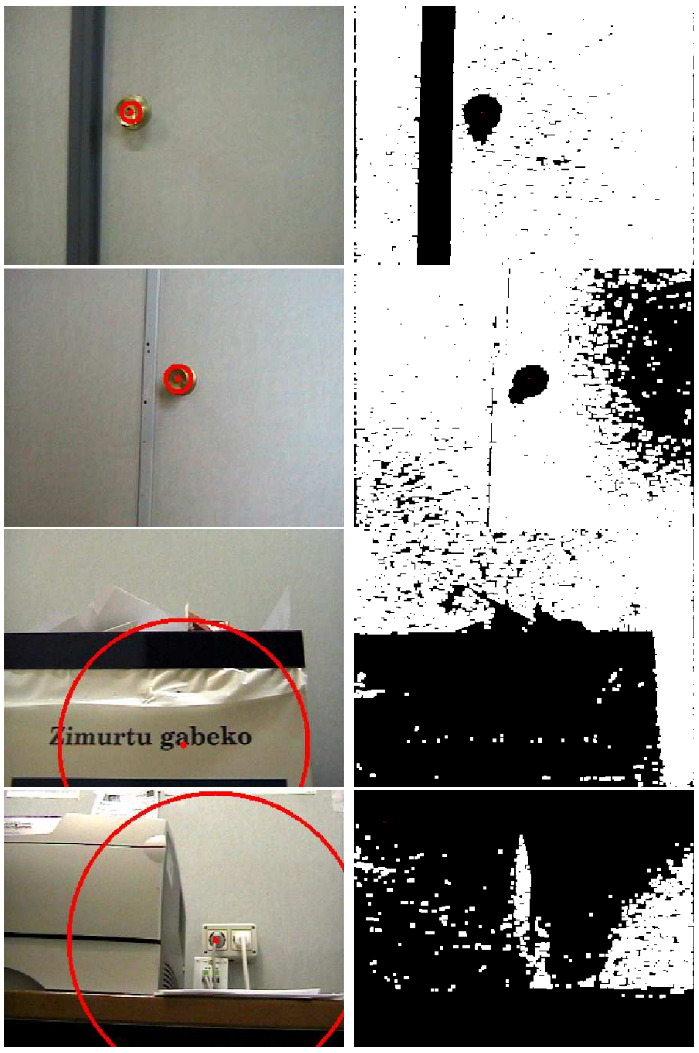


Figure 8.1: Pladour segmentation examples

exclusively wooden coloured pixels:

- R_{min}: minimum value of the red channel.
- G_{min}: minimum value of the green channel.
- B_{min}: minimum value of the blue channel.
- $Rdiff_{min}$: value of the minimum difference among the red and the maximum of the green and blue.
- $Gdiff_{min}$: value of the minimum difference among the green and the maximum of the red and blue.

- $Bdiff_{min}$: value of the minimum difference among the blue and the maximum of the green and red.

Then, given a new image, a pixel will be labelled as belonging to a wooden surface according to Algorithm 8.1. The process is performed using the normalised RGB colour space, and Figure 8.2 shows some examples of this segmentation process.

Algorithm 8.1 Wooden surface segmentation

```
wood_segmenter(R, G, B, $R_{min}$, $Rdiff_{min}$, $G_{min}$, $Gdiff_{min}$, $B_{min}$, $Bdiff_{min}$)
$Rdiff = R - max(G, B)$;
$Gdiff = G - max(R, B)$;
$Bdiff = B - max(R, G)$;
**if** ($R > R_{min}$ and $G > G_{min}$ and $B > B_{min}$ and $Rdiff > Rdiff_{min}$ and $Gdiff > Gdiff_{min}$ and $Bdiff > Bdiff_{min}$) **then**
  return 1;
**end if**
return 0;
```

Statistics for decision making

So far we have described two procedures to recognise the handle and its surroundings. However, both are prone to errors due to noise, lighting conditions and the presence of objects in the environment (printers, dustbins, tables and so on). As we cannot entirely rely on their accuracy, a third step is needed to analyse the segmented pixels of the candidate handle and of its surrounding in order to definitively confirm or reject the handle candidate. The segmentation process produces a black-and-white image, where white points are those classified as belonging to the *door blade*, and black ones are those classified as not belonging to it. To analyse the surroundings of the candidate handle, the centre (x_0, y_0) is obtained using the information of the detected region, and then the following values are calculated (see Figure 8.3):

- *Perc1*: percentage of white points in the whole image. When the robot is close to a door, a huge percentage of the pixels in its visual field are likely to be segmented as white. So the bigger *Perc1*, the more likely the robot is opposite a pladour door.

- *Perc2*: the pixels around the centre should belong to the handle, not to the pladour class. This value represents the percentage of black points in the 5×5 grid centred at (x_0, y_0). Therefore, the bigger *Perc2*, the more likely the robot is opposite a handle.

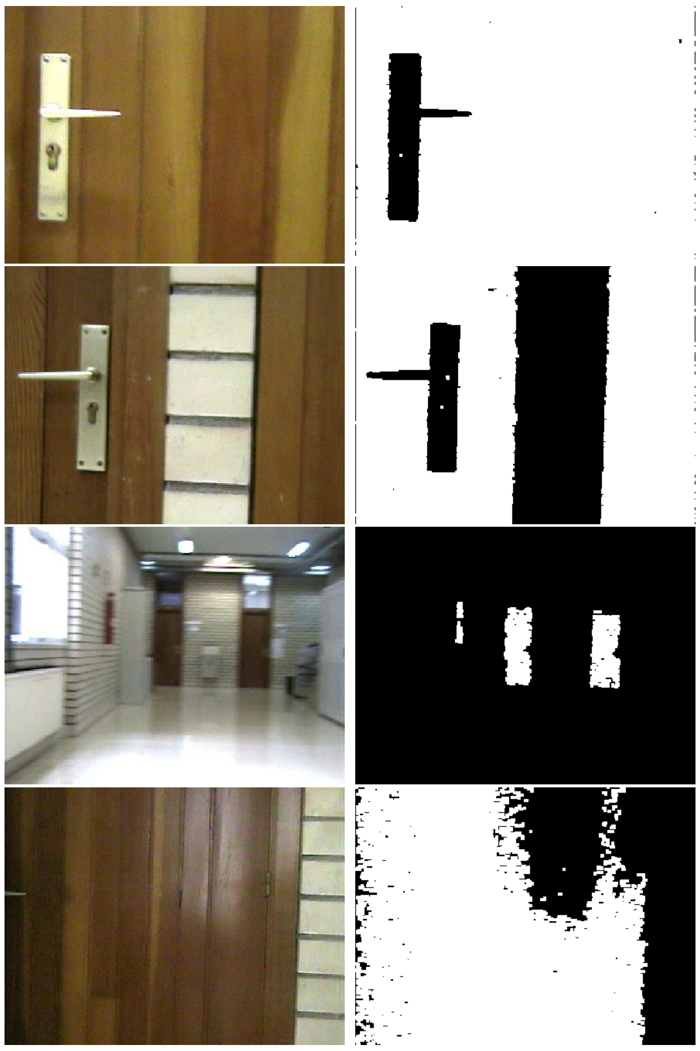


Figure 8.2: Segmentation of wooden surfaces

- *Perc3*: when the procedure that returns the region has really located a handle, the area that lies further from the centre is expected to be segmented white, they do not fall in the handle, but in the door blade. We define:
  - $S1$: set of points in the squared area centred at (x_0, y_0) and of size length $2 \times r$, where r is the radius of the detected circle.
  - $S2$: set of points in the squared area centred at (x_0, y_0) and of size length $2 \times (r + d)$, where d represents the desired shift from circle perimeter.

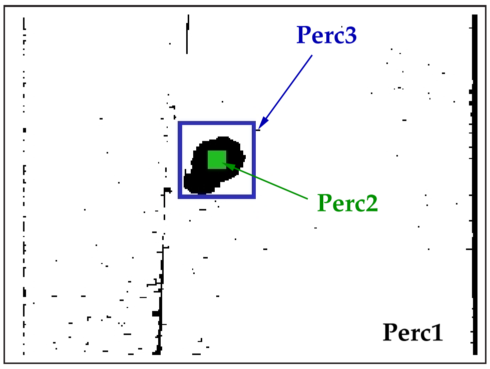


Figure 8.3: Zones from which *Perc2* and *Perc3* are computed; *Perc1* is computed over the whole image

Hence, $S = S_2 - S_1$ defines the handle's closest surroundings of the handle and *Perc3* represents the percentage of white points (*pladour*) in S. Again, the bigger *Perc3*, the more likely the robot is opposite handle.

The CHT gives the radius of each detected circle, together with some more information, whereas using region labelling r is defined as the maximum value between the width and the height of the obtained region.

The combination of these percentages (weighted mean) give us a measure of confidence that the robot is opposite a door and that the region recognition procedure has correctly identified a circular handle (cl_{handle}).

8.1.2 Results

At this stage of the experimentation, the handle recognition was performed for each class of handle using separate databases.

All the images (learning and testing DBs) were taken while the robot followed the corridors using its local navigation strategies, which allow it to navigate within a certain distance from the walls. Both databases contained almost equal positive and negative cases.

The databases characteristics were the following:

1. For circular handles, the database contained about 3000 entries.

2. For rectangular handles, the size was bigger with about 5000 images.

The experiments were executed for a maximum radius of 20 pixels ($r_{max} = 20$), a shift of 15 pixels from the circle perimeter ($d = 15$), and the confidence level of being in front of a handle should rise above 0.6 to confirm the door ($cl_{handle} > cl_{TH} = 0.6$).

In order to evaluate the improvement achieved by the three-stage process, the classification accuracy was computed for the sequence and compared with the accuracy that would be obtained if only region detection was used.

However, accuracy is considered a fairly crude score that does not give much information about the performance of a categoriser. The F1 measure, also known as F-score or F-measure, is a measure of a test's accuracy and can be viewed as a weighted average of the precision and recall (see Equation 8.1), where an F1 score reaches its best value at 1 and its worst at 0. F1 measure combines both precision and recall into a single metric and favour a balanced performance of the two metrics (Chen, 1996).

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{8.1}$$

Table 8.1 shows the obtained classification accuracies and the corresponding F1 measure values.

	Circular		Rectangular	
Method	Acc.	F1	Acc.	F1
CHT	30.6	0.319	–	–
CHT three-stage	85.1	0.685	–	–
Blobs	91.1	0.867	61.0	0.36
Blobs three-stage	96.7	0.946	93.6	0.72

Table 8.1: Experimental results

These results show how both performance measures, the classification accuracy and the F1 value are highly increased using the three stage approach for both types of handles. Surprisingly, the blob detector proves to be more efficient at identifying the circular handles. This is because from the robot's angle of vision, circles are distorted into ellipsoids so the CHT does not identify then as handles.

The door identification process should be performed in real time by the robot. Hence, the time needed to compute the identification must be analysed. Table 8.2 shows the time requirements of the developed three-stage algorithm.

Method	Circular	Rectangular
CHT three-stage	0.043	–
Blobs three-stage	0.050	0.057

Table 8.2: Computational payload (s) for processing an image applying the three-stage procedure

8.2 Feature extraction based identification

The approach described above proved to be too specific to the robot's particular environment and not easily generalisable. Although the region detector is robust enough, the segmentation processes are specific and new ones should be developed to detect handles located in different surfaces.

Feature extraction in images is an important issue in mobile robotics, as it helps the robot to understand its environment and fulfil its objectives. There are several applications where it is mandatory to recognise objects or scenes: image retrieval (Ledwich and Williams, 2004), mobile robot localisation (Gil et al., 2005; Tamimi et al., 2006), SLAM (Se et al., 2002), physical sign recognition (Roduner and Rohs, 2006) and automatic guidance of vehicles (Primdahl et al., 2005). Although different sensors can be used, given the rich information that can be extracted from images, vision is the most appropriate one. However, object identification becomes a complex task specially because of changeable environmental conditions, as well as changes in object scale and camera viewpoint.

Objects can be identified in images by extracting local image descriptors. These descriptors should be distinctive and invariant to image transformations. The idea is to detect image regions co-variant to a class of transformations which are then used as support regions to compute invariant descriptors (Mikolajczyk and Schmid, 2005). While complete invariance has yet to be achieved, features which are robustly resilient to most image transforms have been proposed by D. G. Lowe (2004).

Several invariant feature extraction techniques can be found in the literature, like SIFT and (U)SURF. These methods have applied in this research work.

8.2.1 Extracting features from door images

As mentioned previously, in our robot's environment there are circular and rectangular handles. Circular ones are on pladour blades. These have the advantage that almost every keypoint is located on the door handle and only a few appear also on the handle surroundings (see Figure 8.4(a)). On the other hand, rectangular handles are located on wooden door blades that are not textureless and therefore keypoints appear outside the handle (see Figure 8.4(b)).

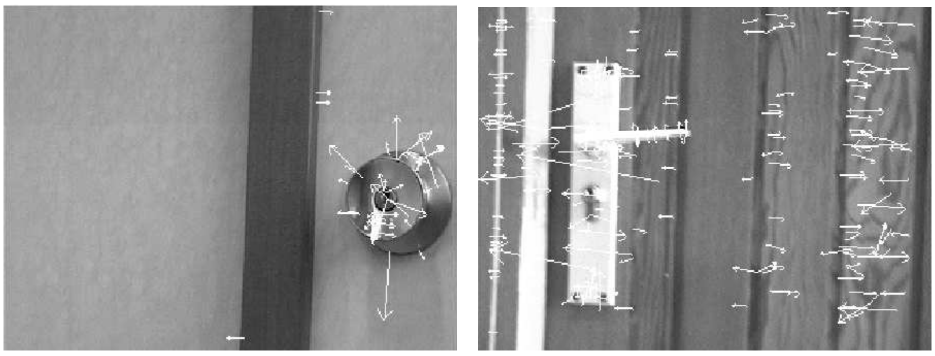

(a) Circular handle (b) Rectangular handle

Figure 8.4: SIFT keypoints in 320 × 240 images

When testing the adequateness of the feature extraction techniques for the handle identification task, the same databases were used as in the previous experiments, albeit a few extra images were taken to be used as reference objects (reference DBs). It must be noted that the test cases were collected in a different environment to the one where reference cases were taken. Therefore, the test databases did not contain images of the handles included in the reference database.

The databases characteristics were the following:

1. For circular handles, the testing database contained about 3000 entries and the reference database contained 32 images.

2. For rectangular handles, the testing database contained about 5000 images and the reference database contained 19 images.

The keypoint matching criteria used in these experiments is the 1-NN, the same proposed in (Lowe, 2004).

8.2.2 Results

Table 8.3 shows the obtained results. Again, accuracy and F1 measure are calculated. USURF outperforms SIFT in both, accuracy and F1 measure in the case of circular handles, but its performance degrades for rectangular ones.

	Circular		Rectangular	
	Acc.	F1	Acc.	F1
SIFT	62.39	0.592	**55.41**	**0.362**
SURF	72.5	0.691	42.34	0.324
USURF	**72.5**	**0.691**	47.0	0.344

Table 8.3: Performance of the keypoint extraction methods

These values are poor by comparison to the ones shown in table 8.1. Both the classification accuracy and the F1 measure are much lower that the values obtained using the three-stage procedure previously described.

8.3 Combining region extraction and features extraction

This section presents a new two-step algorithm that aims at reducing the superfluous keypoints obtained by methods like SIFT and (U)SURF by firstly extracting the region of the image where the object or objects to be identified are likely to be located.

Instead of computing the invariant features of the whole image, the size of the image to be processed is reduced by extracting the portion with a high probability of containing crucial features. Here on, we will call that portion the *Region Of Interest* (ROI) of the image.

Several methods can be used to extract the ROI but the experiments performed while developing the segmentation based door identifier showed that blob extraction yields robust results for handle identification.

Once the most relevant blob is located, its length and width values are used to find its centre and extract a square subimage. The size of the square is determined by the maximum value of either the length of the width of the candidate blob. The subimage is then scaled to a fixed size to obtain the portion of the image with a high probability of containing the object of interest, i.e. a ROI. Then, the keypoint extraction and matching procedure is performed on the extracted ROI. Figure 8.5 summarises the process.

Although it is not the case for the problem stated in this research work, the algorithm is drawn in a general form where more than one object can be searched in an image and hence, more than one ROI should be extracted and afterwards, processed for identification purposes.

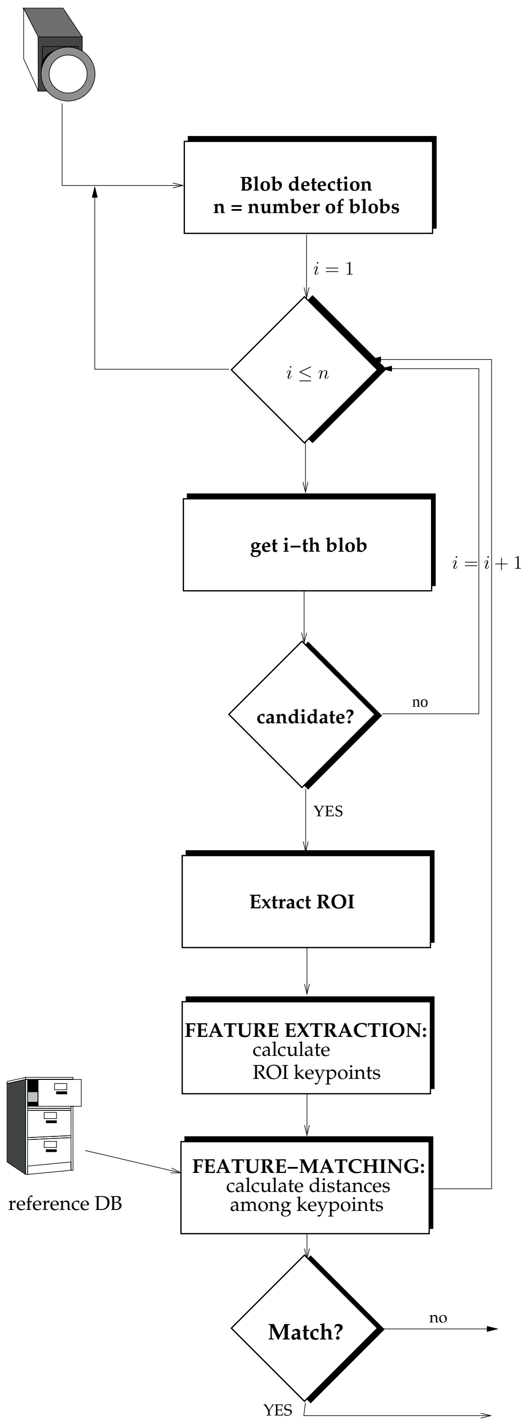
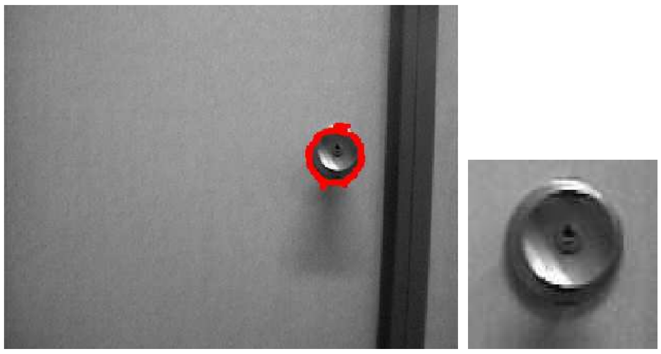


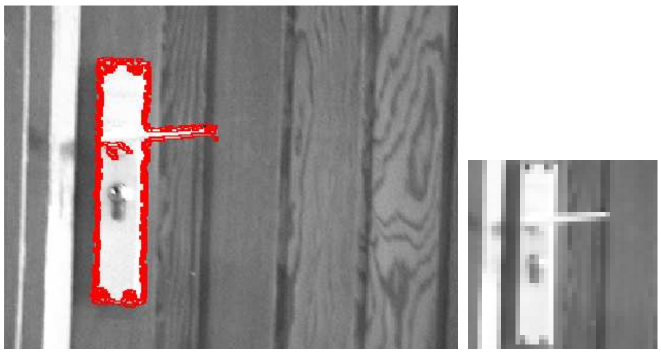
Figure 8.5: Flow diagram of the new algorithm

8.3.1 Application to handle identification: results

Figure 8.6 shows examples of the result of extracting the ROI for both handle types. Obviously, rectangular handle identification problem is hindered by the asymmetry of the handle itself and the textured blades where they are located.



(a) Circular handle: blob and ROI



(b) Rectangular handle: blob and ROI

Figure 8.6: Blob extraction and ROI scaling

Table 8.4-a) shows the improvement obtained applying the method proposed in the chapter. Again, USURF seems to be the most adequate feature extraction method for circular handles, whereas for the identification of non circular handles the best results are obtained by applying SIFT to the obtained ROI.

The accuracy is increased by almost a 30% for SIFT and about $20 - 22\%$ for SURF and USURF respectively for the case of circular handles. The improvement is more significant for rectangular handles, with an increase of 35% for SIFT and about 40% for SURF. The fact that this improvement is also reflected on the F1 measure proves the adequateness of the methodology.

As mentioned in (Pfeifer and Bongard, 2006), vision may take advantage of the physical interaction of the agent with its environment. Taking into account the robot's morphology and the environmental niche, more specifically, the height at which the camera is mounted on the robot and the height at which the handles are located on the doors, the handles should always appear at a specific height on the image. The improvement introduced by this *Morphological Restriction* (MR) is also showed in Table 8.4-b).

	Circular			Rectangular		
	acc.	size	F1	acc.	size	F1
SIFT	91.82	150	0.863	**91.52**	80	**0.669**
SURF	92.94	100	0.890	87.77	40	40.6
USURF	**94.35**	150	**0.911**	87.77	40	0.406

a) Region labelling + feature extraction

	Circular			Rectangular		
	acc.	size	F1	acc.	size	F1
SIFT	94.48	240	0.909	**92.67**	80	**0.699**
SURF	95.70	80	0.931	89.87	100	66.84
USURF	**96.09**	150	**0.936**	89.71	100	0.666

b) Region labelling + feature extraction + MR

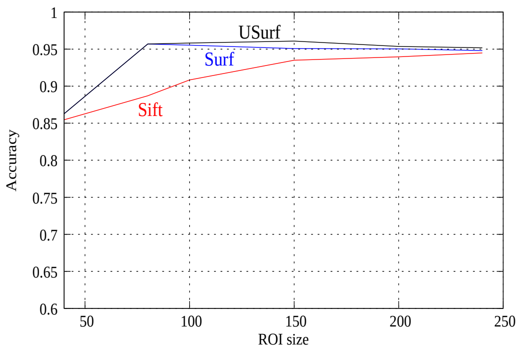
Table 8.4: Best results obtained by the new two-step algorithm

8.4 Discussion

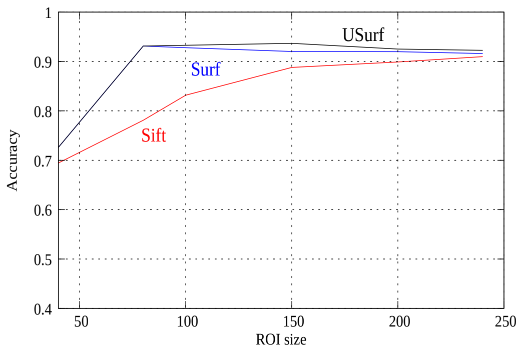
Figures 8.7 and 8.8 show the evolution of the performance, for circular and rectangular handles. Both, the classification accuracy and the F1 measure are plotted when the ROI size is increased. These figures correspond to the proposed two-step method together with the morphological restriction previously mentioned. It can be appreciated how USURF is the best approach for circular handles whereas SIFT improves speeded-up variants when rectangular handles need to be detected, regardless of the size of the ROI.

The results are similar to those obtained with the three-stage approach.

Table 8.5 reflects how the average number of keypoints increases according to the ROI size. These values were calculated using the reference databases used in each experimental trial and hence are only tentative values. Although the number of keypoints increases according to the ROI size, stable keypoints are also found on small images. A higher number of keypoints is obtained

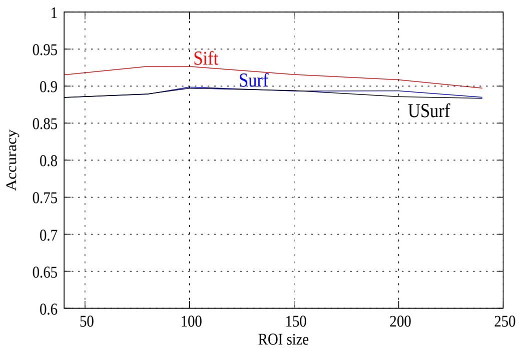


(a) Accuracy

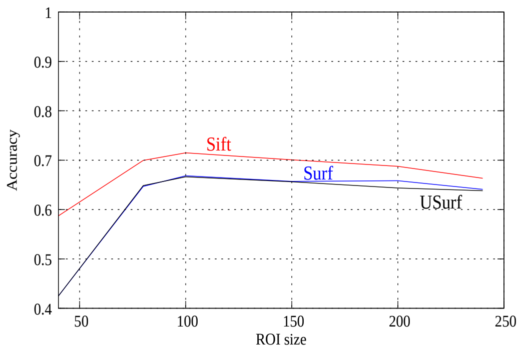


(b) F1

Figure 8.7: Performance on circular handles according to ROI variation



(a) Accuracy



(b) F1

Figure 8.8: Performance on rectangular handles according to ROI variation

when increasing the scale of the ROI, but the repeatability of these new keypoints is not satisfactory.

Ref. DB	40	80	100	150	200	320×240
Circular	10.09	21.43	41.5	74.03	105.53	49.28
Rectangular	5.95	23.47	38.79	72.10	97.78	139.89

Table 8.5: Average number of keypoints

Table 8.6 shows the average time needed to process a single image using SIFT (blob location, keypoint extraction and matching against the reference database). Note that the time needed is sensibly higher for larger ROI sizes due to the larger number of keypoints that occur in those images. Moreover, the computational requirements to process a single image are larger than the 20 fps attainable with the three-stage process; the same frequency can be reached using ROI size of 40.

Database	40	80	100	150	200	No ROI
Circular	0.06	0.18	0.22	0.46	0.75	0.31
Rectangular	0.06	0.13	0.17	0.42	0.58	0.89

Table 8.6: Computational payload (s) for processing an image using SIFT

To summarise, SIFT showed a degraded accuracy for ROI size 40 in the identification of circular handles. However, the better performance it showed for the identification of rectangular handles, together with the short computational time it needed confirms it as an appropriate method for the real time problem stated in this chapter.

8.5 Experiments with the robot

To evaluate the robustness of the handle identification system developed, it has been integrated in a behaviour-based control architecture that allows the robot Tartalo to travel across corridors without bumping into obstacles. When the robot finds a door, it stops, turns to face the door and knocks on it with its bumpers a couple of times asking for the door to be opened. If after a certain time the door is still closed, Tartalo turns back to face the corridor and carries on, looking for a new handle. However, if the door is opened the robot detects the opening with its laser and activates a door crossing behaviour module that allows it to enter the room. All the computation is carried out in its on-board Pentium (1.6 GHz).

The overall control architecture is composed of nine threads that communicate and activate/deactivate each other when appropriate (see Figure 8.9). Dashed lines in Figure 8.9 represent behaviour activation/deactivation links, while thin links are data communication connections among modules. Only the most relevant connections are drawn for sake of clarity.

These nine threads can be grouped as follows:

- Four device managers: these modules are not behaviour modules themselves, but they are needed to get/send up-to-date data from/to physical devices. They are needed for laser, bumper, speech and motor control.

- Three local navigation strategies: *corridor following*, *door knocking* and *door crossing*.

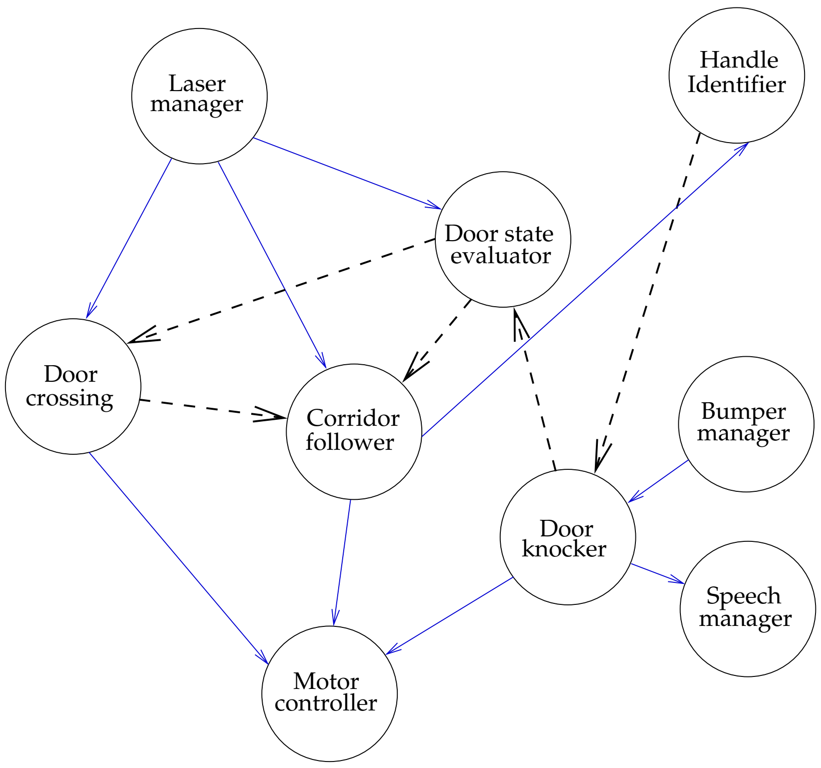


Figure 8.9: Behaviour modules and communication

– The *corridor following* behaviour represents the main local navigation strategy and its aim is to balance free space on both sides of the robot, so that corridors are followed in smooth trajectories while avoiding obstacles when necessary.

– The *door knocking* behaviour performs the sequence of movements that are needed to knock a door once a closed door is identified.

– The *door crossing* module is responsible for guiding the robot for a short period of time when an open door is recognised.

- Two landmark identification subsystems: *handle identification* and *door state evaluator* subsystems. The *handle identification* subsystem includes any of the door identification approaches developed and explained earlier, and the door state evaluator calculates the difference between the laser reading taken right after the robot stops to wait for the door to open and the current laser reading, and decides if the door has been opened according to the difference between the two measures.

As mentioned previously, experiments within the real robot/environment system were performed using a ROI size of 40 and applying the SIFT feature extraction method. Also, to make its behaviour more robust, instead of relying on a single image classification, the robot based its decisions upon the sum of the descriptor matchings accumulated in the last five consecutive images. Experiments were carried out in three different environments.

Environment 1: circular handles

Figure 8.10 shows the environment together with the evolution of the sum of the matched keypoints over time. The horizontal line represents the value at which the threshold was fixed. The 18 doors present in the environment were properly identified and no false positives occurred.

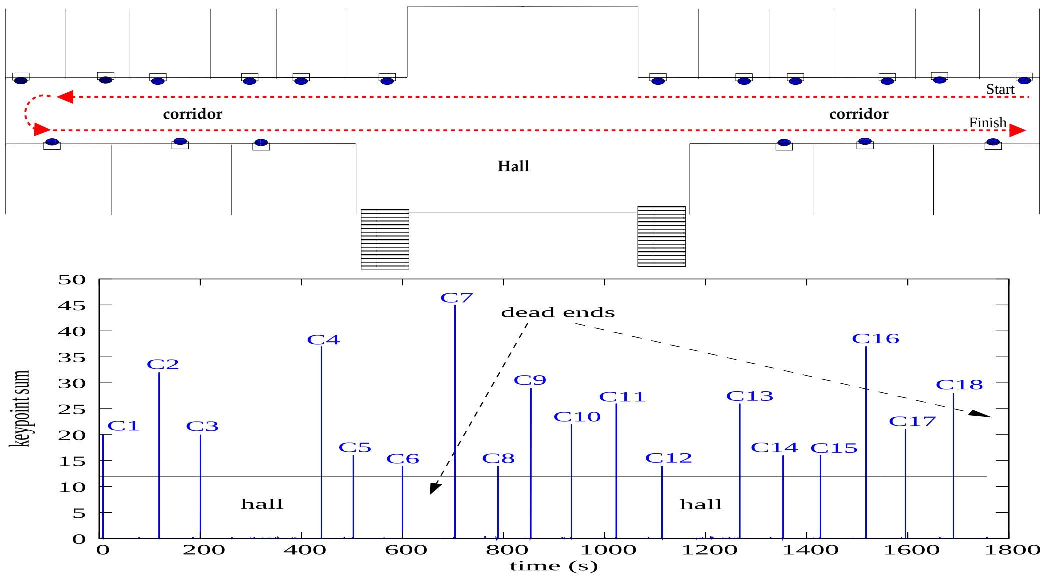


Figure 8.10: First floor results

Environment 2: rectangular handles

A second experiment was performed at the lower corridor of the second floor of the building where 39 rectangular handles were to be identified. Figure 8.11 shows the original environment and the evolution of the keypoint sum over time.

The robot started on the left side of the corridor, with its camera pointing to its right and travelled all the way along the corridor successfully fulfilling the sequence of 6+7+7 handle identification, and then turned at the dead-end. In its way back, only one of the handles of the central sector of the

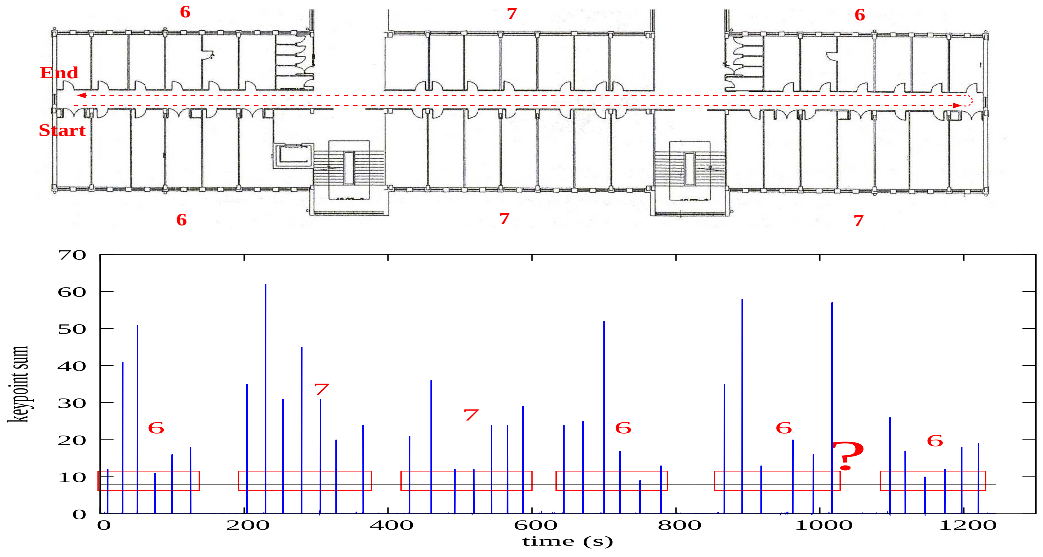


Figure 8.11: Second floor results

corridor, the one marked with a circle in the upper image of Figure 8.11, was not recognised (6+6+6) and again, no false positives were given. Hence, a success ratio of 0.97 was achieved.

Environment 3: mixed identification

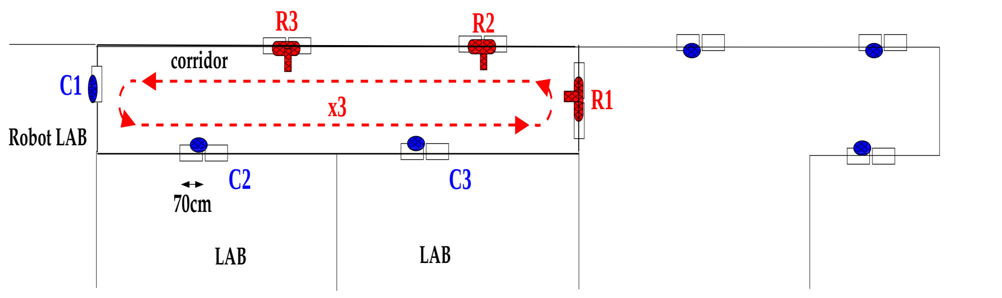
A last experiment was made on a part of the third floor of the Faculty of Computer Science (Figure 8.12), an environment that contained both types of handles. Three circular handles and three rectangular handles were to be identified. The robot was left running for three rounds in the corridor.

In each round, three circular handles and three rectangular ones were positively identified and no false positive was given by the system. The total amount of handles correctly recognised was 18.

Summing up, during the experiments performed within the real robot/environment system, 74 handles were identified, which meant an achieved success of 98.66% and not a false positive occurred.

8.6 Generality of the method I: application to new handles

As mentioned in Chapter 6, while developing this research work the Faculty of Computer Science was completely refurbished. The inside structure was



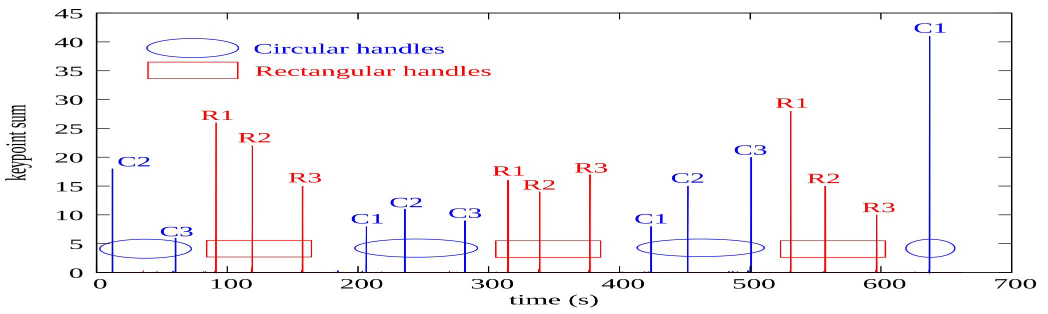


Figure 8.12: Mixed environment

modified, walls were either painted or substituted by prefabricated panels. As a consequence, old doors were removed and at present doors and handles were standardised as shown in Figure 8.13.

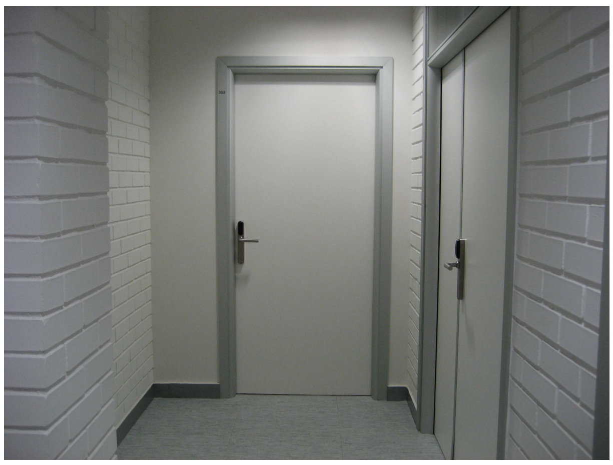
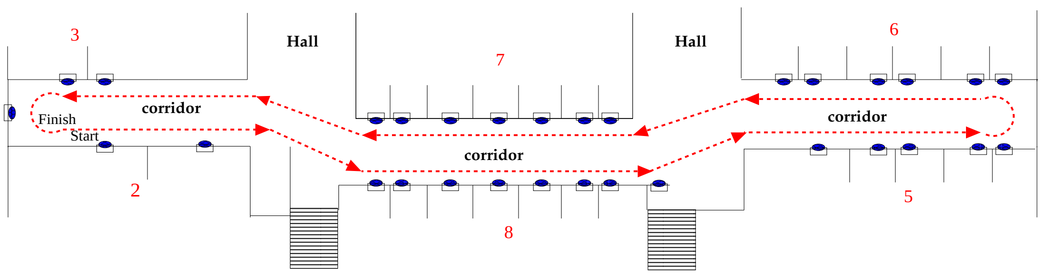
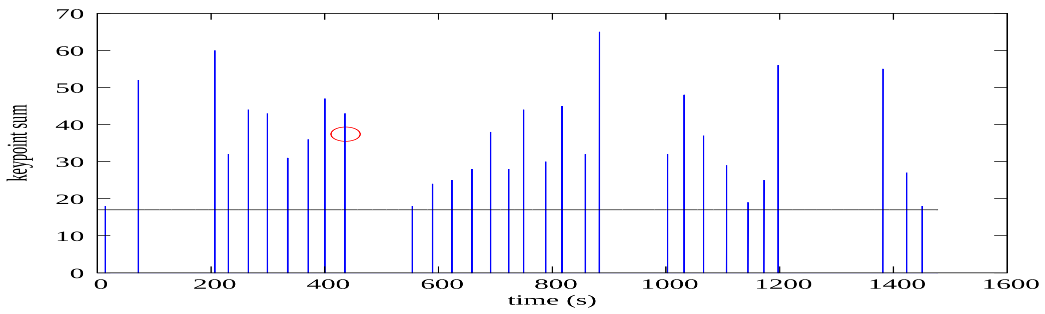


Figure 8.13: New doors at the Faculty of Compuer Science

This refurbishment gave us the opportunity to test the level of generalisation of the developed algorithm and to confirm the robust performance of the approach. To that purpose, some pictures of the new handles were taken and stored to be used as the reference database. Nothing was changed in the algorithm and the parameters used for rectangular handles were inherited for the blob extraction process. The new experiment was run at the front wing of the third floor. This wing contains 30 doors plus an additional one that closes one of the corridors which is normally opened. Figure 8.14 shows this wing of the environment and the evolution of the keypoint sum over time. To replicate previous procedures, experiments within the real robot/environment system were performed using a ROI size of 40 and applying the SIFT feature extraction method. Once more, to make the behaviour more robust, instead of relying on a single image classification, the robot based its decisions upon the sum of the descriptor matches accumulated in the last five consecutive images.



(a) New environment



(b) Keypoint sum over time

Figure 8.14: New doors identification results

The robot started on the left side of the corridor, with its camera pointing to its right and travelled all the way along the corridor successfully identifying all the handles, and then turned at the dead-end. The handle marked with a circle in the plot in Figure 8.14(b) represents the door that closes the corridor. Again, no false positives occurred.

It is worth pointing out that the door identification demo was performed during the open day held at the Faculty of Computer Science (see Figure 8.15). A short video can be watched at rstp://streaming.i2basque.es/i2basque/Informatica2011/ Infor2011peli2.mov (timestamp 1.55–2.15) .

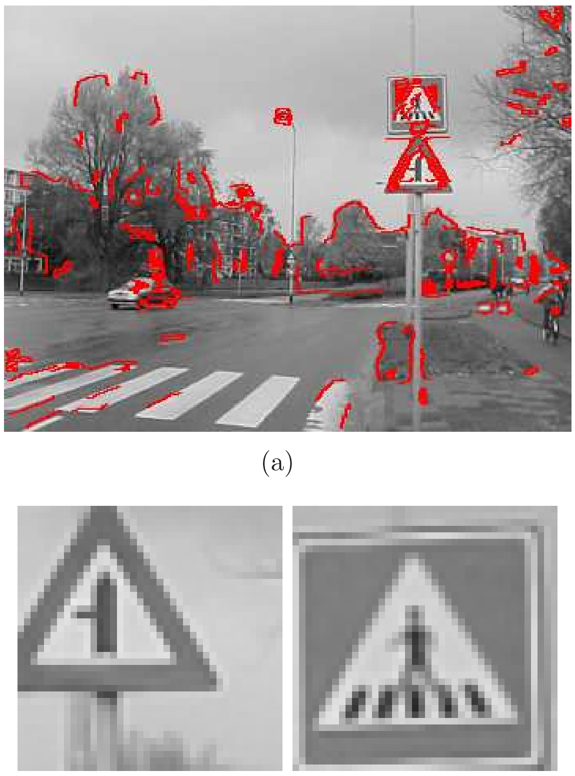
Figure 8.15: Door identification demo's picture

8.7 Generality of the method II: traffic signal identification

In order to test the potential of the proposed approach, it was applied to the road signal identification problem using the database used in (Grigorescu and Petkov, 2003). This database[1] contains 360×270 sized 48 images and three signals are to be recognised: pedestrian, bicycle and crossing. As in our case, the problem of signal recognition is a problem of different object recognition. The images showed very cluttered views and contained more than one road signal. Thus, several ROIs were extracted and processed in each image. Again, the blob properties were restricted in order to reduce the blobs to be considered and we used a reference database containing only three images, one per signal. Figure 8.16 shows an example of the detected blobs and the subimages extracted from them.

To calculate the classification accuracy we assumed that an image could contain at most one signal of each type, i.e. there could not be more than

[1]http://www.cs.rug.nl/~imaging/databases/traffic_sign_database/traffic_sign_database.html



(a)

(b)

Figure 8.16: Detected blobs and extracted signals

three signals per image. In this way, the accuracy was computed summing up the true positive and true negative cases.

Figure 8.17 shows the obtained results. Both, accuracy and F1 measure are shown in order to compare the strengths/weaknesses of the approaches. For ROI sizes larger that 150 the number of false positives increased considerably when applying SIFT. The best results were obtained for small ROI sizes because the signal size in the images approximated that geometry. In spite of the difficulty of the task, the accuracy raised up to 95% for ROI size of 40×40. For larger ROI sizes, the scaling of the subimage seemed to drastically affect the stability of the keypoints with the consequent degradation on the classification performance.

However, both SURF and USURF need larger ROI sizes compared to SIFT to achieve their best classification performances. Small ROI sizes were not appropriate for the speeded up variants because the number of keypoints extracted was very small, as can be seen in Table 8.7.

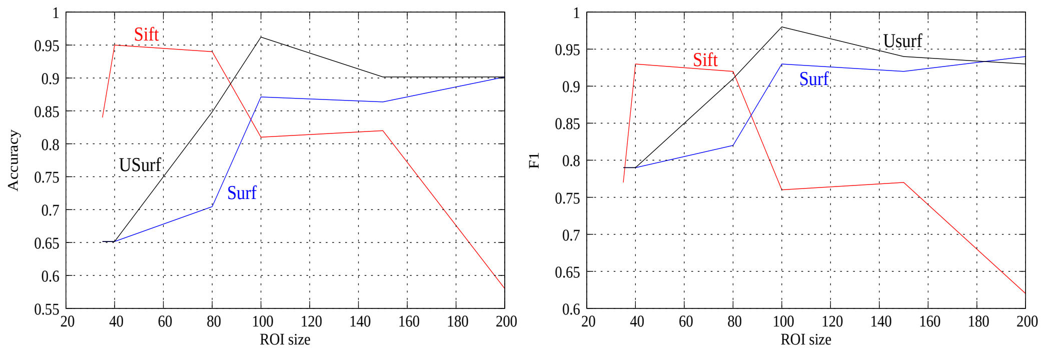


Figure 8.17: Results depending on ROI size

	30	40	80	100	150	200
SIFT	0.66	18.66	49.66	72	119.33	159
(U)SURF	0.33	1.66	14	24	48.33	59.66

Table 8.7: Average number of keypoints in the images of the signals reference database

8.8 Conclusions

This chapter summarises several approaches aimed at identifying doors for robot navigation purposes. While environmental specific methods showed a good performance, the segmentation based approach is not easily applicable to other environments since new segmenters need to be developed. Feature extraction methods were easy to apply since they only required building a reference database containing the objects to be identified. However, the classification performance and the computational payload needed to process a single image made these methods unsuitable for robot navigation. A new two step algorithm was presented based on feature extraction. The objective of adding a region detection step before extracting the features is to obtain the most relevant part of the image, i.e. the subimage with a high probability of containing the most representative region of the image, and to limit the application of the feature extraction techniques to that region. The developed algorithm improves the relevance of the extracted features, reducing the superfluous keypoints to be compared as well as increasing the efficiency of the process by improving accuracy and reducing the computational time.

ROI extraction improved the handle identification procedure and depending on the ROI size, the computational time to classify an image was considerably reduced. The system showed a very low tendency to give false positives while providing a robust identification. The author's opinion is that the presented approach is appropriate for situations where the background is so changeable that its characteristics are either irrelevant for object identification or just add noise to the recognition process.

Part IV

Behaviour-based localisation and mapping

Chapter 9

Behaviour-based probabilistic localisation

Contents

As it was noted earlier, an important drawback of the behaviour-based (BB) navigation system deployed in Galtxagorri resides in the deterministic nature of the localisation system. The initial position must be known and the spread of activation mechanism allows the tracking of the robot's position. But, if for any reason the successor's signature is not identified, the robot gets lost and it has no means of relocalising itself.

One way to keep localisation is through the use of odometry; with this sensor it is possible to calculate the trajectory of the robot by calculating the movement of its wheels. However, the farther the robot goes and the more turns it takes, the more inaccurate odometry becomes, which produces an accumulative error over time (see Section 3.2.5).

The alternative to a navigation system that is fundamentally based on a global Cartesian reference frame within which the robot integrates its own motion through odometry would be to use a system in which the robot navigates using landmarks (Nehmzow, 1999). Galtxagorri's localisation system relayed exclusively on landmark identification.

But motion information is still needed to cope with the perceptual aliasing problem. In general, no totally position independent method can distinguish between two landmarks of the same qualitative type and compass bearing. It is common to use odometry measurements to calculate the robot's motion over time, despite the fact that it is only available once the robot has moved and that suffers from accumulative error.

To tackle these drawbacks, the field of probabilistic robotics offers nowadays a full set of algorithms that provide the necessary tools to maintain belief distributions over the state space (Thrun et al., 2005). By contrast, robots developed within the BB paradigm only make use of a subset of environmental properties needed for localisation, and shows a higher degree of adaptability to dynamic environmental changes.

Given the simplicity, low computational cost and the efficient functioning of BB systems this chapter pretends to demonstrate that it is possible to benefit from the soundness and mathematical foundation of probabilistic techniques within the field of BB robotics to improve in the navigation process.

9.1 Probabilistic navigation

In mobile robotics several methods have been developed for indoor navigation (Latombe, 1991). As mentioned in Chapter 5, hybrid architectures tackle the problem of navigation in three steps: mapping, localisation and planning. These are old problems from the perspective of manipulation robotics and are nowadays treated in a probabilistic manner. Hence the name of the field *probabilistic robotics* (Thrun et al., 2005). The field of probabilistic robotics makes explicit the uncertainty in sensor measurements and robot motion by using probabilistic methods. Let us summarise the three aspects involved in the navigation task.

Mapping: the main approaches for environment representation are grid-based approaches and topological maps and both, grid-based and topological representations exhibit orthogonal strengths and weaknesses. Grid-based maps are considerably easier to learn, because they facilitate accurate localisation, and are easy to maintain. On the other hand, being more compact, topological maps facilitate fast planning. In statistical terms, the problem of mapping is the problem of finding the most likely map given the data available. A key feature of the statistical approach is that it can equally be applied to both topological and metric maps. But hybrid representations are also possible. For instance, in (Thrun et al., 1998b) a robot seeks to find the most likely map from a set of observations and motion commands. Howard and Kitchen (2001) present a vision-guided robot navigating around the corridors at a university building. Two coordinate systems are used; the model is defined in a global coordinate system (GCS) and the odometry measurements are made in a local one (LCS), with and arbitrary origin. The localisation process consists of a search for an invertible transformation function between the GCS and the LCS. These estimation problems are solved by a variant of the *Expectation Maximisation* algorithm first constructing a coarse grained topological map, and then building a detailed metric map based on the obtained coarse representation. Paz et al. (2008) propose an algorithm to reduce the computational complexity of the EKF-SLAM named *Divide and Conquer SLAM* in which lower level maps are computed with the EKF-SLAM and then, these maps are further joined in a hierarchical fashion.

A classic approach for the generation of maps is based on *Kalman filters* (Kalman, 1960). Kalman filter-based mapping algorithms are often referred to as SLAM algorithms. SLAM (Simultaneous Localisation and Mapping) or CML (Concurrent Mapping and Localisation) is not a solution in itself, but a problem concerned with building the map while jointly computing the robot's localisation (Thrun, 2003). The coupling of these two tasks should relieve the *correspondence problem* (Leonard and Durrant-Whyte, 1992). That is to say, the recognition of already visited places, a problem also known as *loop-closing*, which is hard to solve when mapping and localisation are tackled separately. The loop-closing is performed in two main steps: an exploration phase to reach different places, and location revisiting for consistency that can also drive the robot to new unvisited locations (Yamauchi et al., 1998; González-Baños and Latombe, 2002).

During the mapping process different exploration strategies can be used. Wall following is probably the simplest strategy but it does not guarantee that the whole environment will be explored. On the other hand, coastal navigation (Roy and Thrun, 1999) keeps the robot close to environment properties in order to extract sufficient features to obtain the map. Whenever multiple choices are available, the option to go to the least explored region seems the most attractive (Lee, 1996). In (Yamauchi et al., 1998) frontier-based exploration is used with the purpose of building the environmental map with continuous localisation. A different approach is used when multiple robots have to cooperate to obtain the map of the environment. The inherent difficulty of the mapping process is increased in the sense that interferences among robots must be avoided. Most multi-robot mapping techniques proceed in two steps: first, potential target locations or areas are selected, usually located close to unknown areas so that the robots can observe the unknown places. Then, robots are assigned to the selected locations and as they approach them, they include in the map the information obtained in the surroundings. The key for an efficient mapping process is then to properly selecting the robots' potential target locations. Stachniss et al. (2008) proposed a Hidden Markov Models (HMM) based technique that takes into account spacial dependencies between nearby locations to obtain a lower error rate for places located at frontiers, and the robots get a higher reward for exploring corridors since they typically provide more branchings to unexplored areas.

Localisation: generally speaking, self-localisation consists on determining the robot's position within the model starting from an initial unknown state. There are many methods for localisation, but there is no general solution to deal with it easily (Thrun, 2002). Among the different approaches that try to solve the localisation task, the *Markov localisation*, a method based on an extension of HMMs, maintains a belief over the robot global configuration space. Markov localisation is a passive probabilistic process that calculates the probability of the location of the robot at each state as a function of the acquired sensor model and the kinematic model (Cassandra et al., 1996; Fox et al., 1998; Nourbakhsh, 1998). Markov localisation is further explained in Section 9.3.

Since the Markov localisation is usually applied in grid maps of high resolution, the requirements it imposes on updating and maintaining the probability density function over the whole set of states are very demanding.

Alternatively, some authors propose localisation methods based on particle filters, where the *a posteriori* belief is represented by a set of particles together with an associated weighting factor of each particle, which constitute a discrete subset of the probability distribution (Rekleitis, 2004; Fox et al., 2000; Thrun et al., 2001).

The best known of these methods is the *Monte Carlo localisation* (Fox et al., 1999a). In it, the a posteriori belief is represented by a set of samples or particles with an associated weight called *importance factor*. To update the samples, firstly the kinematic model is applied to a randomly selected sample according to its weight, and then, the sensory model updates the weight of each sample according to the probability of obtaining the perception observed in the position represented by the sample. The main advantage of the Monte Carlo localisation is that it reduces the computational cost of the localisation process as well as the memory requirements by concentrating the computational effort only on points of the space that are interesting at each moment, instead of on the whole. To obtain the best results with this method, it is essential to select an appropriate number of samples (Thrun et al., 2001) since a limited diversity in the set of particles selected may lead to the incapacity of the robot to relocate itself if it gets lost.

The most important limitation of probabilistic localisation methods is that it assumes an unchangeable perception in any given state. This is a strong assumption for dynamic environments, where there may be unexpected obstacles such as wastepaper baskets, small furniture or people walking. To overcome this restriction, Fox et al. (1999b) try to modify the perception model using two filters. The first one, called *entropy filter*, measures the relative change of entropy upon incorporating a sensor reading into the position belief, and the second one, the *distance filter*, selects the readings according to how much shorter they are than the expected value. In the same vain Yamauchi and Beer (1997) try to include techniques in the architecture that sense changes in the environment and adapt the environment representation accordingly.

Planning: planning is a traditional field of Artificial Intelligence (AI) and many algorithms and techniques have been developed to cope with different problems, ranging from mobile robotics and manipulators up to graphics animation or non-invasive surgery (Latombe, 1999; Khatib, 1996; Latombe, 1991; Thorpe, 1984). Given a map and a goal location, path planning involves identifying a trajectory that will cause the robot to reach the goal location when executed (Siegwart and Nourbakhsh, 2004). Typically, there are many possible paths between the robot location and its goal, but the major task on planning is to look for the optimal way based on some criterion like the shortest path or safety.

Generally, global planners should be combined with some obstacle avoidance strategy or local planner such as the Vector Field Histogram (VFH) (Ulrich and Borenstein, 2000) or the Nearness Diagram (Minguez and Montano, 2004) in order to cope with dynamic environments and reduce the need and the computational payload of replanning when changes in the environment make the original path unfeasible. To this aim the local path between two consecutive waypoints is completed by local planners in a reactive way (see (Siegwart and Nourbakhsh, 2004) for a comparison among different local planners). This solution has the advantage of introducing the sensory information within the control cycle.

Three are the basic approaches to planning. *Roadmap based techniques*, such as visibility graphs and Voronoi diagrams, try to identify sets of routes in the configuration of free space of the environment. On the other hand, *cell decomposition based approaches* decompose the environment in regions named cells. These cells can be of variable or fixed size. After determining the free cells and the adjacency of the cells, cell decomposition searches for routes that link the starting and the goal cells.

Whilst planning using the topological map is more efficient than planning with the grid-based map, plans generated with topological maps are usually longer that plans generated with grid-based maps. Hence, to make a computationally efficient search using graph theory algorithms such as A* , Dikjstra's shortest path algorithm or *Value Iteration*, the map is usually turned into a graph, a set of nodes and arcs that connect them (Matarić, 2009). For instance, in (Owen and Nehmzow, 1998), *best-first* search is applied to find the shortest path on the topological model acquired by a self-organising map.

The third approach to planning are *potential field based methods*. These methods are based on the idea that the robot is a particle that moves in a potential field. Obstacles generate repulsive forces whereas the free space and the target position generate attractive forces. Within that configuration the robot reaches the goal by moving on the negative of the gradient. For instance, in (Kortenkamp et al., 1998) a potential field proportional to the distance from the start position to the goal is computed and a gradient descent is performed, selecting via points whenever the slope of the path changes. And the *Fast Marching Square method* (FM^2) (Garrido et al., 2009) is a motion planner based on the Fast Marching method, a particular case of the Level Set methods proposed by Sethian (1997). Level Set methods provide a way for representing a front propagating with speed F. The Fast Marching method is an efficient numerical algorithm to solve the front propagation when the sign of the speed function F does not change (that is, when the front moves only forwards or backwards).

Traditional planning algorithms do not consider the uncertainty associated to robot actions. The state transition function is considered deterministic, so an action performed in a given state brings the robot to a known state without uncertainty; the state transition function is considered deterministic. By contrast, the field of probabilistic robotics does take into account uncertainty when deciding what the robot can do. Instead of static plans, the aim is to generate action selection policies or universal plans. This generation can be approached from different viewpoints and with a higher or lower degree of difficulty.

A popular formalism for decision making under uncertainty is the *Markov Decision Process* (MDP) framework. In this paradigm, an agent interacts with a given system by executing actions that change the state of the system stochastically and that provide rewards or penalties to the agent. The environment is considered observable and uncertainty is only associated only to the execution of actions. Thrun et al. (1998a) combine metric and topological representation and they use a variation of value iteration dynamic programming algorithm to select the best action to be taken in the exploration of the environment during the mapping process. However, if uncertainty is linked to both state perception and actions, then the planning task is formalised as *Partially Observable Markov Decision Process* (POMDP) (Thrun et al., 2005).

9.2 Related work

We found some references that bear resemblance with our approach. In the work by Barber and Salichs (2001) an event driven navigation method is presented. The topological map (*Navigation Chart*) is not made up of environmental element successions, but of a succession of tasks to be executed during navigation, which means going through all the possible plans. Similarly, Egido et al. (2004) propose to represent the environment by a succession of elementary skills. Concerning the metric encoding, Lankenau and Röfer (2002) present a self-localisation system where maps are represented as route graphs that contain (geo-)metric data about the length of the corridors as well as about the included angles.

Related to the action model, in the work by Tomatis et al. (2001) a combination of topological and metric approaches for SLAM is presented where a global topological map which contains a metric map associated to each of the nodes is used. The topological localisation system is also Markovian but within the action model, they condition the probability that a state transition occurs to the observation of an opening. If no opening is visible, then the probability of remaining in a node is high while transitions to other states are almost improbable. On the other hand, when an opening is visible, the most probable transition is the closest one according to the distance travelled while the rest of the transitions are less probable.

Finally, in (Zanichelli, 1999) a landmark-based topological representation is used in a behaviour-based navigation system. The topological map is a connected graph where each node represents a landmark completely characterised by a set of attributes. It is mentioned that state transition probabilities are estimated from the reachability information contained in the topological map, but no detailed explanation is given about the process. Moreover, navigation is solely based on wall following.

9.3 Markov localisation

Probabilistic methods offer the necessary tools with a sound theoretical basis for handling self localisation but they are generally applied to rigid environment representations and therefore they are hardly capable of coping with dynamic environments. As mentioned earlier, the key idea of probabilistic robotics is to explicitly represent the uncertainty associated to robot localisation by means of probability calculus theory, thus representing the ambiguity in a sound manner.

Generally speaking, probabilistic modelling can be performed using different formalisms (see Figure 9.1). Each technique is based on different assumptions about the action and sensor models and also about the initial state. The resulting algorithms have then different computational overheads (Diard et al., 2003).

When addressing the robot localisation problem, a common technique is the Markov Localisation (ML) algorithm which is an adaptation of the Bayes filter for the robot localisation problem and, as such it is a recursive algorithm. This is a probabilistic process that maintain a density function at all time steps, a belief on the global configuration space of the robot. It is based on *Markov assumption* or *the static world assumption*, which is to assume that the current robot's location is the only factor that systematically affects the sensor readings. Formally, if $L_t = l$ represents the location of the robot at time t is l and z_i is the sensory input at the instant of time i, then:

$$P(z_{t+1}, z_{t+2}....|L_t = x, z_0, z_1, \cdots, z_t) = P(z_{t+1}, z_{t+2}....|L_t = x) \tag{9.1}$$

So, with the Markov localisation it is possible to calculate, at each instant of time t, the probability that the robot is in every possible location l given a

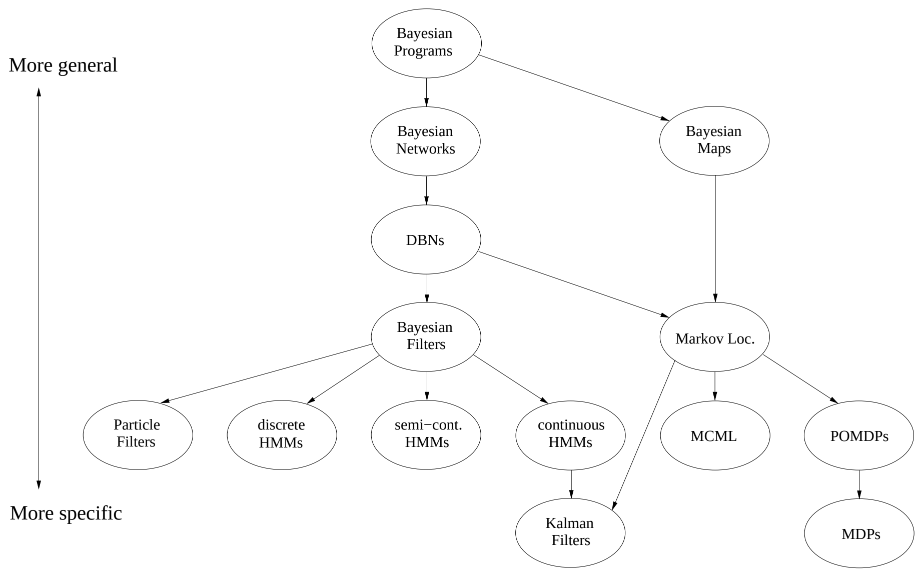


Figure 9.1: Probabilistic models ordered from general to specific

sequence of sensor readings $d = z_0, z_1, \cdots, z_{t-1}$, i.e. $P(X_t = x|d)$. Therefore, according to the Equation 9.2, the probability of being in state x at a given time depends on the probability of receiving the perception z_t in that state – sensory model – and also on the probability of reaching that state executing action u at any previous state – kinematics or movement model.

$$Bel(x) = P(z_t|x) \sum_{x'} P(x|u, x')Bel(x') \tag{9.2}$$

The ML algorithm (see Algorithm 9.1) addresses the global localisation problem, the position tracking problem and the kidnapped robot problem, although it is normally applied in static environments.

Algorithm 9.1 Discrete form of the ML algorithm

```
1: Markov_localisation($bel(x_{k-1}), u_k, z_k, m$)
2: for all $x_i$ do
3:     $\overline{bel(x_i)} = \sum_h p(x_i|u_k, x_h, m)bel(x_h)$
4:     $bel(x_i) = \mu p(z_k|x_i, m)\overline{bel(x_i)}$
5: end for
6: return($bel(x_t)$)
```

Note that step 3 implies a loop over the whole set of states.

9.4 Probabilistic behaviour activation

In order to apply the Markov Localisation (ML) algorithm, two probability distributions must be defined: the probability of observing a sensor reading z_t at each state (*sensor model*, $\forall i, P(z_t|x_i)$) and the state transition probability or *action model*, $\forall i, j, P(x_i|x_j, u_t)$, being u_t the action executed by the robot at time t. Then, the belief of being at a certain location i is calculated in two steps (see Algorithm 9.1):

1. The probability of having reached location x_i is approximated by summing up the probabilities of arriving to that particular node from any location x_j as a consequence of action u_t multiplied by the belief of being in x_j at the previous step.

$$\overline{Bel}(x_i) = \sum_j p(x_i|x_j, u_t)Bel(x_j)$$

2. The previous value is multiplied by the probability of perceiving z_t at location x_i. Once this process is repeated for every possible robot location x_i, the beliefs are normalised.

   This is usually implemented in two steps:

   - For all i,
   $$\hat{Bel}(x_i) = p(z_k|x_i, m)\overline{Bel}(x_i)$$
   and
   $$\mu = \frac{1}{\sum_k \hat{Bel}(x_k)}$$
   - For all i,
   $$Bel(x_i) = \mu\hat{Bel}(x_i)$$

Note that the action model implicitly defines the full connectivity among the nodes; in principle, all transitions are possible and hence there is no need for an explicit definition of the connectivity graph.

The next subsections explain how the ML algorithm is integrated in the procedural map description previously described.

9.4.1 Sensor model

The control architecture includes several landmark identification subsystems that output a confidence level for each type of landmark. These values are filtered through node signatures that give at each time step the node activation level according to the sensor readings.

Then the probability of seeing a concrete landmark k (cl_k^t) in a node x_i is a function of the expected values (signature of the node) and the confidence level values returned by the landmark identification subsystems:

$$P(cl_k^t|x_i) = f(cl_k^t, signature_i) \in (0, \cdots, 1]$$

And assuming independence among the different landmarks:

$$P(z_t|x_i) = \prod_{h=1}^{k_i} P(cl_h^t|x_i)$$

9.4.2 State transition probability

In general, no totally position independent method is able to distinguish between two landmarks of the same qualitative type and compass bearing.

The state transition probability function should reflect the probability that the transition occurs if the robot has travelled the distance (translational and rotational) accumulated by the odometry system.

Several motion models exist in the literature (see (Thrun et al., 2005)). The one selected as a starting point for this research was the *odometry motion model* for its simplicity. In it, any robot movement is decomposed in three steps: initial rotation (δ_{rot1}), translation (δ_{trans}) and final rotation (δ_{rot2}) (see Figure 9.2). It is assumed that each motion variable is perturbed by an independent error source.

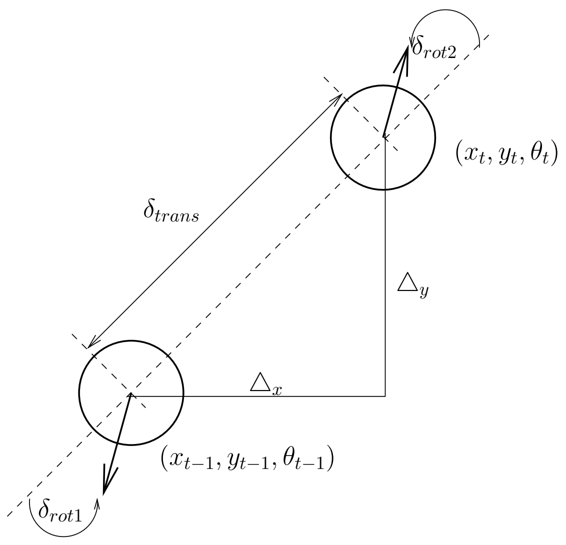


Figure 9.2: Odometry motion model

If according to two consecutive odometry measurements the robot's movement vector is $(\overline{\delta_{rot1}}, \overline{\delta_{trans}}, \overline{\delta_{rot2}})$ and the movement vector associated to the transition from state x_j to state x_i is $(\delta_{rot1}, \delta_{trans}, \delta_{rot2})$, the state transition probability is then the product of the component probability functions. Each component function is described as a zero mean normal distribution based on the difference between the occurred value and the theoretical value:

$$\left.\begin{array}{l} P_{rot1} = N(\delta_{trans} - \overline{\delta_{trans}}, \sigma_{rot1}) \\ P_{trans} = N(\delta_{trans} - \overline{\delta_{trans}}, \sigma_{trans}) \\ P_{rot2} = N(\delta_{rot2} - \overline{\delta_{rot2}}, \sigma_{rot2}) \end{array}\right\} \Rightarrow P(x_i|x_j, u_t) = p_{rot1} \times p_{trans} \times p_{rot2}$$

9.4.3 Topological places

Within the behaviour-based approach, topological maps should be composed of tightly coupled behaviours specific to meaningful locations. The overall "map" is then composed of sets of behaviours, each launched on a different thread, with an associated functionality and differing in the node signature that makes them distinguishable.

To apply the Markov localisation and for global localisation, the definition of the map has been modified. Predecessor and successor links have been removed and the location identifier has been modified to contain position information obtained from odometry.

Formally, a map is a set of nodes, each one consisting of:

1. A set of inputs (from landmark identification subsystems) and outputs. These outputs should serve to reduce the distance between the current state and the goal.

2. A signature that uniquely identifies the node from its immediate predecessors and successors: $signature_i$. Locations are recognised by identifying what makes each place singular, and therefore the signature reflects the state of a set of specific landmarks.

3. A function α_i to be executed when the node is active and that will output the action to be performed at the node specific current state. The behaviour the robot will show may vary depending on the location, and so can the associated function of several nodes.

4. By contrast with grid models, the topological representation approach does not imply a uniform distribution of locations. Behaviours are maintained for different space scales, depending on the environment the robot moves in. Moreover, in grid-based approaches the "do-nothing"

action reflects the probability of remaining in a node, whereas for the topological approach the probability of remaining in a node must be defined as a function of the space represented by that node.

Hence, the location identifier has been replaced by a vector that contains initial and final position of the node:

$$(x_{i0}, y_{i0}), (x_{if}, y_{if})$$

and the definition of nodes have to be extended to include the length of the node itself (d_i).

9.4.4 Behavioural organisation

The previous subsections described how the probabilistic models were defined but how the nodes calculate and maintain their beliefs remains to be clarified. Figure 9.3 shows the behavioural organisation of the localisation subsystem of the architecture being developed.

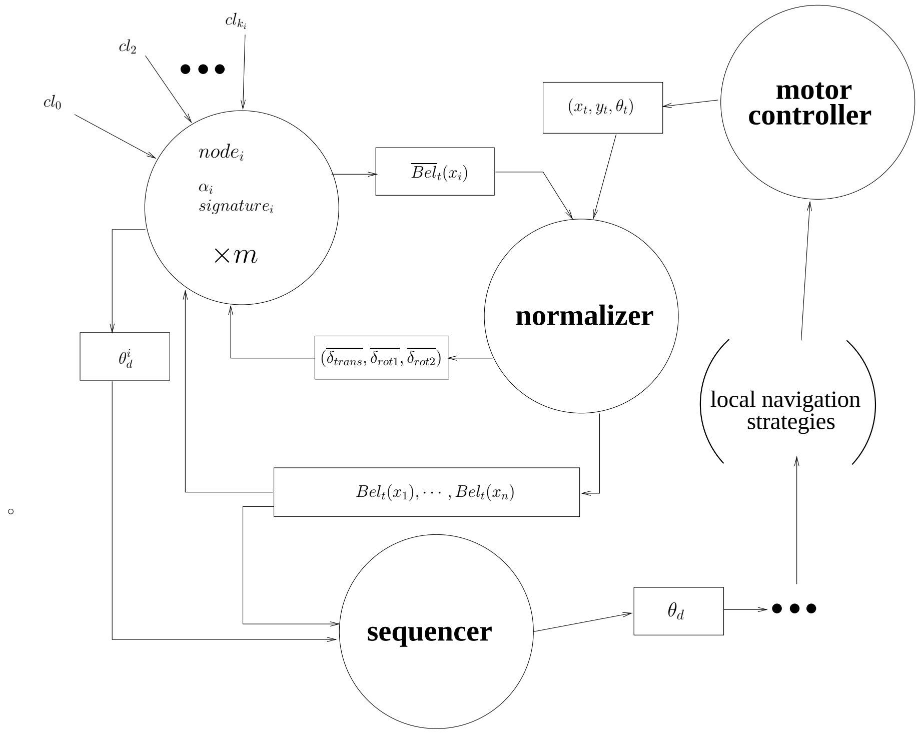


Figure 9.3: Localisation subsystem

Each node i stores the information necessary to calculate $P(x_i|x_j, u_t), \forall j$, i.e. the parameters $(\delta_{rot1}, \delta_{trans}, \delta_{rot2})$ that need to occur to arrive to that node from anywhere else (a single row of the transition matrix). In this manner, every node receives as input the values $(\overline{\delta_{rot1}}, \overline{\delta_{trans}}, \overline{\delta_{rot2}})$ calculated from odometry readings and it is able to calculate the non-normalised belief $\overline{Bel_i}$ of being in that node. An extra thread (*normaliser*) is responsible for reading individual beliefs and normalising them.

The main peculiarity of the system being developed is the qualitative nature of the actions performed by the robot. The nodes decide which bearing the robot must follow to reach the goal while avoiding obstacles. This property is of high relevance because it allows the robot to behave robustly in dynamic environments. The fact that no deterministic action set is defined makes it difficult to determine the effect of an action, and consequently to know when the transition probability must be updated. To cope with this problem, odometric information is accumulated until there is a high degree of certainty about the robot's location, i.e. the belief of being in a node exceeds certain threshold. This normalising process is also responsible for deciding when to accumulate distance and when to restart the reference point according to the normalised belief values.

Finally, the *sequencer* receives the normalised beliefs and the orientation proposals of the nodes as inputs, and acts as a conflict solver, selecting the orientation suggested by the most probable node.

9.5 Experimental evaluation

This section explains how the method has been applied and tested in a simulated and in a real robot/environment system. First, the environment description is given together with the description of the basic navigation modules and landmark identification subsystems needed by the robot. Then, the results obtained in simulation and in the real system are provided.

9.5.1 Definition of the procedural modules

As described in Section 5.5, Galtxagorri's BB control architecture was composed of several independent behavioural modules, mainly:

- Local navigation strategies.
- Landmark identification subsystems.

In order to integrate the ML, new modules were properly added. It must be pointed out that although to a great extent many behaviours were the same as those described in Section 5.5, their implementation was modified to adapt their functionality to the Leuze RS4 laser mounted in Galtxagorri. That is why these behaviours are so briefly described bellow and only the algorithms for new functionalities are shown when necessary.

The whole upgraded architecture is described bellow.

Local navigation strategies: two local navigation strategies allow the robot to navigate towards a goal while avoiding obstacles:

1. Balance free space around the robot (see Algorithm 9.2).

Algorithm 9.2 Laser based free space balancing

```
1: if (first_time) then
2:     first_time = false;
3:     initialise_w_weights_with_gaussians();
4:     initialise_v_weights_with_gaussians();
5: end if
6: average = calculate_laser_readings_average();
7: th = calculate_appropriate_threshold(average);
8: $w_{left} = \sum_{i}^{left,laser_i<th} w\_weight_i * \frac{laser_i}{th} + \sum_{i}^{left,laser_i \geq th} w\_weight_i;$
9: $w_{right} = \sum_{i}^{right,laser_i<th} w\_weight_i * \frac{laser_i}{th} + \sum_{i}^{right,laser_i \geq th} w\_weight_i;$
10: $v_{tmp} = \sum_{i}^{181,laser_i<th} v\_weight_i * \frac{laser_i}{th} + \sum_{i}^{181,laser_i \geq th} v\_weight_i;$
11: $front_{min}$ = calculate_front_shortest_reading();
12: $left_{min}$ = calculate_left_shortest_reading();
13: $right_{min}$ = calculate_right_shortest_reading();
14: $diff = left_{min} - right_{min};$
15: if $(front_{min} < front_{threshold})$ then
16:     $v_{aux} = -\log \frac{front_{min}+CONSTANT\_FRONT\_FAR}{front_{min}+CONSTANT\_FRONT\_SHORT};$
17:     $w_{aux} = sign((int)diff) * CONSTANT\_VEL;$
18: else
19:     $v_{aux} = V\_MAX * v_{tmp};$
20:     $w_{aux} = W\_MAX * (w_{left} - w_{right});$
21: end if
22: $v_t = (0.8 * v_{aux}) + (0.2 * v_{t-1})$
23: $w_t = (0.8 * w_{aux}) + (0.2 * w_{t-1})$
24: return $(v_t, w_t);$
```

2. Maintain a privileged compass orientation.

The overall robot velocities are a result of a cooperative control strategy, obtained by a weighted sum of individual proposals.

Landmark identification subsystems: each landmark identifier outputs a confidence level (*cl*) as a measure of the confidence of the identification process:

1. Corridor identification. Wide halls are considered as non-corridor places (Algorithm 9.3).

$$cl_{corr} \in [0 \cdots 1]$$

Algorithm 9.3 Laser based corridor identification

```
1: shortest_left_value = calculate_laser_left_shortest_value();
2: shortest_right_value = calculate_laser_right_shortest_value();
3: sum = shortest_left_value + shortest_right_value;
4: insert_in_buffer(sum);
5: average = calculate_the_buffer_average;
6: $cl_{corr} = 1 - 1/(1 + e^{-(average - CORRIDOR\_WIDTH)})$;
7: return $cl_{corr}$;
```

2. Mean compass orientation:

$$cl_{\theta} \in [0^{\circ} \cdots 360^{\circ}]$$

3. Left/right wall identification (Algorithm 9.4), useful to distinguish among non corridor places:

$$cl_{wall} \in [-1 \cdots 1]$$

where -1 means right wall and 1 means wall on the left.

The perception of the robot at time t is then:

$$z^t = \left\{ cl^t_{corr}, cl^t_{wall}, cl^t_{\theta} \right\}$$

Algorithm 9.4 Laser based wall identification

```
1: shortest_left_value = calculate_laser_left_shortest_value();
2: shortest_right_value = calculate_laser_right_shortest_value();
3: if ($shortest\_left\_value < shortest\_right\_value$) then
4:     $cl_{wall} = left\_wall$;
5: else
6:     $cl_{wall} = right\_wall$;
7: end if
8: return $cl_{wall}$;
```

Action triggering landmark identifiers: two more landmark identification subsystems are needed for action triggering (orientation changing):

- Crossroad or junction recognition (cl_{cross}): laser based crossroad recognition has substituted the use of emergency exit panels. The new module is insensitive to lightning conditions and produces a better behaviour by identifying the crossroads in situ instead of anticipating the places (Algorithm 9.5).

Algorithm 9.5 Laser based junction identification

```
1: shortest_left_value = calculate_laser_left_shortest_value();
2: shortest_right_value = calculate_laser_right_shortest_value();
3: count_scans = sum_left_scans_larger_than(shortest_left_value);
4: left_scans_values = sum_left_scans_larger_than(shortest_left_value);
5: count_scans += sum_right_scans_larger_than(shortest_right_value);
6: right_scans_values = sum_right_scans_larger_than(shortest_right_value);
7: total_sum = (left_scans_values + right_scans_values)/count_scans;
8: insert_in_buffer(total_sum);
9: average = calculate_the_buffer_average;
10: $cl_{cross} = 1/(1 + e^{-(average - CROSS\_WIDTH)})$;
11: return $cl_{cross}$;
```

- Dead ends identification (cl_{dead_end}) modified as shown in Algorithm 9.6.

Notice that dynamic landmarks (landmarks that persist in spite of robot movement) are used for node definition whereas static landmarks (momentary or robot position specific landmarks) are used for action triggering. In both cases, only the desired compass bearing is changed when these landmarks are positively identified; the combination of the local navigation strategies produce the desired behaviour.

Algorithm 9.6 Laser based dead-ends identification

```
1: calculate_laser_shortest_reading();
2: calculate_laser_largest_reading();
3: $cl_{dead\_end} = 1 - 1/(1 + e^{-(largest\_reading - shortest\_reading)})$;
4: return $cl_{dead\_end}$
```

Action model specification: for the presented approach, the transition probability is decomposed into two components, the translational (p_δ) and the rotational components (p_θ).

For the translational component, instead of defining δ_{trans} as the Euclidean distance, the Manhattan distance (or Minkowsky's L_1 distance) is used: $\delta_{trans} = \triangle_x + \triangle_y$. This distance measures the distance between two points along axes at right angles and has shown to be more appropriate for the given environmental setup. The translational and rotational differences are then:

$$\begin{cases} \delta = \delta_{trans} - \overline{\delta_{trans}} \\ \theta = |\delta_{rot1} - \overline{\delta_{rot1}}| + |\delta_{rot2} - \overline{\delta_{rot2}}| \end{cases}$$

The transition probability is calculated assuming independent error sources for each component:

$$P(x_i|x_j, u_t) = p_\delta \times p_\theta \tag{9.3}$$

where[1]:

$$\begin{cases} p_\delta = N(\delta, \sigma_{trans}) \\ p_\theta = N(\theta, \sigma_{rot}) \end{cases}$$

Again, $P(x_i|x_i, u_t)$ is computed as in equation (9.3) but the component probability values are calculated as follows:

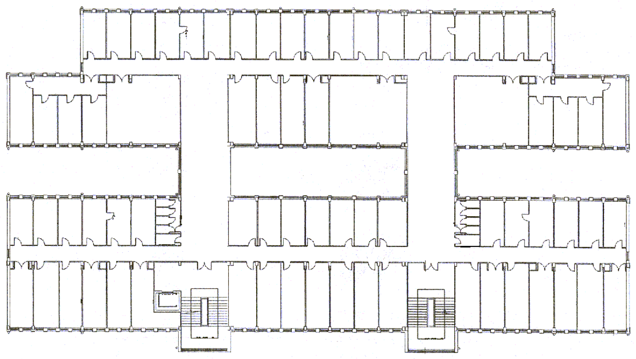
$$\begin{cases} p_\delta = \begin{cases} 0.99 & \text{if } \delta \leq d_i \\ \frac{0.99}{2\times\delta} & \text{otherwise} \end{cases} \\ \\ p_\theta = N(\theta, \sigma'_\theta) \text{ where } \sigma'_\theta = 2 \times \sigma_\theta \end{cases}$$

9.5.2 Simulated experiments

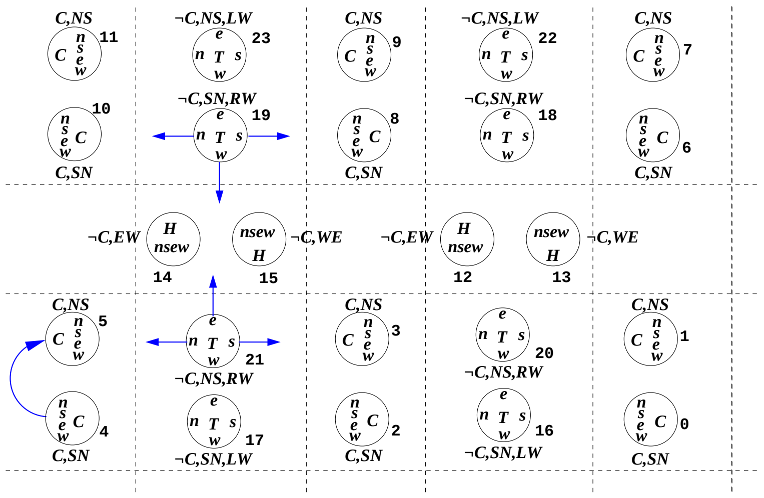
Figure 9.4 shows the map of 24 processes (nodes) associated to the experimental environment in Figure 9.5.

Note that no link information is used for the localisation task. In principle, all transitions are possible. The high degree of symmetry of the environment facilitates the development of clone functions for different locations.

[1] $N(0, \sigma)$ represents a zero centred normal distribution with deviation σ



(a)



(b)

Figure 9.4: The environment and its corresponding procedural map of nodes

Thus, the whole control architecture is composed by 38 threads that communicate amongst themselves.

The robot can be launched at any unknown position of the environment, i.e. the belief distribution is initialised uniformly. It is important to note that landmarks are ambiguous; that is, the robot cannot localise itself simply by observing a single landmark. Rather, the robot must undertake several relatively complex series of actions, and may have to travel a considerable distance to unambiguously determine its location (Howard and Kitchen, 2001). Therefore, the robot first needs to localise itself and then complete the trajectory. An additional *localisator* thread has been added with this aim. This thread is responsible for initially performing a simple strategy, looking for free space into the four nominal orientations to guide the robot to the closest corridor and then following it until the localisation subsystem is stabilised.

In order to test the developed localisation subsystem, three different routes (robotic tasks) have been defined, and several intermediate goals are included in each route. As it can be seen in the left column of Figure 9.5, the route A has two intermediate goals, the first one on the bottom right-hand corner (**node 0**) and the second one on the top left-hand corner (**node 19**); similarly, the route B defines 5 intermediate points as goals, while route C has 7 locations to visit.

The same experiment was made in routes A, B and C, with the robot starting in an unknown place, localising itself and completing the routes three times and showing persistence. The robot completes well its task in all the cases. The right column of Figure 9.5 shows the robot following these routes after self-localisation.

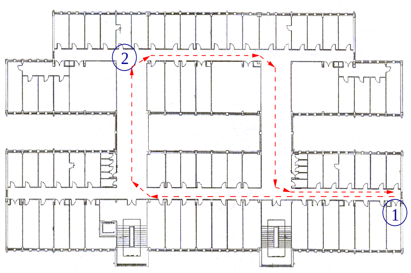
Figure 9.6 shows how the robot performs the localisation strategy and starts completing route A afterwards, starting from a rather complicated initial position (zoomed in the circled area) and verifying the adequateness of the initial localisation strategy (*localisator* module).

9.5.3 Experiments in a real robot/environment system

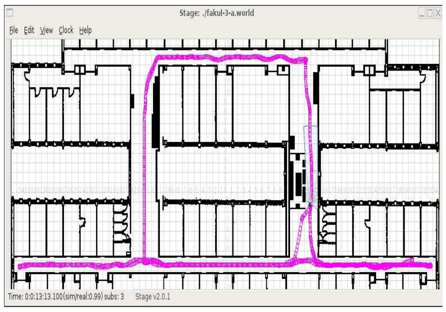
The experiments in the real robot/environment system were performed in half of the third floor of the building (the covered area is represented by a dashed rectangle in Figure 9.7(a)). 8 nodes were needed to complete the map: 4 corridors, 2 transitions and 2 halls (Figure 9.7). The corresponding procedural map is represented in Figure 9.7(b). Eleven nodes were defined: 6 corridor nodes, 3 transition nodes and 2 hall nodes. The robot goals were to arrive to the hall at the end of the middle corridor and to go back to the lab after visiting the small corridor covered by **nodes 2** and **3**.

Figure 9.8 summarises the state of the different modules over time (seconds). The plots in Figure 9.8(a) reflect the state of the landmark identificators during the trajectory, and Figure 9.8(b) shows the index of the node with the maximum belief over time. A location is confirmed when its belief reaches 0.7. Note that although there is an initial uncertainty, once the robot localises itself it never gets lost again even if there are open doors and/or changes in the environment. This sequence confirms that the robot's trajectory matches the predefined sequence of goals.

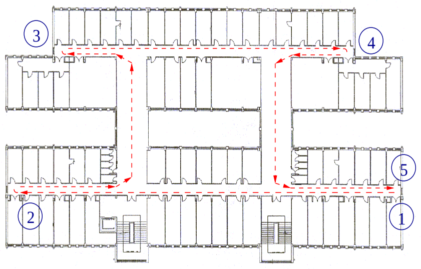
The robot starts in **node 0** although the beliefs are initialised uniformly, i.e. the robot completely ignores where it is. Hence, it starts with the autolocalisation strategy. Due to the different lengths of the two corridors and the slightly different compass bearing defined for each one of them, the



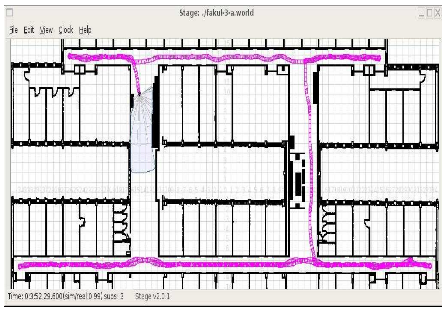
(a) Route A


Stage: ./fakul-3-a.world
File Edit View Clock Help
Time: 0:0:13:13.100(sim/real:0.99) subs: 3 Stage v2.0.1

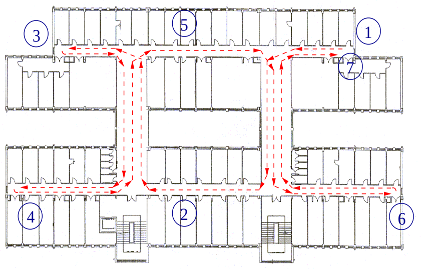
(b) Robot navigating after being localised on Route A



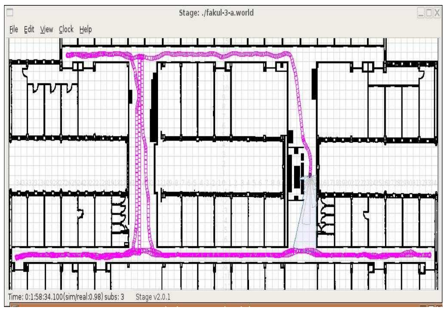
(c) Route B


Stage: ./fakul-3-a.world
File Edit View Clock Help
Time: 0:3:52:29.600(sim/real:0.99) subs: 3 Stage v2.0.1

(d) Robot navigating after being localised on Route B



(e) Route C


Stage: ./fakul-3-a.world
File Edit View Clock Help
Time: 0:1:58:34.100(sim/real:0.98) subs: 3 Stage v2.0.1

(f) Robot navigating after being localised on Route C

Figure 9.5: Left: different routes and intermediate goals. Right: robot performing routes A, B and C

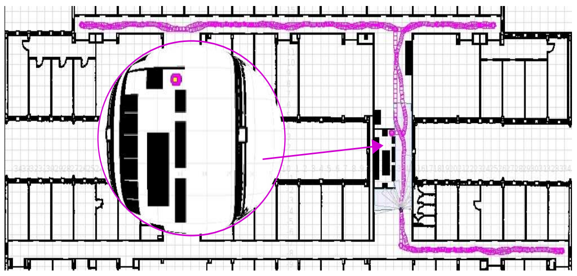
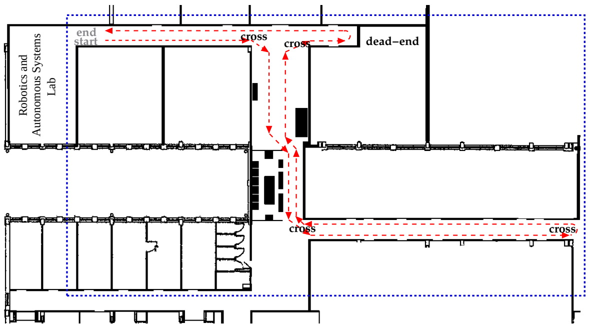
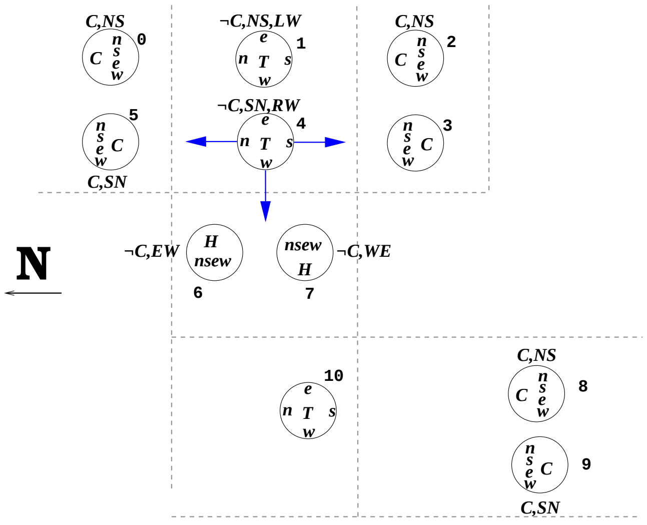


Figure 9.6: Robot localising before route completion

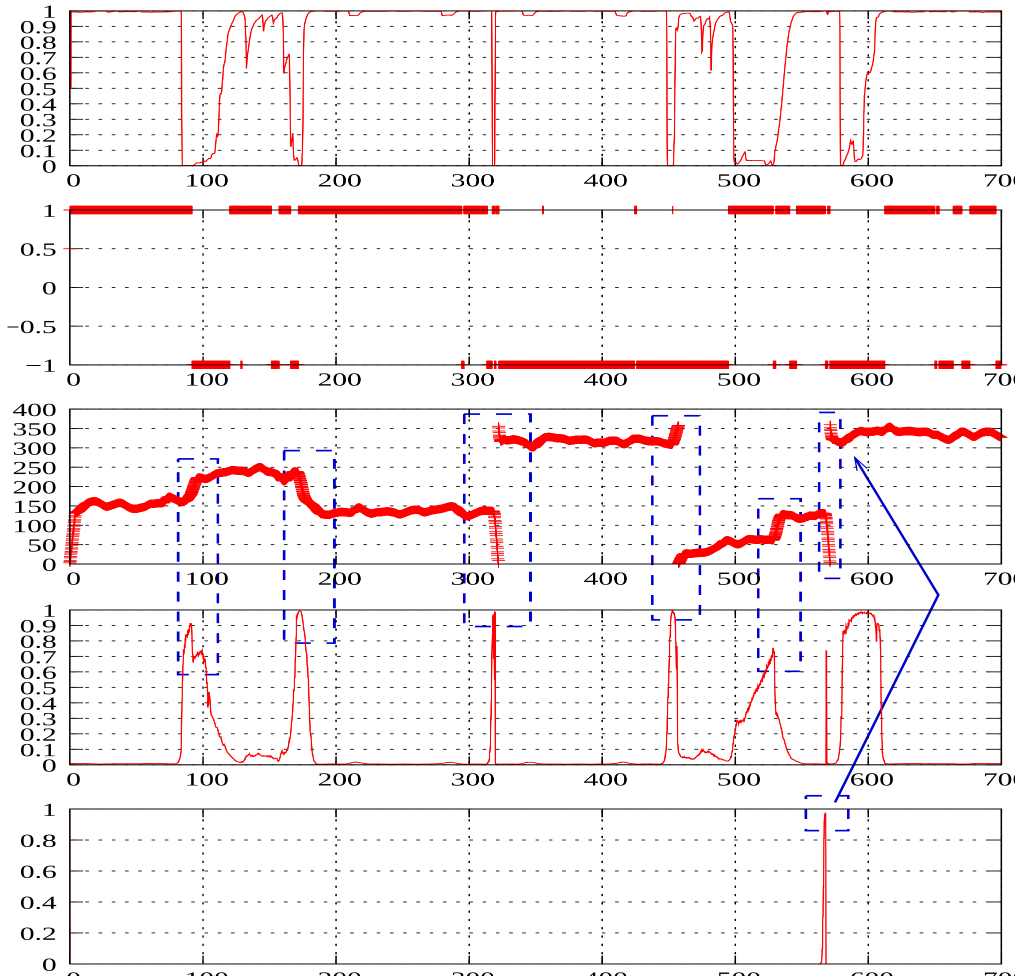


(a) Part of the third floor

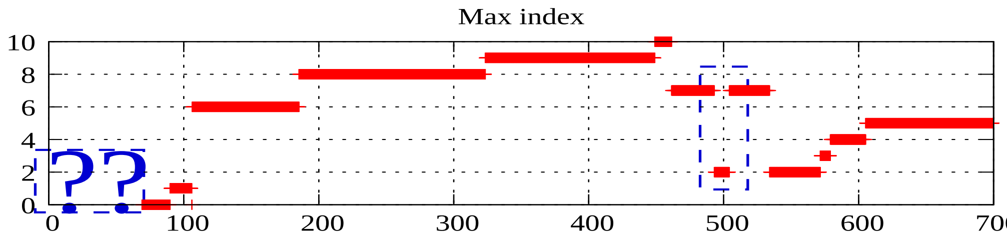


(b) Map of procedural nodes

Figure 9.7: Third floor environment and description



(a) Landmark cl: cl_{corr}, cl_{wall}, $\theta_{compass}$, cl_{cross}, and cl_{dead_end} respectively



(b) Index of the node with the highest belief

Figure 9.8: Floor 3 results

robot only needs a short period to confirm its location at **node 0** ($\sim 70s$) and to start searching for its goals. Once the first transition node is correctly identified ($\sim 100s$), the robot recognises a crossroad ($\sim 180s$) and the desired bearing is changed to direct the robot to the hall; then the robot traverses the hall, turns to its left when it identifies a second crossroad ($\sim 320s$) and follows the larger corridor until it sees a new crossroad ($\sim 450s$). Having reached its first goal fulfilled, the change in the desired compass bearing makes the robot turn on the spot. Afterwards, it retraces its steps and reaches the location corresponding to the transition **node** numbered **10**. To attain its next goal, the transition node changes the desired compass orientation to the east and produces the activation of the upward hall node (**node 7**) up to the end. When the robot reaches that point, it recognises the new crossroad and the corresponding node changes the orientation to the south. It is important to point out that there is a magnetic field that disturbs the compass reading while the robot travels across **node 7** ($\sim 500s$). However, the robot is able to recover from this misreading. Moreover, after identifying the cross ($\sim 520s$) at the end of **node 7**, the laser readings anticipate the presence of the corridor (**node 2**) and although there is no chance for that transition (**node 1**) will be activated, the distance values allow the robot to correctly identify the corridor (**node 2**). Once it reaches the end of the corridor, the activation of the dead-end landmark identification ($\sim 570s$) again makes the robot face North once again and Galtxagorri goes directly to the last goal (**node 5**).

Figure 9.9 confirms the trajectory completed by the robot during its journey. The accumulated odometry error, clearly visible in this figure does not affect the robot localisation.

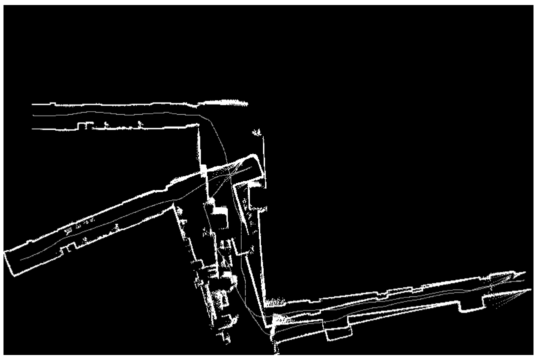


Figure 9.9: Map drawn by Galtxagorri using laser and odometry readings

9.6 Conclusions

In this chapter a preliminary approach to the Markovian localisation can be used in a distributed system and successfully integrated in a behaviour-based control architecture has been described. Galtxagorri's procedural map definition has been modified removing the predecessor/successor links and replacing the location identifier with odometric information. Experimental results are promising. There is no need for the FSA defined in Galtxagorri and the robot is able to recognise and maintain its localisation even when its initial position is unknown. The proposed approach requires low storage and computational resources and it is, in author's opinion, particularly suitable for dynamic environments. The coordinate information included in the nodes allows the adaptation of a common action model used in probabilistic approaches and can easily be acquired within an exploration strategy.

To summarise: on the one hand, the topological quality of the system makes it less sensitive to odometry errors. On the other hand, its procedural nature allows the preservation of the basic functions of the robot at low cost as well as the performance of more complex tasks.

Chapter 10

Behaviour-based mapping: loop-closing as a typicality approach

Contents

As mentioned in Chapter 5 (Section 5.6), Galtxagorri does not have an automatic mapping mechanism. The topological map was given to the robot for the first architecture implementation as well as for testing the probabilistic localisation method described in Chapter 9.

Not having an automatic mapping mechanism represents a big burden for the designer of the map because the perception of robots and humans differs significantly from each other. Moreover, changing the robot's environment implies that the whole topology of the map should be rebuilt and that an automatic mapping method has to be developed to obtain the node information and to establish the spatial relationship among the nodes. In addition, the *loop-closing* problem must be addressed, i.e. correspondences among already visited places must be identified during the mapping process.

In this chapter a place recognition approach based on match testing is proposed using the INCA statistic (Irigoien and Arenas, 2008), a typicality test which follows a distance-based approach. The typicality problem refers to the identification of new classes in a general classification context. This typicality concept is used in this research work to help a robot acquire a topological representation of the environment during its exploration phase.

To build the map, an exploration behaviour is developed whose aim is to build the topological representation of a partially unknown indoor environment. In this context, *partially* means that the robot is endowed with the capability of identifying some environmental structures such as corridors, junctions and halls. The author hypothesised that the robot world is a typical office-like regular environment with parallel walls. Nodes of the topological map represent areas of the environment with the same perceptual properties. During the exploration process, previously visited locations should be identified and new locations should be integrated in the topological structure.

We describe the theoretical basis of the proposed approach and present extensive experimental results performed in both a simulated and a real robot-environment system; as in the previous distributed localisation approach, the behaviour-based philosophy and the definition of the topological procedural map are maintained.

10.1 Literature review

While mapping an environment the robot must determine whether or not it is the first time it visits a certain location. Loop-closing has long been identified as a critical issue when building maps from local observations. Topological mapping methods isolate the problem of how loops are closed from the problem of how to determine the metrical layout of places in the map and how to deal with noisy sensors.

The loop-closing problem cannot be solved neither relying only on extereoceptive information (due to sensor aliasing) nor on propioceptive information (cumulative error). Both environmental properties and odometric information must be used to disambiguate locations and to correct robot position. Fraundorfer et al. (2007) present a highly scalable vision based localisation and mapping method that uses image collections, whereas Se et al. (2005) use vision mainly to detect the so called *loop-closing* –the place has already been visited by the robot– in robot localisation; Tardós et al. (2002) introduce a perceptual grouping process that permits the robust identification and localisation of environmental features from the sparse and noisy sonar data. On the other hand, the probabilistic Bayesian inference, along with a symbolic topological map is used by Chen and Wang (2006) to relocalise a mobile robot. More recently, Olson (2009) presents a new loop-closing approach based on data association, where places are recognised by performing a number of pose-to-pose matchings; a review of loop-closing approaches related to MONOSLAM can be found in (Williams et al., 2009). Within the field of probabilistic robotics (Thrun et al., 2005), Kalman filters, Bayesian Networks and particle filters are used to maintain probability distributions over the state space while solving mapping, localisation and planning.

But the mapping problem can also be stated from a classification perspective. In most classification problems, there is a training data available for all classes of instances that can occur at prediction time. In this case, the learning algorithm can use the training data to determine decision boundaries that discriminate among the classes. However, there are some problems that exhibit only a single class of instances at training time but are amenable to machine learning. At prediction time, new instances with unknown class labels can either belong to the target class or to a new class that was not available during training. In this scenario, two different predictions are possible: *target*, an instance that belongs to the class learnt during training, and *unknown*, where the instance does not seem to belong to the previously learnt class. Within the machine learning community, this kind of problems are known as *one-class* problems and as *typicality* problems within the statistics research.

To give some examples, in (Hempstalk et al., 2008) the probability distributions of the class variable known values are used to determine if a new case belongs to the known class values or if it should be considered as a different class member. One-class classification categorizers have a wide range of applications; in (Manevitz and Yousef, 2007) one-class classification is used to document categorisation in order to decide whether a reference is relevant in a database searching query. The same authors combine this approach with the Support Vector Machine (SVM) paradigm for document classification purposes (Manevitz and Yousef, 2002); and in (Sánchez-Yáñez et al., 2003) the same idea is applied to texture recognition in images. A thorough review of one-class classification can be found in (Tax, 2001).

Regarding the mobile robotics area, one-class classification approaches can be applied to robot mapping, i.e. to learn the structure of its environment in an automatic manner. In this way, Brooks and K. Iagnemma (2009) present a use of this approach to deal with terrain recognition, and Wang and Lopes (2005) use it to identify user actions in human-robot-interaction. However, direct uses of this approach, with this particular name, have not been found in the robotics literature.

There are different approaches found in the literature to deal with the typicality problem (Rao, 1962; McDonald et al., 1976; Cuadras and Fortiana, 2000; Bar-Hen, 2001; Irigoien and Arenas, 2008). Some of them are only suitable for normal multivariate data, others are appropriate for any kind of data but are limited to $k = 2$, being k the number of classes. The latter case offers the most general framework to be applied. However, and in spite of the high diversity of the used methods, to the best of the author's knowledge, neither typicality nor one-class approaches appear in the mapping literature.

The approach proposed in this chapter combines the INCA statistic (Irigoien and Arenas, 2008) with the topological properties of the environmental locations considered and thus represents a new approach to tackling the robot mapping problem as a typicality case.

10.2 Typicality test by means of the INCA statistic

In this section the INCA statistic is introduced and the INCA test is proposed as a solution to the typicality problem.

10.2.1 Preliminaries

The data we consider are random vectors and we assume that distinct classes exist. Let C_1, C_2, ..., C_k be k classes represented as k independent S-valued random vectors $\mathbf{Y}_1$, $\mathbf{Y}_2$, ..., $\mathbf{Y}_k$, with probability density functions f_1, f_2, ..., f_k with respect to a suitable common measure λ. Let $\delta(\mathbf{y}, \mathbf{y}')$ be a distance Gower (1985) function on S. We say that δ is a Euclidean distance function if the metric space (S, δ) can be embedded in a Euclidean space, $\Psi : S \longrightarrow \mathbb{R}^p$, such that:

$$\delta^2(\mathbf{y}, \mathbf{y}') = \|\Psi(\mathbf{y}) - \Psi(\mathbf{y}')\|^2, \tag{10.1}$$

and we may understand $E(\Psi(\mathbf{Y}_i))$ as the δ-mean of $\mathbf{Y}_i$, $i = 1, ..., k$.

In this general framework the following concepts are considered. The geometric variability of C_i, $i = 1, ..., k$ with respect to δ is defined (Cuadras and Fortiana, 1995) as

$$V_\delta(C_i) = \frac{1}{2}\int_{S\times S} \delta^2(\mathbf{y}_{i1}, \mathbf{y}_{i2}) f(\mathbf{y}_{i1}) f(\mathbf{y}_{i2}) \lambda(d\mathbf{y}_{i1})\lambda(d\mathbf{y}_{i2}).$$

This quantity is a variant of Rao's diversity coefficient (Rao, 1982). When δ is the Euclidean distance and $\Sigma_i = COV(\mathbf{Y}_i)$, then $V_\delta(C_i) = tr(\Sigma_i)$. For other dissimilarities $V_\delta(C_i)$ is a general measure of dispersion of $\mathbf{Y}_i$. In the context of discriminant analysis (Cuadras et al., 1997) the squared distance between C_i and C_j is defined by

$$\Delta^2(C_i, C_j) = \int_{S\times S} \delta^2(\mathbf{y}_i, \mathbf{y}_j) f(\mathbf{y}_i) g(\mathbf{y}_j) \lambda(d\mathbf{y}_i)\lambda(d\mathbf{y}_j) - V_\delta(C_i) - V_\delta(C_j) \tag{10.2}$$

This quantity is the Jensen difference (Rao, 1982) between the distributions of C_i and C_j. If the metric space (S, δ) can be embedded (see (10.1)) in a Euclidean space $\mathbb{R}^p$ and if $E(\|\Psi(\mathbf{Y}_i)\|)$ and $E(\|\Psi(\mathbf{Y}_i)\|^2)$ are finite, then $V_\delta(C_i) = E(\|\Psi(\mathbf{Y}_i)\|^2) - \|E(\Psi(\mathbf{Y}_i))\|^2$, $i = 1, ..., k$, and $\Delta^2(C_i, C_j) = \|E(\Psi(\mathbf{Y}_i)) - E(\Psi(\mathbf{Y}_j)\|^2$. If there is only one element $C_i = \{\mathbf{y}_0\}$, (10.3) gives the proximity function of $\mathbf{y}_0$ to C_j,

$$\phi^2(\mathbf{y}_0, \mathbf{Y}_j) = \int_S \delta^2(\mathbf{y}_0, \mathbf{y}_j) f(\mathbf{y}_j)\lambda(d\mathbf{y}_j) - V_\delta(\mathbf{C}_j). \tag{10.3}$$

In applied problems the distance function is typically a datum, but the probability distribution for each population is unknown. Natural estimators given samples $\mathbf{y}_1^1, ..., \mathbf{y}_{n1}^1$, ..., $\mathbf{y}_1^k, ..., \mathbf{y}_{nk}^k$, of sizes n_1, ..., n_k coming from C_1, ..., C_k are the following:

- The *geometric variability* of C_j,

$$\hat{V}_\delta(Cq_j) = \frac{1}{2n_j^2}\sum_{l,m} \delta^2(\mathbf{y}_l^j, \mathbf{y}_m^j).$$

- The *proximity function* of a new object $\mathbf{y_0}$ to C_j,

$$\hat{\phi}^2(\mathbf{y}_0, C_j) = \hat{\phi}_j^2(\mathbf{y}_0) = \frac{1}{n_j}\sum_l \delta^2(\mathbf{y}_0, \mathbf{y}_l^j) - \hat{V}_\delta(C_j).$$

- The *squared distance* between C_i and C_j,

$$\hat{\Delta}^2(C_i, C_j) = \hat{\Delta}^2_{ij} = \frac{1}{n_i n_j} \sum_{l,m} \delta^2(\mathbf{y}^i_l, \mathbf{y}^j_m) - \hat{V}_\delta(C_i) - \hat{V}_\delta(C_j). \qquad (10.4)$$

See (Arenas and Cuadras, 2002) and references therein for a review of these concepts, their application, different properties and proofs.

10.2.2 INCA statistic

Consider that n units are simply divided into k classes $C_1, ..., C_k$, of sizes $n_1, ..., n_k$. Consider a fixed unit $\mathbf{y}_0$, which may be an element of a C_j, $j = 1, ..., k$ or may belong to an unknown class, i.e. it may be an atypical unit. Consider a new class with δ-mean the linear combination $\sum_{i=1}^k \alpha_i E(\Psi(\mathbf{Y}_i))$, where $\mathbf{Y}_i$ is the random vector representing the class C_i, $i = 1, ..., k$. The INCA statistic is defined as follows:

$$W(\mathbf{y}_0) = \min_{\alpha_i} \{L(\mathbf{y}_0)\}, \qquad \sum_{i=1}^k \alpha_i = 1, \qquad (10.5)$$

$$L(\mathbf{y}_0) = \sum_{i=1}^k \alpha_i \phi_i^2(\mathbf{y}_0) - \sum_{1 \le i < j \le k} \alpha_i \alpha_j \Delta^2_{ij}.$$

$\phi_i^2(\mathbf{y}_0)$ is the proximity function of $\mathbf{y}_0$ to C_i and Δ^2_{ij} is the squared distance between C_i and C_j. The INCA statistic $W(\mathbf{y}_0) = \min_{\alpha_i} L(\mathbf{y}_0)$ trades off between minimising the weighted sum of proximities of $\mathbf{y}_0$ to classes (which takes into consideration the within-group variability) and maximising the weighted sum of the squared distances between classes (between-groups variability) - a common behaviour of a classing criterion. The values of $\alpha' = (\alpha_1, \ldots, \alpha_{k-1})$ together with $\alpha_k = 1 - \sum_{i=1}^{k-1} \alpha_i$, verifying (10.5) are $\alpha' = M^{-1}N$, where M is the $(k-1) \times (k-1)$ matrix

$$M = \left(\Delta^2_{ik} + \Delta^2_{jk} - \Delta^2_{ij}\right)_{i,j=1,\ldots,k-1}$$

and N is the $(k-1) \times 1$ vector

$$N = \left(\Delta^2_{ik} + \phi_k^2(\mathbf{y}_0) - \phi_i^2(\mathbf{y}_0)\right)_{i=1,\ldots,k-1}.$$

The statistic $W(\mathbf{y}_0)$ has a very nice geometric interpretation. It can be interpreted (see Figure 10.1) as the (squared) orthogonal distance or height h

of $\mathbf{y}_0$ on the hyperplane generated by the δ-mean of C_i $(i = 1, ..., k)$, denoted in Figure 10.1 by a_i, $i = 1, ..., k$. Then, points which lie significantly far from this hyperplane are held to be outliers. This intuitive idea is used to detect outliers among existing classes.

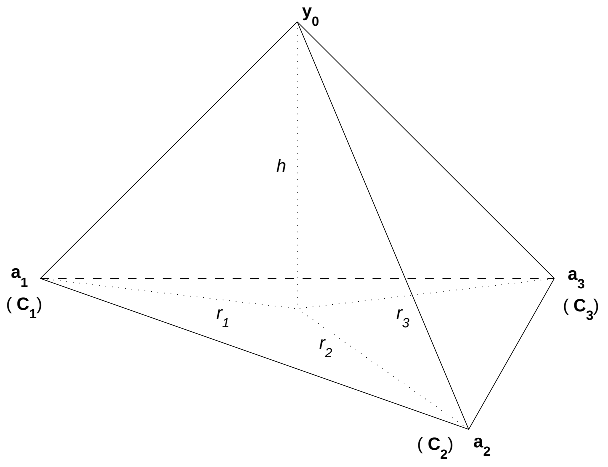


Figure 10.1: For $k = 3$, new observation $\{\mathbf{y}_0\}$, centres of classes $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$ and (squared) projection $\mathbf{r}_i$ of the edges $\{\mathbf{y}_0,\ \mathbf{a}_i\}$ on the plane $\{\mathbf{a}_1,\ \mathbf{a}_2,\ \mathbf{a}_3\}$. The (squared) height $\mathbf{h}$ is $W(\mathbf{y}_0)$

Suppose now that the data are classified in k classes. Let $\mathbf{y}_0$ be a new observation and consider the test to decide whether $\mathbf{y}_0$ belongs to one of the fixed classes C_j, $j = 1, ..., k$ or, on the contrary, it is an outlier or an atypical observation which belongs to a different and unknown class. Consider the INCA test,

$$\begin{aligned} H_0 \ &: \ \mathbf{y}_0 \text{ comes from the class with} \\ &\quad \delta\text{-mean} \sum_i \alpha_i E(\Psi(\mathbf{Y}_i)), \quad \sum_{i=1}^{k} \alpha_i = 1,\ i = 1, ..., k, \\ H_1 \ &: \ \mathbf{y}_0 \text{ comes from another unknown class,} \end{aligned}$$

and compute statistic (10.5). If $W(\mathbf{y}_0)$ is significant it means that $\mathbf{y}_0$ comes from a different and unknown class. Otherwise we allocate $\mathbf{y}_0$ to C_i using the rule:

$$\text{Allocate } \mathbf{y}_0 \text{ to } C_i \text{ if } U_i(\mathbf{y}_0) = \min_{j=1,\ldots,k}\{U_j(\mathbf{y}_0)\}, \tag{10.6}$$

where $U_j(\mathbf{y}_0) = \phi_j^2(\mathbf{y}_0) - W(\mathbf{y}_0)$, $j = 1, ..., k$.

It can be observed (Irigoien and Arenas, 2008) that $U_j(\mathbf{y}_0)$ represents the (squared) projection of $\{\mathbf{y}_0, E(\Psi(\mathbf{Y}_i))\}$ on the hyper plane $\{E(\Psi(\mathbf{Y}_1)), \ldots,$

$E(\Psi(\mathbf{Y}_k))\}$. See Figure 10.1, where for simplicity the (squared) projection $U_j(\mathbf{y}_0)$ is denoted by r_j, $j = 1, ..., k$. Hence, criterion 10.6 follows the next geometric and intuitive allocation rule: Allocate $\mathbf{y}_0$ to C_i if the projection $U_i(\mathbf{y}_0)$ is the smallest.

We obtained sampling distributions of $W(\mathbf{y}_0)$ and $U_j(\mathbf{y}_0)$ $(j = 1, ..., k)$ by re-sampling methods, in particular drawing bootstrap samples as follows. Draw N units $\mathbf{y}$ with replacement from the union of $C_1, \ldots, C_k$ and calculate the corresponding $W(\mathbf{y})$ and $U_j(\mathbf{y})$ $(j = 1, ..., k)$ values. As usual, this process is repeated $10P$ times with $P \geq 1$ selected by the user. In this way, the bootstrap distributions under H_0 are obtained.

10.3 Topological places

Within the behaviour-based approach, topological maps should be composed of tightly coupled behaviours specific to the meaningful locations. The overall "map" is then composed of sets of behaviours, each launched on a different thread. Let us remember the definition of the topological places.

A topological map is formally defined as a set of nodes where each node consists of:

1. A set of inputs (from landmark identification subsystems) and outputs. These outputs should serve to reduce the distance between the current state and the goal.

2. A signature that identifies the node: $signature_i$. Each locations has a signature that reflects the state of a set of specific landmarks and that is used by the robot for localisation purposes.

3. A function α_i to be executed when the node i is active and that will output the action to be performed at the node specific current state. The behaviour of the robot as well as the associated function of the nodes can be different depending on the location.

4. The location identifier that contains initial and final position of the node:
$$(x_{i0}, y_{i0}), (x_{if}, y_{if})$$

As mentioned earlier, the environment is only partially unknown to the robot since it is provided with behaviour modules to properly identify certain features such as corridors and straight walls.

The same geometric properties of the world used in the previous experimental phases of the development of the architecture have been used and three types of places have been defined: corridors, crossings or junctions and halls, each of them identifiable using distance sensors such as a laser scanner. The orientations of the possible alternative ways at the junctions are registered according to the robot heading provided by the compass sensor and the indexes of the laser scan that define the different intervals, the orientation of the possible alternative ways at the junctions are registered.

The same action triggering landmark identification subsystems are used for the exploration behaviour. *Dead-ends* are also identified for action execution purposes such as changing the orientation of the robot and starting the mapping process. Again, laser readings are used to classify the robot location as a dead-end.

The goal of the mapping process is to fill in the nodes with the information that they must contain. More precisely, the contents of the signature and the location identifier. For this aim, during the learning process and depending on the state of the landmark identification subsystems, i.e. the confidence level of the corridor/hall/junction (cl_{corr}, cl_{hall} and cl_{cross}), the following information is given to the INCA test:

- Initial and mean heading values: θ_0, θ_{mean}.
- Initial and final pose obtained by the odometric subsystem. These poses correspond to the position values of the robot when the node signature activates/deactivates: $(x_0, y_0), (x_f, y_f)$.
- Length (previously named as *duration*) of the area calculated using the initial and final pose information: d.
- Number of alternative ways and their associated orientation: num_ways and $\theta_{w_1}, \cdots, \theta_{w_{num\_ways}}$.

These measurements will constitute the observations of the random vectors $\mathbf{Y}$ considered in the INCA statistic, as represented in Equation 10.7.

$$\begin{array}{l}\text{Corridors, Halls:}\\ \mathbf{Y} = (\sin(\theta_0), \cos(\theta_0), \sin(\theta_{mean}), \cos(\theta_{mean}), (x_0, y_0), (x_f, y_f), d)\\ \\ \text{Junctions:}\\ \mathbf{Y} = (\sin(\theta_0), \cos(\theta_0), \sin(\theta_{mean}), \cos(\theta_{mean}), \theta_{w_1}, \cdots, \theta_{w_{num\_ways}}, (x_0, y_0))\end{array} \tag{10.7}$$

Note that there are two types of measurements: variables type coordinates in meters and variables type orientation in degrees.

The corridors/halls/crosses can differ in their orientation (mean compass value that the robot maintains when going through them in its canonical path). This is why each physical place will correspond to two or more different nodes in the topological map.

10.4 Proposed approach

The locations the robot must identify are not only single points but areas surrounding these points. Therefore, we propose firstly, a data generation approach to characterise the areas; and secondly, the application of the INCA test.

Let us assume that the robot has recorded the geometric information (see section 10.3) of k different places $C_1, \ldots, C_k$, all of them of the same type. There is only one $\mathbf{y}_i$ measurement for each place C_i ($i = 1, \ldots, k$). However, the place we want to identify topologically is not just a spot but an area or neighbourhood of the recorded measurement $\mathbf{y}_i$. In order to do so we generate $n_i - 1$ new observations for each place i which will make up the observations corresponding to the place C_i. These new observations are generated as $\mathbf{y}_i^l = \mathbf{y}_i + U(-u, u)$, $l = 2, \ldots, n_1$, where $U(-u, u)$ stands for the uniform distribution with parameters $-u$ and u ($u > 0$). Taking into account that the robot records two kinds of variables, metres and degrees, we consider two kinds of values for the parameter of the uniform distribution, let us call them, u_M and u_{DEG}, respectively.

Once the data corresponding to the k classes –places– are generated, and given $\mathbf{y}_0$, the information the robot has recorded when he arrives at a new place, the INCA test can be applied and consequently it is possible to decide whether or not $\mathbf{y}_0$ corresponds to a new place. In case it is decided $\mathbf{y}_0$ is not a new place, the conclusion is that $\mathbf{y}_0$ is one of the places $C_1, \ldots, C_k$ according to rule (10.6).

10.5 Parameter selection

To evaluate the proposed approach, the *Player/Stage* robot environment simulator was used (Gerkey et al., 2003) with a Pioneer 2-DX robot. First of all, the three parameters mentioned in Section 10.4 had to be set up, that is the number of observations n_i generated for each class or place and the values u_M and u_{DEG} of the two uniform distributions used to perform the data generation. Taking into account the kind of environment to be explored, we considered the following values as suitable for these parameters:

- $n_i = 5, 10$ or 20.
- $u_M = 1, 2$ or 5.
- $u_{DEG} = 10, 20$ or 30.

That makes a total of 27 possible combinations; we performed 10 trials for each one of them. In the following section the experimental conditions are described, and the parameter evaluation performed is explained.

10.5.1 Data used

In order to set the value of each of the parameters, the data provided by the robot in the *Player/Stage* simulator is used, with default odometry error (Gaussian distribution with standard deviation per meter 0.03, 0.03 and 0.05 for X, Y and angle, respectively). The robot is able to distinguish among corridors, halls and crossing locations, and writes the information of each recognised place in three different databases. We let the robot wander in its environment for a long time, and wait until each different place is encountered at least once; after this data extraction, we have collected a set of 78 corridors, 61 halls and 73 crossovers.

We then perform the parameter selection using the corridor related data. We know that 17 among the 78 corridors are different from the rest, so that there are 61 cases in the database which belong to an already visited place. It is common practise to measure the goodness of a test by the *Type I error* probability and by the *Power* of the test. Type I error in hypothesis testing consists on rejecting the null hypothesis when the data are consistent with it. In the hypothesis we are testing it would mean to decide that a class is a new class when it actually has been visited before. The probability of Type I error is called the *level* of the test. But additionally, a test should be able to reject the null hypothesis when it is false, and the corresponding probability is what it is called the *Power* of the test. We decided to measure both the *Power* and the *Level* of the INCA test for each possible combination of the parameter values in order to select the appropriate set of values to be used in the experimental phase. It is worth mentioning that we used the Pearson distance to calculate the INCA statistic.

The power of the test

In this phase, we selected the first occurrence of each of the 17 different corridors. Each one of them was tested with respect to the remaining 16 corridors. Therefore, in this leave-one-out procedure, each corridor has to be classified as a new –and unknown– one. This procedure was applied 10 times to each possible combination of parameter values ($27 \times 10 = 270$ total trials).

Only in 7 trials out of the 270 trials carried out, a new and unknown corridor was misclassified as "visited before". Hence, 263 trials showed no misclassified corridor and as for the remaining 7 trials, 4 of them showed only 1 misclassified corridor (out of the 17 corridors tested in each trial), and 3 of them showed 2 misclassified corridors. Moreover and as expected, the number of trials (nb.trials) with one or two misclassified corridors decreased when the size n_i increased ($n_i = 5$, nb.trials = 4; $n_i = 10$, nb.trials = 2; $n_i = 20$, nb.trials = 1). Also, it is worth pointing out that all the trials with misclassified corridors corresponded to the value $u_M = 5$.

The level of the test

In this section, and using the same 17 corridors of the previous phase, the goal was to classify each of the remaining 61 corridors in their corresponding class, i.e. no new class answer is expected, and the label of the existing class returned by the classification approach should be the correct one for each corridor already visited.

The evaluation of the level of the test for different values of the parameters showed also satisfying results which varied according to parameter values. Table 10.1 summarises the distribution of the number of misclassified corridors according to the values of the parameters. Me stands for the median and R for the range of the number of times an already visited corridor has been misclassified as a new corridor out of the 61 corridors tested in each trial. The number of misclassified units decreased as the size n_i increased and so did the dispersion, especially when increased from $n_i = 5$ to $n_i = 10$. The worst results were obtained for $u_M = 1$ along with $u_{DEG} = 10$ and $u_{DEG} = 20$.

According to these results the selected parameter values have been $n_i = 10$, $u_M = 2$ and $u_{DEG} = 30$. To clarify this selection, the distribution of the number of misclassified units out of the 61 units tested is shown in Figure 10.2 for $n_i = 10$ and different values of u_M and u_{DEG}; the selected values obtain a Median equal to 0 with a Range equal to 2, which are the best results among all the values. Note that $u_M = 5$ is discarded as mentioned in the evaluation of the power of the test. The same parameter values have been used to deal with halls and crossovers.

		$n_i = 5$		$n_i = 10$		$n_i = 20$	
u_{DEG}	U_M	**Me**	**R**	**Me**	**R**	**Me**	**R**
10	1	15	9	15	2	15.5	1
	2	11	2	11	2	11	2
	5	0	0	0	0	0	0
20	1	6	8	6.5	7	6.5	4
	2	5	7	4	5	3	4
	5	0	2	0	0	0	0
30	1	1	4	1.5	2	0.5	2
	2	1.5	3	0	**2**	0.5	2
	5	0	0	0	0	0	0

Table 10.1: Number of misclassified units out of 61 corridors tested in each trials, depending on the values of the parameters considered

10.6 Exploration behaviour

As stated earlier, the mapping process requires an exploration strategy to guide the robot for the terrain inspection. The strategy used in this proposal, the exploration behaviour is a coordination of the local navigation strategies and landmark identification subsystems the robot is endowed with. The proper combination of these behaviours, described in Section 9.5.1, and listed bellow for sake of clarity, allow the safe exploration of the environment.

- Two local navigation strategies that are combined in a cooperative manner (weighted sum): balance the free space at both sides of the robot and follow a desired compass orientation (θ_d).
- Landmark identification subsystems that allow the robot to recognise corridors, left/right walls, halls, junctions and dead-ends. These landmarks are used to change robot's desired orientation. To show an example, Figure 10.3 shows the coordination of the modules for the case where a dead-end is recognised.

Although the robot can be positioned at any starting location, initially and until the robot reaches a dead-end the map remains empty. Hence, the map construction starts after a dead-end has been identified. This gives the correct measurement of the length of the locations (nodes). Afterwards, the first corridor, the first crossing and the first hall are always identified as new nodes since there is not any instance of the same type already stored in the map.

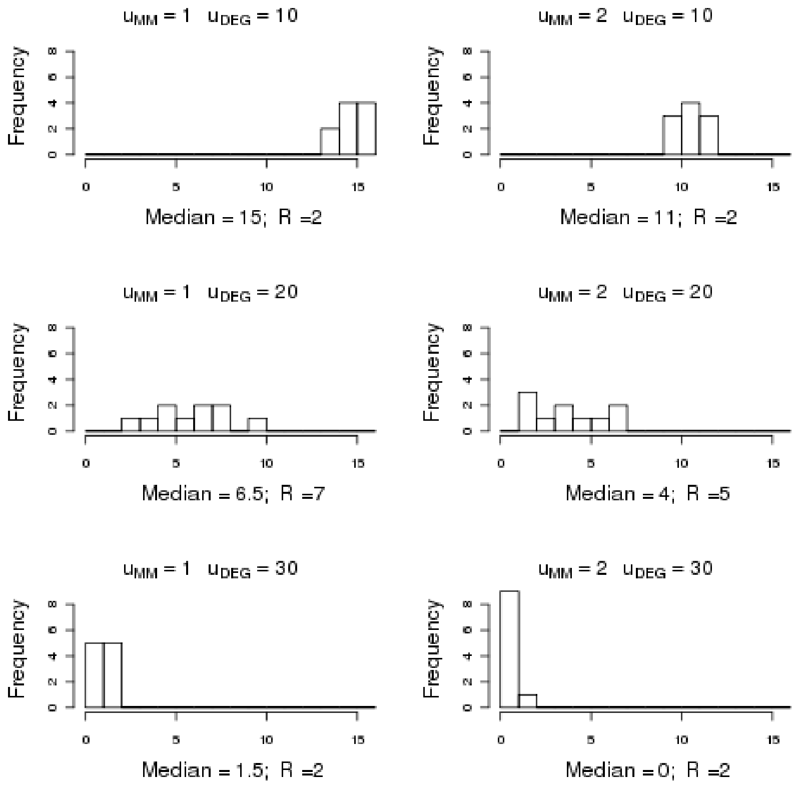


Figure 10.2: Distributions of the number of misclassified units (out of the 61 units tested) for different values of u_M and u_{DEG} when $n_i = 10$, along with the corresponding median and range of the distribution

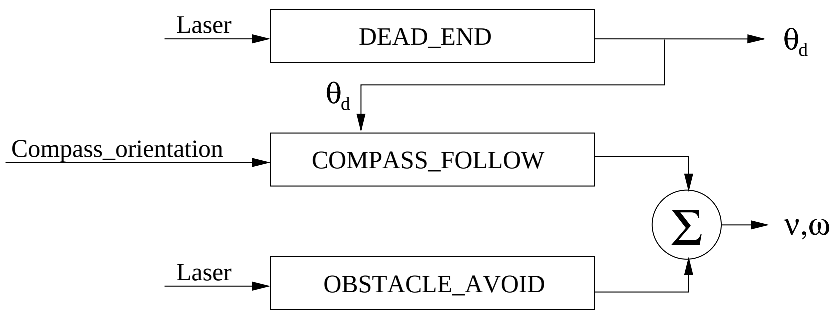


Figure 10.3: Diagram of behaviours modules (v: translational velocity, w: angular velocity)

Once the map building process starts, each time the robot identifies a location – a corridor, a hall or a crossing – the geometric information of the identified location is recorded (the **Y** vector, Equation 10.7), and then the INCA test is applied to evaluate if they are locations already visited or new ones. When the location corresponds to a crossing, i.e. a junction, the orientations of the alternative ways the robot can choose are recorded. If the location has been visited before, one of the non-explored paths is randomly selected. In this way, the robot has the chance to cover all the environment.

The robot finishes the exploration process when all the alternatives of the crossing nodes have been tried.

10.7 Simulated experiments

Experiments were carried out in the third floor of the Faculty of Computer Science. This environment is a semi-structured office-like common environment, with regular geometry as can be seen in Figure 10.4.

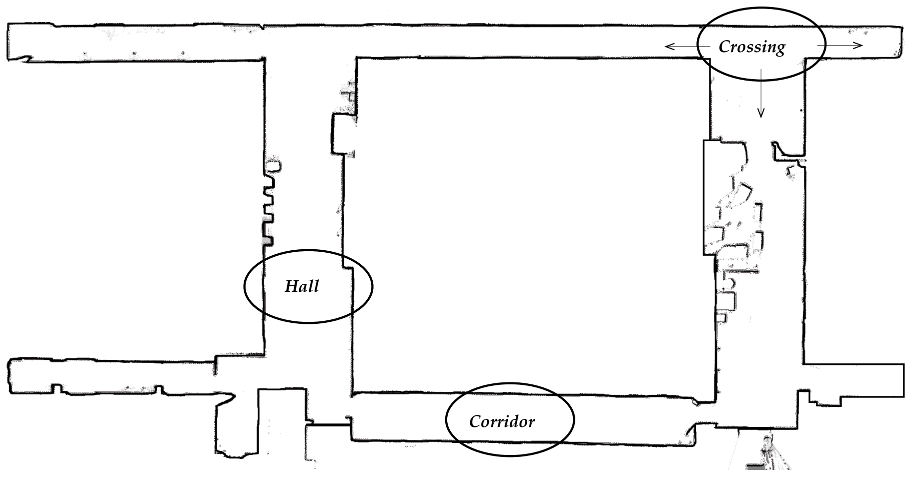


Figure 10.4: Third floor of the Faculty of Computer Science. Approx. 60 × 22 meters

The parameter selection obtained in the previous experimental phase was applied to the more general problem of identifying the whole set of environmental locations during an exploration phase performed in simulation. To this purpose the *Stage* simulation tool was used together with the *Player* robot server.

In order to have a wider view of the mapping process, we let the robot move in the environment for a long time (more than 6500 seconds).

Figure 10.5 shows the robot's path starting from the dead-end at the bottom left corner.

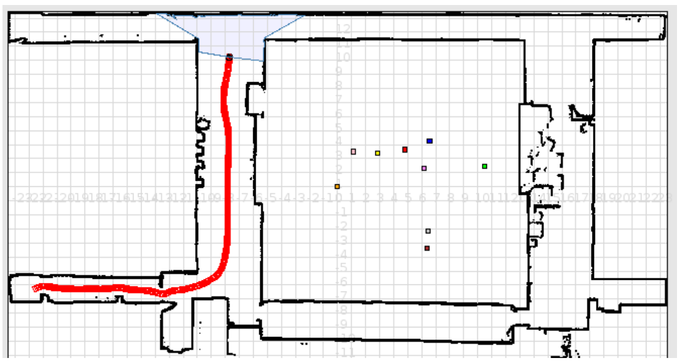


Figure 10.5: The map construction starts after a dead-end

Figure 10.6 shows the path drawn by the robot during the exploration phase.

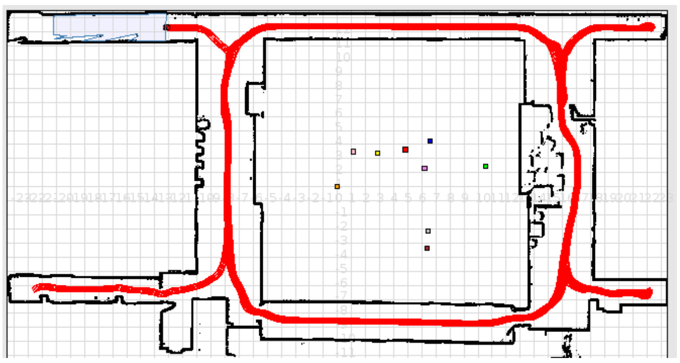


Figure 10.6: The path followed during the exploration

Figure 10.7 shows the complete path followed during the exploration of the environment.

Related to the number of nodes, the map converged to 38 nodes: 17 corridors, 8 halls and 13 crosses (Figure 10.8).

Table 10.2 shows the number of nodes that have been traversed in the path followed by the robot.

		Corr	Hall	Cross
New	Expected	17	8	13
	Found	100%	100%	100%
Known	Traversed	47	23	38
	Classified	100%	100%	100%

Table 10.2: Experimental results

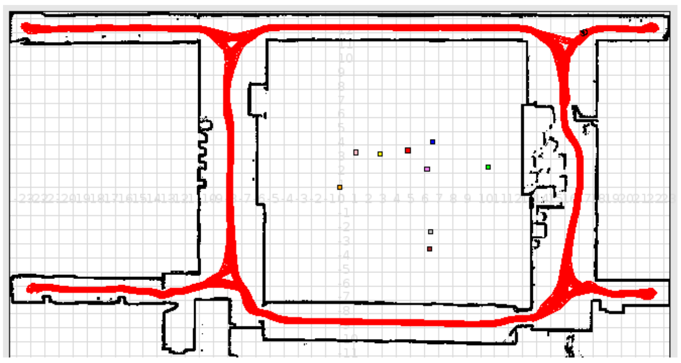


Figure 10.7: The complete path resulting from the exploration process

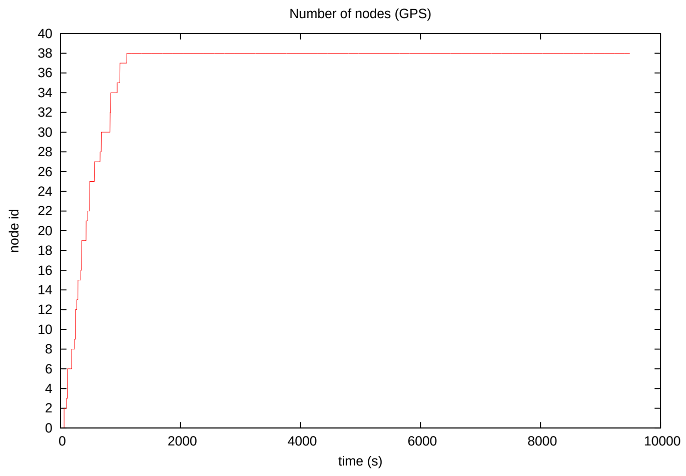


Figure 10.8: Evolving number of nodes

As it can be seen, all the nodes are correctly classified:

- Each of the 17 existing corridors were properly labelled as new places the first time the robot went along them; the same happened with the new traversed halls and crossing nodes.
- The nodes visited more than once by the robot in this long journey were also properly classified with their corresponding label; a total number of 47 corridors, 23 halls and 38 crossing nodes were visited in the robot path.

Figure 10.9 shows the distribution of the locations (plotted according to their central poses) and the evolution of the number of nodes over time.

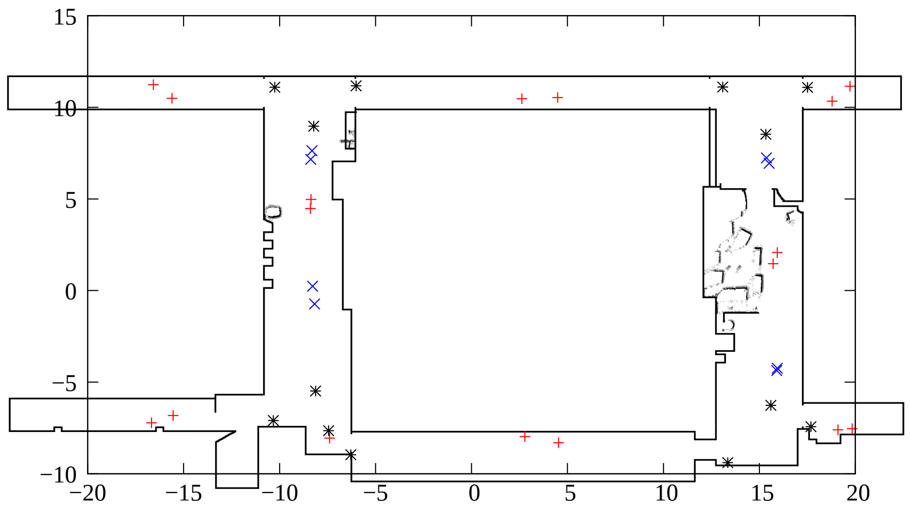


Figure 10.9: Location distribution over the map. Corridors: +; Halls: x; Crossings: *

In spite of the degree of symmetry of the environment, the spatial configuration of the obtained locations does not show the same degree of symmetry. This is due to the fact that robot's and humans' perception differ from each other, and since the robot navigates according to a desired compass heading, depending on its orientation it makes the same physical place correspond to several nodes in the topological representation.

10.8 Experiments in the real robot/environment system

The simulation experiments showed that the proposed approach can solve the stated problem. To test the robustness of the approach experiments were extended to the real robot-environment system. This time the robot Tartalo described in Section 3.1 is used for the empirical evaluation of the mapping system developed. But instead of relying on raw odometry information, two odometry correction methods were tested to smooth the positioning error:

- *Laser stabilised odometry* (by means of the LODO driver provided by *Player*). Laser data is used to correct the raw odometry estimate that once corrected exhibits a drift rate that is an order of magnitude less than the rate observed using pure odometry (Howard, 2005).
- Compass based odometry (CODO), where compass heading is used to correct raw odometric poses.

Experiments were performed in the third floor of the Faculty of Computer Sciences. Figure 10.10(a) shows the path completed by the robot (according to compass based odometry) and Figure 10.10(b) shows the evolution of the number of nodes over time (s) for the different positioning methods. Clearly, the compass odometry obtained with the proposed approach offers the most precise position information.

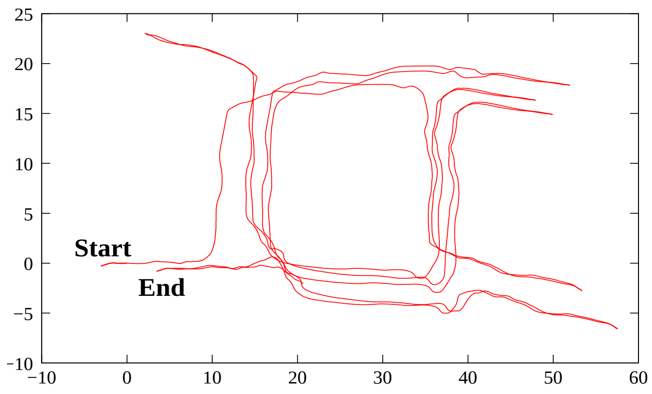
On the other hand, Figure 10.11 shows the distribution of locations of the different nodes obtained from the simulated experiments previously described (Figure 10.11(a)) and the run performed by the robot (Figure 10.11(b)). As mentioned in the previous section, the difference in perception explains the fact that the number of nodes acquired by robot and humans differ from each other.

And, as expected, the number of nodes is higher when the mapping is performed by the robot because of its perception of the environment and its positioning error. However, although the number of junction nodes identified is higher in the real run, this is mainly due to the people and furniture the robot comes across, which produce nodes that lead to any number of alternative paths. However, after an exploration of about an hour and a half (more than 500 meters), the robot was able to close the loop and to recognise several times the final location as the starting one, thus confirming the suitability of the proposed approach.

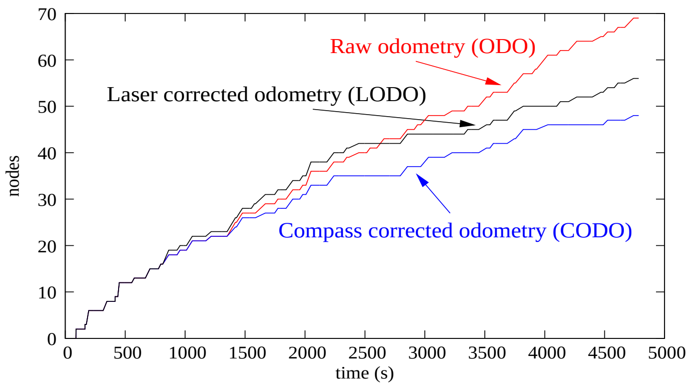
As mentioned earlier, the experiments performed in simulation cannot be directly compared with the experiments with the real robot; the simulated sensor readings produce nodes with different characteristics specially when junction nodes are identified. Hence, the path produced by the exploration strategy in simulation differs from the path executed by the real robot. However, it is interesting to compare the evolution of the learning process using exact odometry with the evolving number of nodes when the odometry is corrected using the compass sensor. The map obtained simulating ideal odometry converged to 38 nodes and the map obtained by the robot after 4500 seconds contained 48 nodes (see Figure 10.12).

10.9 INCA for localisation

During the previous experiments the learning process was not stopped once the loop was closed. This methodological criterion was chosen to asses the appropriateness of the approach, and as a result, there was a slow increase in the number of nodes over time mainly due to odometry error. However, in practical terms the map learning process can be stopped and then use the learnt map for localisation purposes.



(a) Robot's path (CODO)

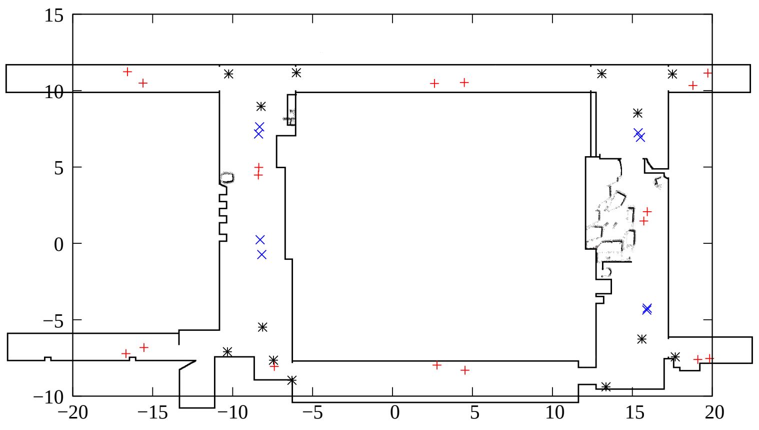


(b) Evolving number of nodes

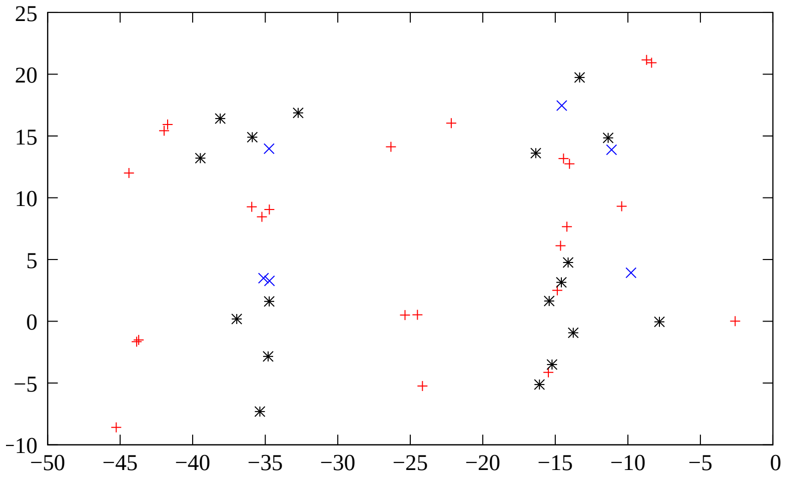
Figure 10.10: Results

The experiments described in this section were carried out to measure the usefulness of the acquired map for localisation. In this occasion, instead of a non-stop learning process, a criterion was set so that the generation of the map would stop once a certain number of nodes was included. Once the procedure reaches this value, no more nodes are allowed to be created and hence, classification rule 10.6 (Section 10.2.2) gives the closest node according to the available data. In this manner, after the map is completed the robot continues moving according to its exploration strategy while the mentioned rule gives its localisation. It is worth to mention that classification rule 10.6 is equivalent to the distance based classifier introduced in (Cuadras, 1992).

Experiments were conducted both in simulation and in the real robot/environment system.



(a) Simulated run



(b) Real run

Figure 10.11: Location distribution over the map. Corridors: +; Halls: x; Crossings: *

10.9.1 Simulated experiments

Once more the *Stage* simulator was used to simulated the robot and its environment. The criterion to stop the learning process was established in 38 nodes, which was the number of nodes the map converged to when simulating the mapping process with ideal odometry.

Two experiments were carried out in the simulator:

- Ideal odometry (GPS). Figure 10.13 shows the journey together with the spatial node configuration learnt whereas Figure 10.14 shows the results of the mapping and localisation process, and thus the identified set of nodes over time. The mapping process lasted about 1100 seconds and the fact that no error occurred during the localisation phase

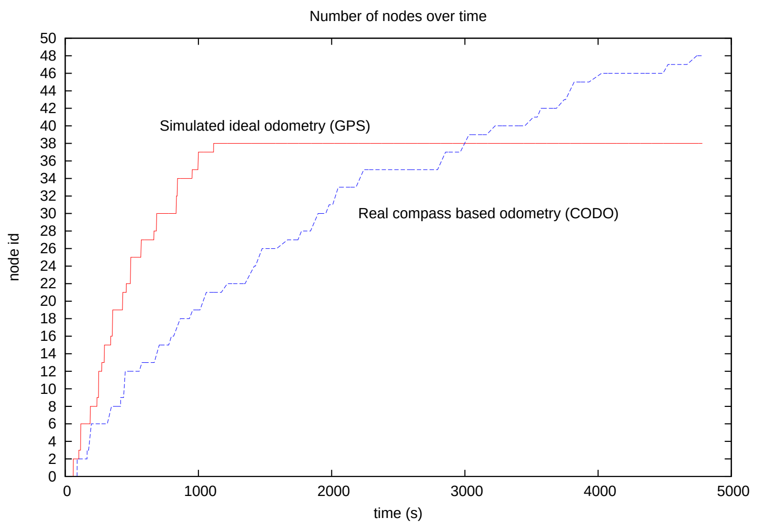


Figure 10.12: Comparison with ideal odometry

(seconds 1100-12000) confirmed that INCA is a valid approach also for localisation.

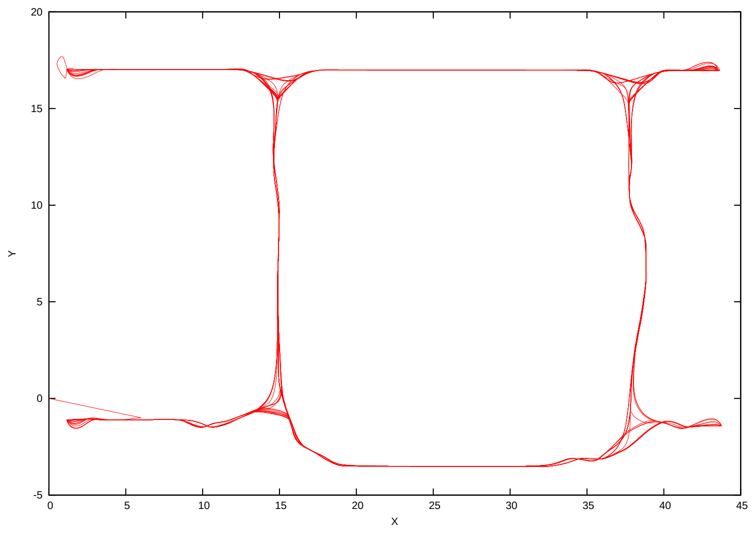
Once the map has been generated, the trajectory of the robot is randomly decided at run-time. The resulting unpredictability means that instead of following a static route, the robot will randomly select the orientation at each junction. As a consequence, the robot does not produce repeatable sequences of nodes in the path, but the probability that it will revisit the whole set of nodes increases.

Table 10.3 shows several path patterns that can be identified from data plotted in Figure 10.14, together with the associated node sequence, the time interval and the label used in the plot to represent each pattern.

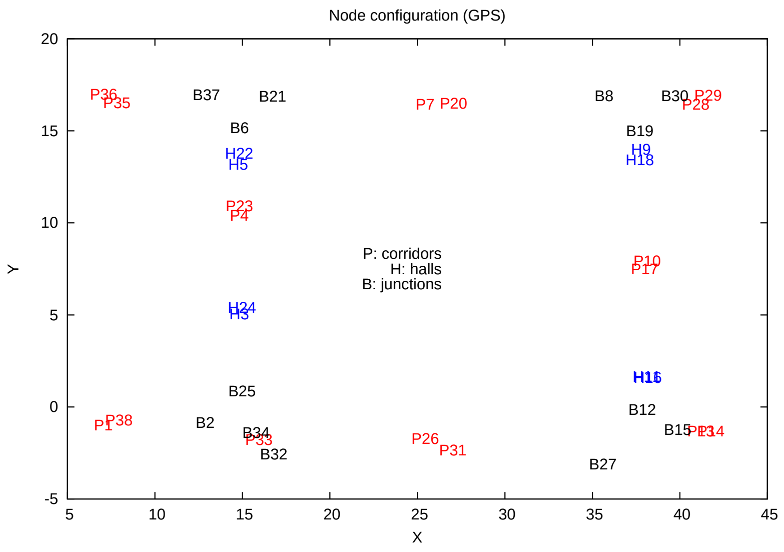
- Laser corrected odometry (LODO). A new experiment was conducted applying the default odometry error value defined in *Player/Stage* and applying the LODO driver to correct the odometry. Figure 10.15 shows the journey together with the spatial node configuration learnt whereas Figure 10.16 shows the results of the mapping and localisation process, and thus the identified set of nodes over time.

  Table 10.4 shows the path patterns extracted from plot in Figure 10.16(a). Again, their associated node sequence, the time interval and the label used in the plot to represent each pattern is included.

  The identified patterns concentrate in the first part of the plot (seconds 2000 to 12000). As times goes by the extracted paths are shorter due



(a) Path corresponding to perfect odometry



(b) Obtained map

Figure 10.13: *Stage* (GPS): robot's path and the obtained map

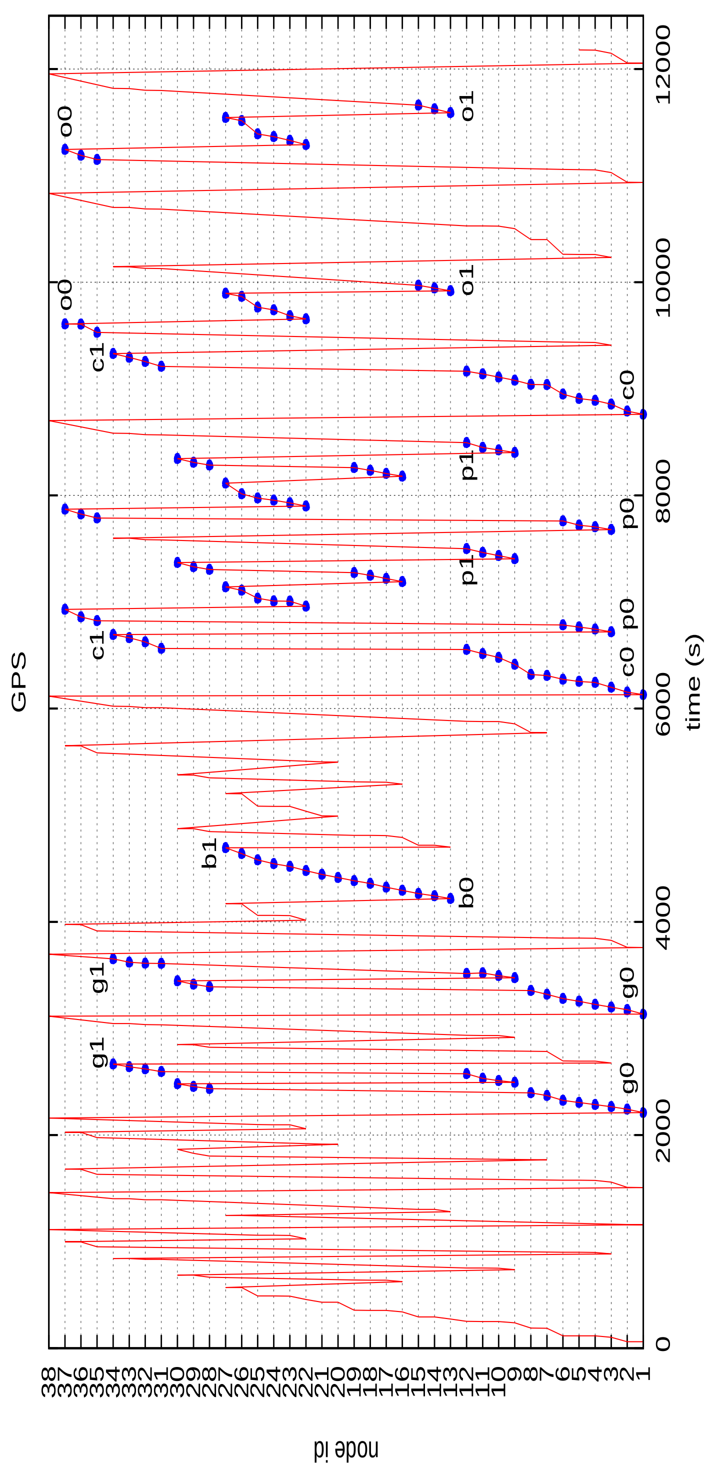
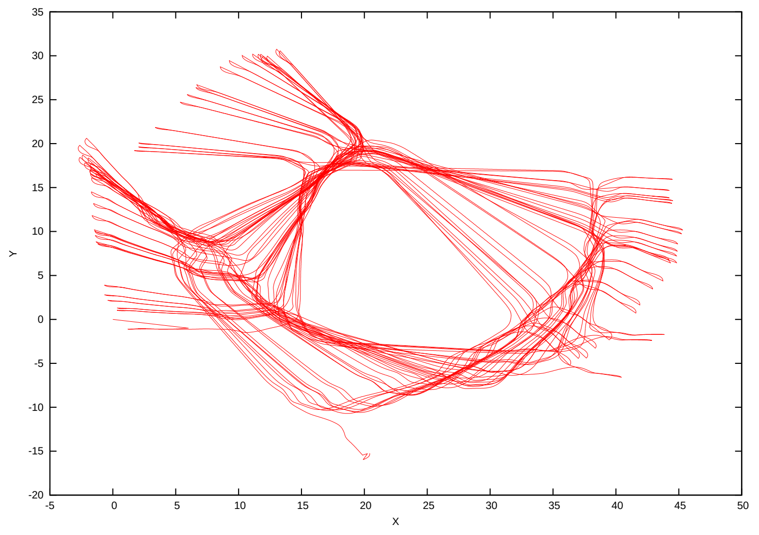
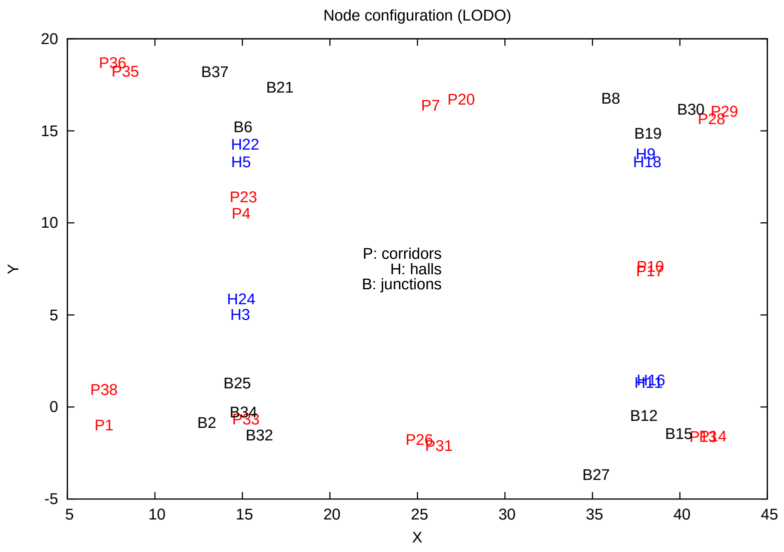


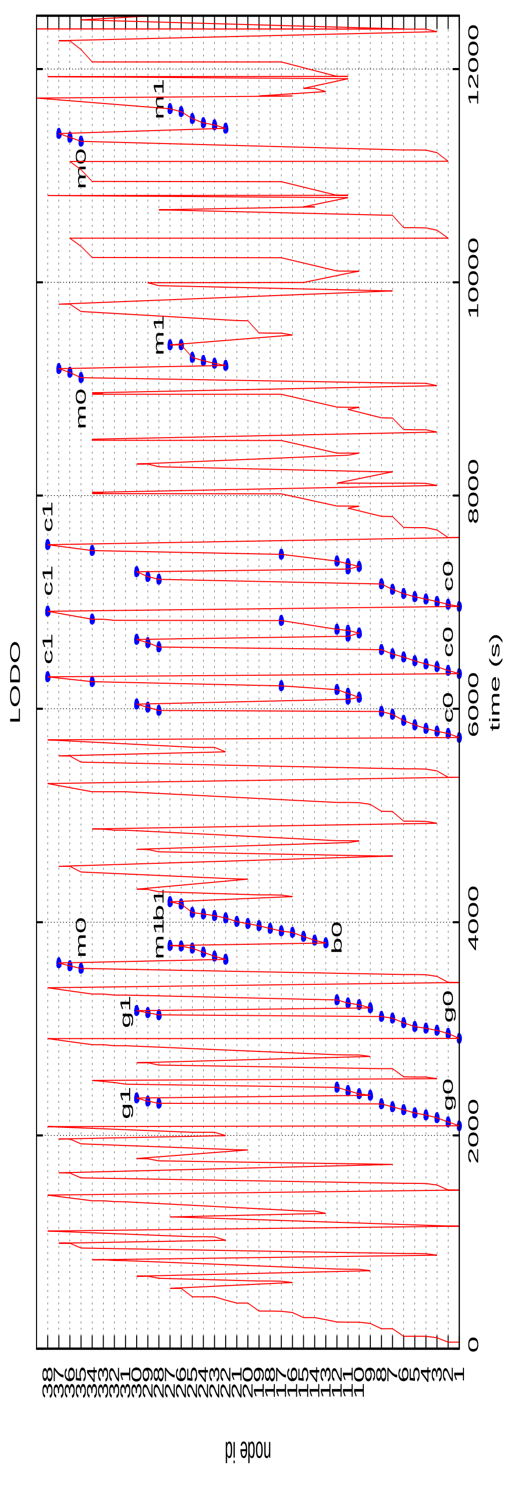
Figure 10.14: *Stage* (GPS): node identification over time


(a) Path corresponding to LODO

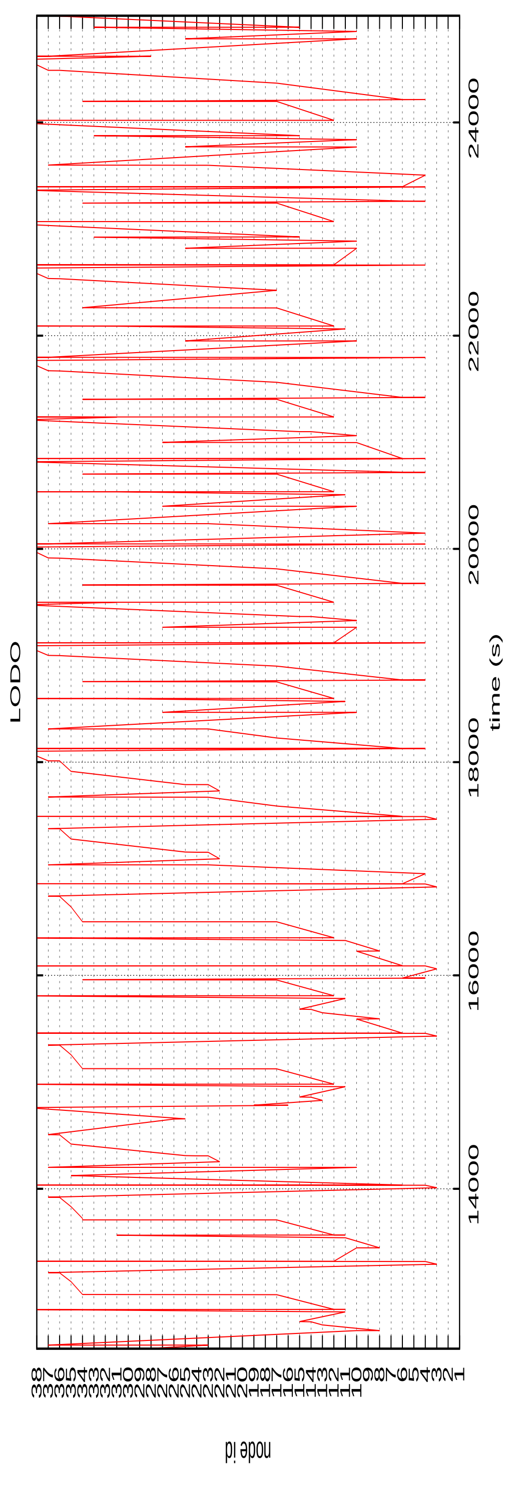


(b) Obtained map

Figure 10.15: *Stage* (LODO): robot's path and the obtained map



(a)



(b)

Figure 10.16: *Stage* (LODO): node identification over time

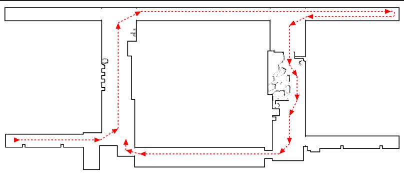
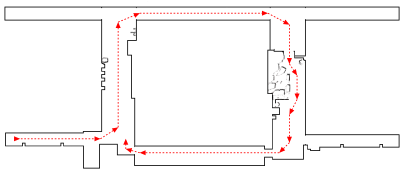
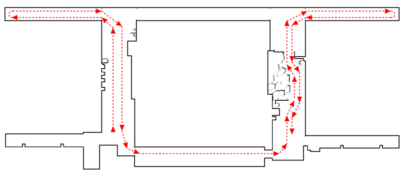
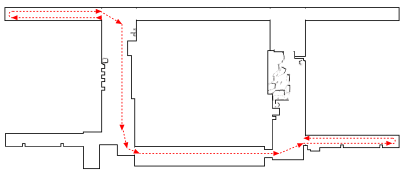
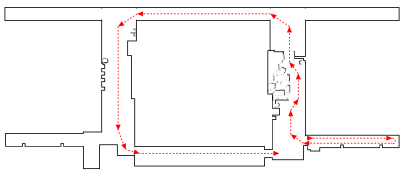
Robot's path	node sequence	time stamp	label
	P1, B2, H3, P4, H5, B6, P7, B8, P28, P29, B30, H9, P10, H11, B12, P31, B32, P33, B34	2210 - 2620, 3174 - 3622	g0-g1
	P1, B2, H3, P4, H5, B6, P7, B8, H9, P10, H11, B12, P31, B32, P33, B34	6193 - 6661, 8791 - 9323	c0-c1
	H3, P4, H5, B6, P35, P36, B37, H22, P23, H24, B25, P26, B27, H16, P17, H18, B19, P28, P29, B30, H9, P10, H11, B12	6734 - 7447, 7675 - 8428	p0-p1
	P35, P36, B37, H22, P23, H24, B25, P26, B27, P13, P14, B15	9529 - 9968, 11154 - 11798	o0-o1
	P13, P14, B15, H16, P17, H18, B19, P20, B21, H22, P23, H24, B25, P26, B27	4218 - 4690	b0-b1

Table 10.3: GPS: extracted path patterns

to localisation failures and the task becomes extremely difficult from second 12000 and there on. Although an odometry correction method is applied, the accumulating error severely affects the localisation of the robot. The type of error remaining after the LODO correction procedure produces a rotation on the robot's trajectory (see for instance Figure 10.15(a)) and thus, a misclassification of nodes with different orientations assigned. This effect was detected in the sequences labelled as $c0 - c1$ in Table 10.4. Chain P17, B34, P38 should have been P31, B34, P38. Node P31 with assigned orientation SN was misclassified as node P17 with assigned orientation WE.

Note that both procedures produced the same configuration of nodes, 17 corridors, 13 crosses and 8 halls, although their positional information

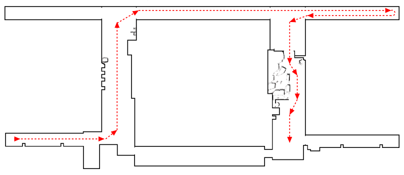
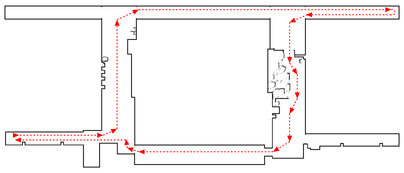
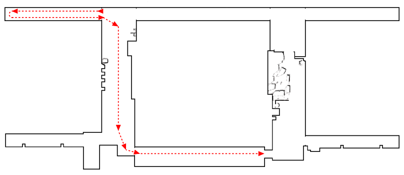
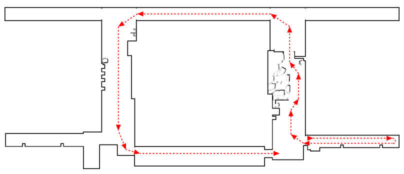
Robot's path	node sequence	time stamp	label
	P1, B2, H3, P4, H5, B6, P7, B8, P28, P29, B30, H9, P10, H11, B12	2128-2391, 2955-3227	g0-g1
	P1, B2, H3, P4, H5, B6, P7, B8, P28, P29, B30, H11, P10, H11, B12, P17*, B34, P38	5767-6217, 6301-6916, 6981-7606	c0-c1
	P35, P36, B37, H22, P23, H24, B25, P26, B27	3565-3781, 9130-9413, 11322-11627	m0-m1
	P13, P14, B15, H16, P17, H18, B19, P20, B21, H22, P23, H24, B25, P26, B27	3821-4169	b0-b1

Table 10.4: LODO: extracted path patterns (*: localisation error)

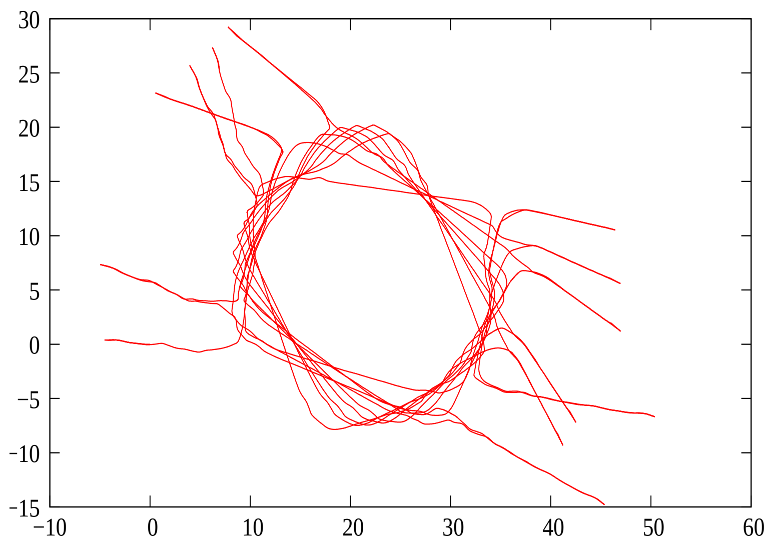
differed due to odometry values.

10.9.2 Experiments in the real robot/environment system

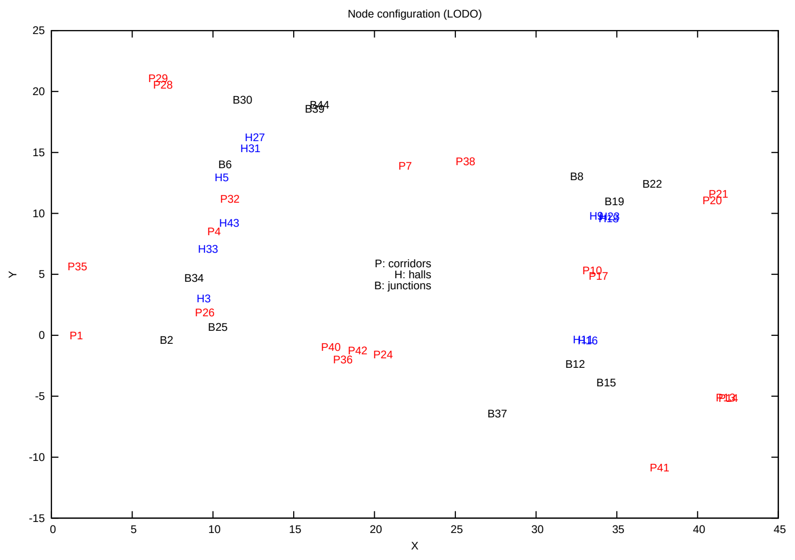
A second set of experiments were carried out with the real robot. This time the node threshold was established in 44.

Figure 10.17 shows the robot's path and the obtained node distribution using laser corrected odometry values (LODO). Figure 10.18 shows the robot's path and the obtained node distribution using compass corrected odometry values (CODO). And Figure 10.19 shows the mapping process, and the localisation over time for both. LODO and CODO.

The results were disappointing but confirmed what the simulated experiments showed for the LODO case. Although the robot localises properly for about 2000 seconds, afterwards the localisation starts to degrade. It is not possible to extract valid path patterns from the plots in Figure 10.19. Both the LODO and CODO methods are insufficient for long term localisation.

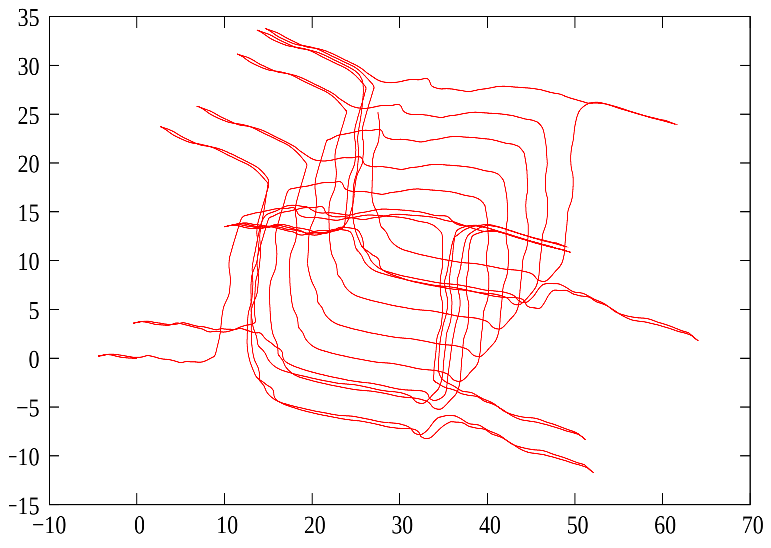


(a) Path corresponding to LODO

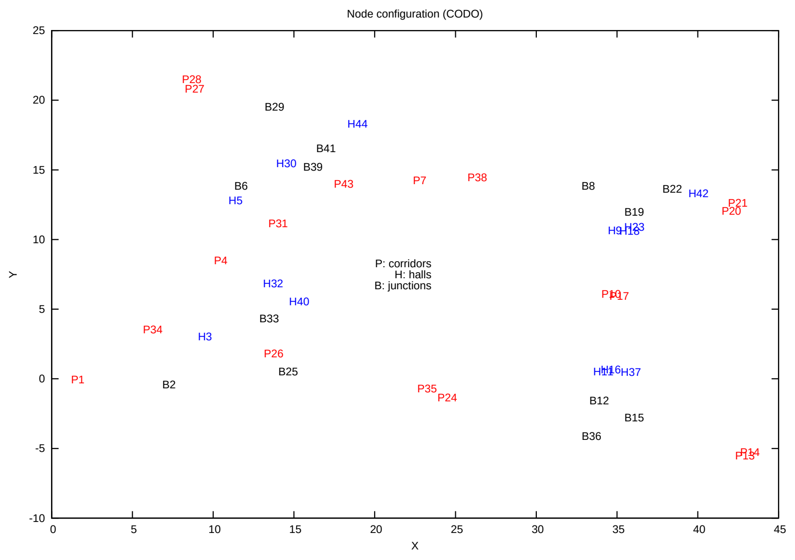


(b) Obtained map

Figure 10.17: Tartalo (LODO): robot's path and the obtained map

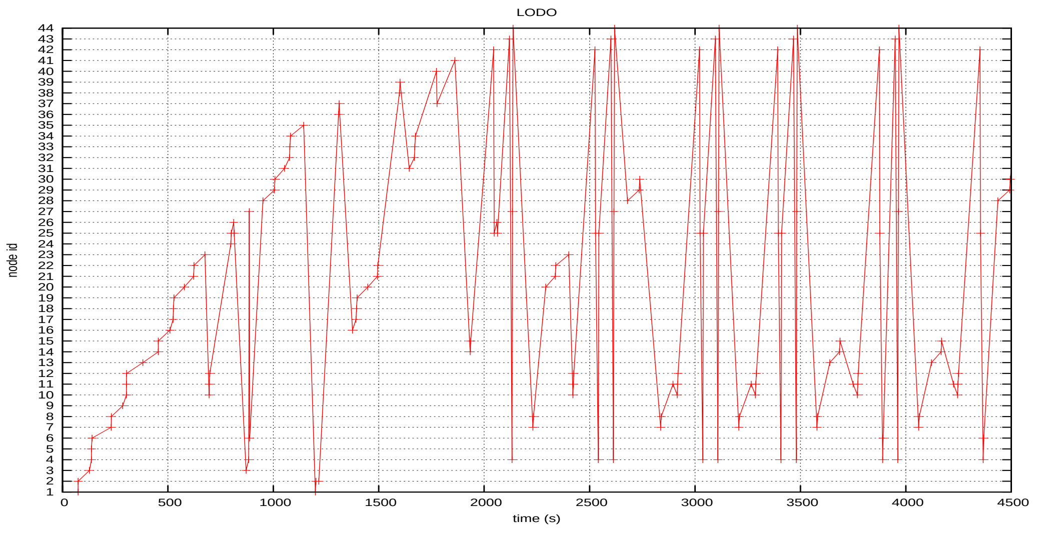


(a) Path corresponding to compass odometry

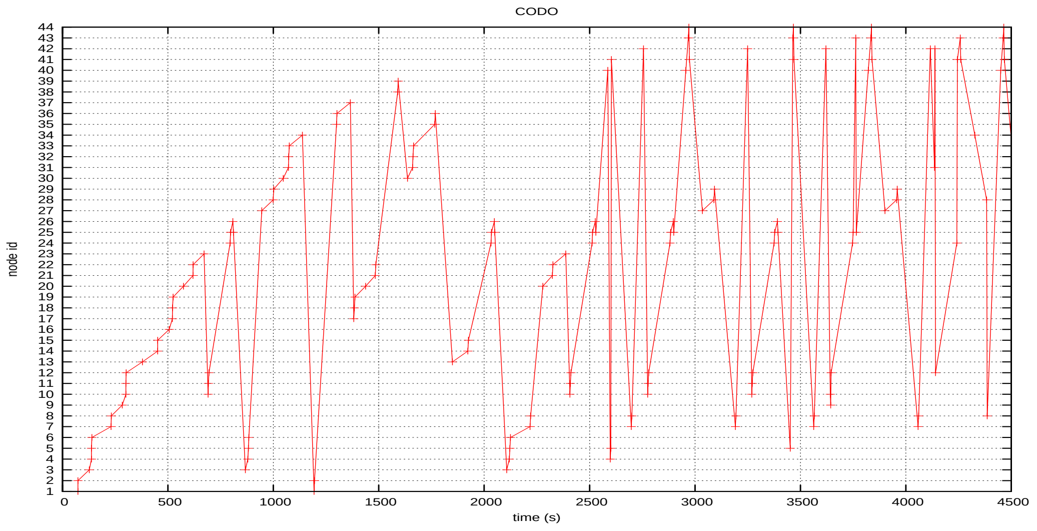


(b) Obtained map

Figure 10.18: Tartalo (CODO): robot's path and the obtained map



(a) Laser corrected odometry



(b) Compass corrected odometry

Figure 10.19: Tartalo(LODO and CODO): node identification over time

Looking at the robot's paths drawn in Figures 10.17(a) and 10.18(a), it can be stated that:

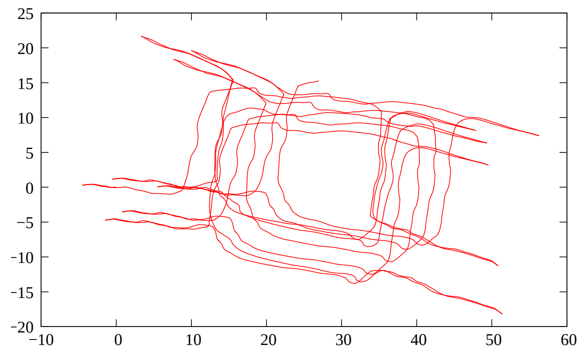
- When using the LODO correction method, the error accumulates more slowly but the error occurs in x, y and θ coordinates. According to LODO odometry, the path rotates over time. While the error is maintained within a certain range the rotation angle is small and the localisation process works correctly. Afterwards, and due to the high dependency the approach has in nominal orientations the system starts to fail and no correspondences are found.
- When using the CODO correction method, only x and y values are affected. θ value is obtained from an absolute reference value and hence, the error is not accumulative. This produces a diagonal shift on the drawn path over time. This shift led to the misclassification of the lower corridors as if they were the upper ones. Oddly, the upper corridors were always well identified.

An intuitive way of coping with this problem is to modify the positional values of the nodes each time they are revisited. Instead of keeping the acquired node information unaltered, during the localisation phase the contents of the nodes can be updated when a positive match occurs.

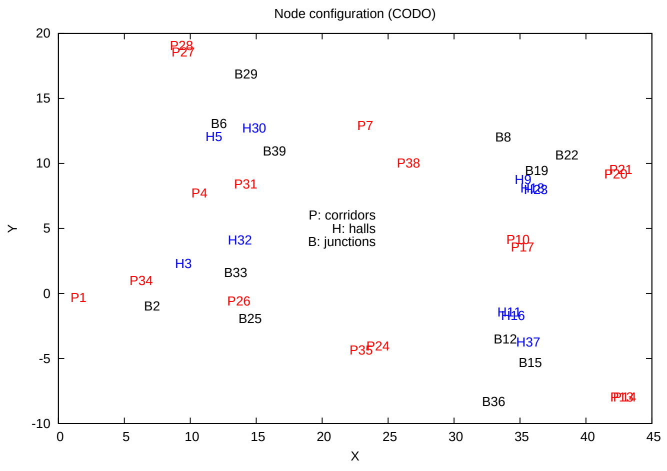
A last experiment was performed with the robot using CODO to correct the odometry to measure the effect of updating the contents of the node. This choice was made because of the lack of accumulated error in orientation values. Figure 10.20 shows the acquired map after reaching the maximum number of nodes (established in 39). The different scales of these two maps reflect the magnitude of the accumulated error in the x and y coordinates over time. Figure 10.21 shows the evolution of the localisation system over time.

Table 10.5 shows the path patterns extracted from plot in Figure 10.21. Again, their associated node sequence, the time interval and the label used in the plot to represent each pattern is included.

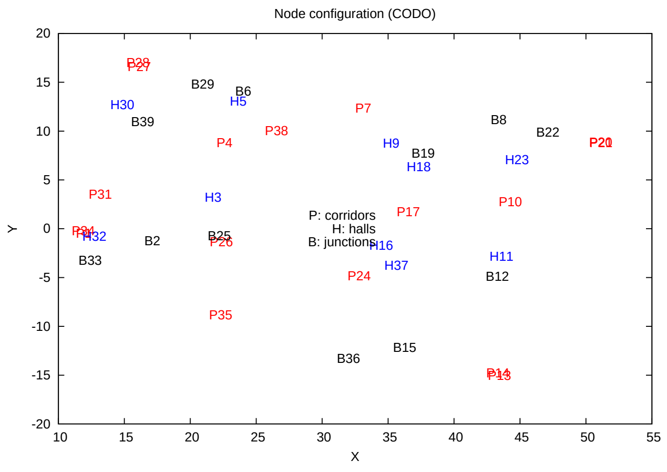
The localisation process last until the robot run out of batteries and only one location was misclassified. As mentioned in Section 10.5, some parameters need to be adjusted for INCA to function properly. The value u_M deeply influences the acceptable deviations from nodes' (x, y) locations. A small u_M value produces failures on loop-closings because of the odometry error. On the contrary, setting u_M to a high value produces that close areas with the same signatures remain indistinguishable. This effect was detected once during the last localisation experiment carried out. Node H30 was wrongly identified as node H32 and thus, the sequence H32, P31, H32 of



(a) Robot's path



(b) Original map



(c) Upgraded map

Figure 10.20: Tartalo (CODO with adaptive node location): robot's path and the obtained map

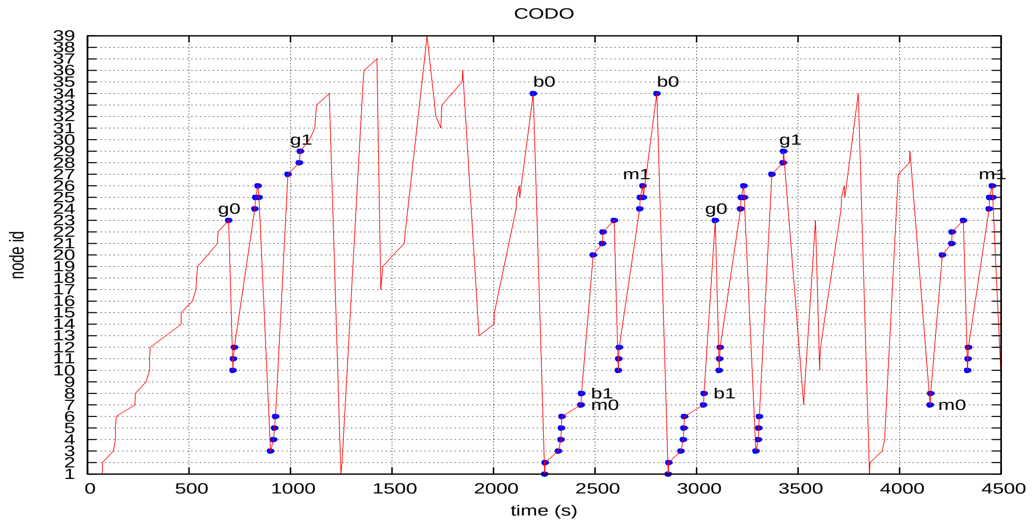


Figure 10.21: Tartalo (CODO with adaptive node location): node identification over time

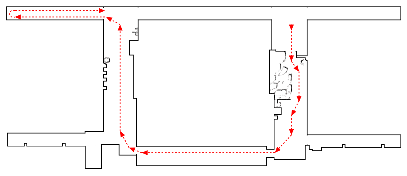
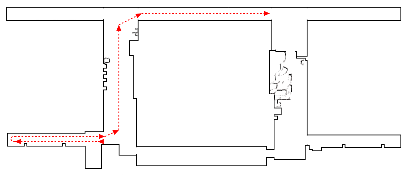
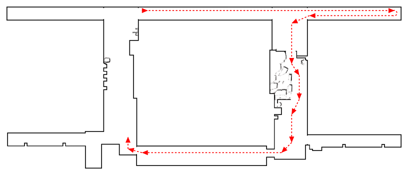
Robot's path	node sequence	time stamp	label
	H23, P10, H11, B12, P24, B25, P26, B25, H3, P4, H5, B6, P27, P28, B29	643-1048, 3092-3428	g0-g1
	P34, P1, B2, H3, P4, H5, B6, P7	2195-2430, 2804-3034	b0-b1
	P7, B8, P20, P21, B22, H23, P10, H11, B12, P24, B25, P26, B25	2430-2737, 4150-4459	m0-m1

Table 10.5: CODO: extracted path patterns

time stamp 1700 should have been H30, P31, H32. Notice that nodes H30 and H32 are separated by a short corridor labelled as P31.

10.10 Conclusions

In this chapter a new approach for incremental topological map construction was presented. A statistical test called INCA was used to this end, combined with a data sampling approach which decided if a topological node found by the robot had already been visited by it. The method was integrated in a behaviour-based control architecture and tested also for localisation purposes.

To measure the adequateness of the approach the map acquisition was performed non-stop until the robot run out of batteries. Afterwards, the experiments were repeated but once the number of nodes in the map reached a given threshold, the learning step was finished and the acquired map was used for localisation purposes.

However, INCA also suffers from odometry error. Of the two error correction methods used in the present work, LODO and CODO, compass corrected odometry was better suited for the developed navigation approach. A last experiment was carried out using CODO and modifying the contents of the acquired nodes each time a location was revisited. The type of error remaining after CODO facilitated the upgrade of the nodes' locations and improved drastically the localisation process.

The experiments conducted confirmed INCA based mapping and localisation as a valid approach and that BB systems can be provided with automatic map acquisition mechanisms.

Part V

Conclusions

Chapter 11

Conclusions and further work

Contents

11.1 Conclusions

The contributions of the present research work can be categorised into two main areas:

1. Development of a visual behaviour for door handle identification.
2. Development of a distributed probabilistic localisation method and a typicality based mapping system for topological procedural maps.

11.1.1 Door identification behaviour

The area of mobile robotics needs general methods for several reasons. The use of algorithms specifically designed to work in a concrete environment made the comparison among methods as well as the application of the approaches in different robot/environment systems difficult.

Part III summarised several approaches aimed at identifying doors for robot navigation purposes. Environmental specific methods, which showed a good performance, were not easily applicable to other environments because they demanded the development of environment-specific segmenters. By contrast, feature extraction methods, which were easily applicable since they only required building a reference database containing the objects to be identified, had a poor classification performance; and the computational payload needed to process single images made them unsuitable for effective/-efficient robot navigation.

A new two step algorithm was presented based on feature extraction that aimed at tuning the extracted features to reduce the superfluous keypoints to be compared at the same time that it increased its efficiency by improving accuracy and reducing the computational time. Contrary to the segmentation based method, the two-step feature extraction method can easily be generalised to other types of handles or even more, to other type of objects such as road signals as shown in the generalisation experiments described in Sections 8.6 and 8.7.

The ROI extraction step improved handle identification procedure and depending on the ROI size, the computational time to classify an image was considerably reduced. The system showed a very low tendency to give false positives while providing a robust identification.

The developed two-step feature extraction based algorithm outperforms the performances obtained without extracting the ROIs, and experiments carried out in a real robot-environment system showed the adequateness of the approach.

11.1.2 Distributed probabilistic localisation and typicality based mapping

Part IV described the steps given towards a terrain inspection bioinspired navigation system to overcome two of the main limitations of Galtxagorri's navigation architecture.

A preliminary approach has been proposed to merge the Markovian localisation in a distributed system. The proposed approach requires low storage and computational resources and is, in the author's opinion, more adequate to be applied in dynamic environments. The odometric information included in the nodes, i.e. the redefinition of the location identifier helps to accommodate a common action model used in probabilistic approaches and could be easily acquired and managed within an exploration strategy as it has been shown by the a posteriori mapping experiments.

On the one hand, the topological quality of the system makes it less sensitive to odometry errors. On the other hand, its procedural nature allows to maintain the basic functions of the robot at low cost allowing at the same time the performance of higher level tasks.

The second contribution to terrain inspection level navigation involved the development of an automatic mapping method for acquiring the procedural topological map. A new approach to incrementally construct the topological map was presented setting out the problem as a typicality one. A statistical test called INCA was used to this end, combined with a data sampling approach which decides if a topological node found by the robot had already been visited by it. The method was integrated in a behaviour-based control architecture and tested in both simulated and real robot/environment systems. Afterwards the same system was used for localisation purposes, and the suitability of the approach was confirmed.

11.2 Further work

The developed work is amenable to improvements and further development.

11.2.1 Feature extraction based two step algorithm

Obviously, the proposed two-step algorithm is opened to any other feature extraction method. But generally speaking, the following pending tasks are identified:

- The keypoint matching criteria has to be analysed more deeply. More sophisticated and efficient algorithms remain to be tested and the performance of different distance measures still needs to be studied.

- The extension of the proposed method to other applications such as face recognition should be considered.
- The development of new multiclassifiers for the identification of interesting places and locations that would help the robot in its navigation task.

11.2.2 Procedural based localisation and mapping

The steps given towards a complete behaviour-based navigation frame can also be improved:

- Although the action model applied in the distributed Markov localisation approach showed to be adequate, different schemes should be tested and compared.
- Odometry correction methods can also be applied such as the compass-based odometry correction or the laser stabilised odometry correction tested during the automatic mapping experimental phase. Also, a Kalman filter could be applied to each node's position identifier vector to cope with the positioning error.
- During the development of the present work the author also researched other probabilistic models such as Bayesian Networks. These models could be used as probability maintenance systems for the set of nodes of the localisation process. This problem is going to be considered in the near future.
- To improve the efficiency of the automatic map acquisition system, when looking for correspondences the use of their associated probability value should be studied.
- The criteria for stopping the learning process, i.e. the maximum number of nodes should be revised. Given that it is not possible to know a priori the number of nodes, the map should be closed when no more alternative ways remain unvisited in the junction nodes.
- Some aspects of the implementation of INCA should be improved and more experiments should be conducted in a systematic manner in order to better identify the advantages and drawbacks of the test.

- The probabilistic distributed localisation system should be compared with the INCA based localisation system.

11.2.3 Merging the developed methods in a general navigation framework

If the final goal is to attain a terrain inspection level BB navigation architecture, then the developed methods should be integrated in a common navigation framework. To that end, the following needs are identified:

- New local navigation strategies and several landmark identification modules need to be incorporated to increase the granularity of the environment in order to reach more interesting goals than halls and corridors, such as offices and laboratories. Adding more topological nodes would allow the generalisation of the experiments to different environments, and the comparison with other approaches. The first step should be to integrate the door identification and door crossing modules already developed, and to enrich the behaviour associated to several nodes with door crossing abilities, and a wall following behaviour. These two modules would help to cover the perimeter of small rooms and improve the exploration strategy.
- Handles could also be helpful for localisation purposes. This could be managed in two ways:
  - Add the door identification as a landmark identification subsystem. The node signature could be enriched with the number of door handles in the area limited by the node. Moreover, each handle could have also its position identifier. This information could also be provided to INCA in order to improve the mapping and the a posteriori localisation system.
  - Add the doors as topological places. This could help increase the granularity of the map although it would be at the expense of loosing the "qualitative" and procedural nature of the followed approach.
- Nothing has been said about planning. Up to now, the proposed modifications were tested using an exploration strategy. The overall map should be used for commanding the robot to fulfil a concrete goal and thus, to reach concrete locations. This process was included in Galtxagorri's original control architecture but it should be further tested and improved to cope with higher resolution topological representations.

11.3 Publications

The research work described produced some publications which are listed below.

- Contributions to robot navigation:

    1. **Environment representation by behavior decomposition**
    E. Jauregi, A. Astigarraga, E. Lazkano, B. Sierra, J. M. Martínez-Otzeta
    WAF06: VII Workshop de agentes físicos, I, ISBN: 85-689-8115-X, 101-108 (**2006**).
    2. **Búsqueda del dispositivo adecuado para la identificación de cruces en entornos indoor**
    E. Jauregi, E. Lazkano, B. Sierra, A. Astigarraga, J. M. Martínez-Otzeta and M. Ardaiz
    Jornadas de Automática XXVII (**2006**).
    3. **Distributed Markov Localisation for Probabilistic Behaviour Activation**
    E. Jauregi, E. Lazkano, B. Sierra, A. Astigarraga, J. M. Martínez-Otzeta and Y. Yurramendi
    ICARA: 3rd International Conference on Autonomous Robots and Agents, *I*, 297-301 (**2006**).
    4. **Behavior-based localization using probabilistic triggering**
    E. Jauregi, E. Lazkano, B. Sierra
    TAROS: Towards Autonomous Robotic Systems, *I, ISSN: 2041-6407*, 157-164 (**2009**).
    5. **Robot Mapping based on Typicality**
    E. Jauregi, I. Irigoien, B. Sierra, E. Lazkano and C. Arenas.
    Intelligent Autonomous Systems (IAS11) (**2010**).
    6. **Loop-Closing: a Typicality Approach**
    E. Jauregi, I. Irigoien, B. Sierra, E. Lazkano, C. Arenas
    Robotics and Autonomout Systems, 218-227 (**2011**).

- Contributions to computer vision:

    1. **Adapting the point of view for behavior based navigation**
    M. Ardaiz, A. Astigarraga, E. Lazkano, B. Sierra, J. M. Martínez-Otzeta, E. Jauregi
    Springer-Verlag, *LNAI 4177*, 69-78 (**2006**).

2. **Door Handle Identification: a Three-Stage approach**
   E. Jauregi, J. M. Martínez-Otzeta, B. Sierra and E. Lazkano
   IAV: 6th International Conference on Intelligent Autonomous Vehicles (**2006**).
3. **Handle identification by keypoint extraction**
   E. Jauregi, J. M. Martínez-Otzeta, B. Sierra and E. Lazkano
   CAEPIA-TTIA: Asociación Española para la Inteligencia Artificial, II 21-30 (**2007**).
4. **Tartalo: the door knocker robot**
   E. Jauregi, J. M. Martínez-Otzeta, B. Sierra and E. Lazkano
   ROBIO: IEEE International Conference on Robotics and Biomimetics, I, ISBN: 978-1-4-4244-1758-2 1114-1120 (**2007**).
5. **Visual approaches for handle identification**
   E. Jauregi, E. Lazkano, J. M. Martínez-Otzeta and B. Sierra
   EUROS: European Robotics Symposium, I, ISSN: 1610-7438 313-322 (**2008**).
6. **Object recognition using region detection and feature extraction**
   E. Jauregi, E. Lazkano and B. Sierra
   TAROS: Towards Autonomous Robotic Systems, I, ISSN: 2041-6407 104-111 (**2009**).
7. **Approaches to door identification for robot navigation**
   E. Jauregi, E. Lazkano and B. Sierra
   INTECH: Mobile Robots Navigation, ISBN: 978-953-7619-X-X 241-261 (**2010**).

- Contributions to machine learning:

1. **Genetically Searched Classifier Hierarchies for Surface Identification**
   J. M. Martínez-Otzeta, B. Sierra, E. Lazkano, E. Jauregi
   European Conference in Artificial Intelligence: workshop on evolutionary computation., 41-45 (**2006**).
2. **On a unified framework for sampling with and without replacement in decision tree ensembles**
   J. M. Martínez-Otzeta, B. Sierra, E. Lazkano, E. Jauregi
   Springer-Verlag: Lecture Notes in Artificial Intelligence., *LNAI-4183*, 118-127 (**2006**).
3. **Naive+Naive=Smart Bayes?**
   J. M. Martínez-Otzeta, B. Sierra, E. Lazkano, E. Jauregi and Y.

Yurramendi
ICARA: 3rd International Conference on Autonomous Robots and Agents., *I*, 291-295 (**2006**).

4. **Analyzing classifier hierarchy multiclassifier learning**
J. M. Martínez-Otzeta, B. Sierra, E. Lazkano, E. Jauregi and Y. Yurramendi
CIARP: 13th Iberoamerican Congress on Pattern Recognition, *LNCS 5197*, 775-782 (**2008**).

5. **Histogram Distance Based Bayesian Network Structure Learning: a Supervised Classification Specific Approach**
B. Sierra, E. Lazkano, E. Jauregi, and I. Irigoien
Journal of Decision Support Systems, *48*, 180-190 (**2009**).

6. **K Nearest Neighbor Equality: giving equal chance to all existing classes**
B. Sierra, E. Lazkano, E. Jauregi, and Y. Yurramendi
Journal of Information Science (**Accepted**)

References

Aeronautics, N. and Administration, S. (2004). Mars exploration rover. Technical Report MSU-CSE-00-2, Jet Propulsion Laboratory (California Institute of Technology), Pasadena, CA 91109.

Arenas, C. and Cuadras, C. M. (2002). Some recent statistical methods based on distances. *Contributions to Science*, 2:183–191.

Arkin, R. (1989). Motor schema-based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112.

Arkin, R. C. (1998). *Behavior-based robotics*. MIT Press.

Astigarraga, A., Lazkano, E., Rañó, I., Sierra, B., and Zarautz, I. (2003). SORGIN: a software framework for behavior control implementation. In *CSCS14*, volume 1, pages 243–248.

Ayala-Ramírez, V., Garcia-Capulin, C., Perez-Garcia, A., and Sanchez-Yanez, R. R. (2006). Circle detection on images using genetic algorithms. *Pattern Recognition Letters*, 27(6):652–657.

Ballard, D. H. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2).

Bar-Hen, A. (2001). Preliminary tests in linear discriminat analysis. *Statistica*, 4:585–593.

Barber, R. and Salichs, M. A. (2001). Mobile robot navigation based on event maps. In *Proceedings of Field and Service Robotics*, pages 61–66.

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). SURF: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision*.

Bekey, G. A. (2005). *Autonomous Robots: From Biological Inspiration to Implementation and Control*. The MIT Press.

Bennett, A. T. D. (1997). Do animals have cognitive maps? *The journal of experimental biology*, pages 219–224.

Borenstein, J., Everett, B., and Feng, L. (1996). *Navigating mobile robots: systems and techniques.* A. K. Peters.

Borenstein, J. and Feng, L. (1996). Measurement and correction of systematic odometry errors. *IEEE Transactions on Robotics and Automation*, 12(6):869–880.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology.* Cambridge, MA: MIT Press.

Brooks, C. and K. Iagnemma, K. (2009). Visual detection of novel terrain via two-class classification. In *Proceedings of the ACM symposium on Applied Computing (SAC)*, pages 1145–1150, New York, NY, USA. ACM.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of robotics and automation*, RA–26:14–23.

Burgard, W., Derr, A., Fox, D., and Cremers, A. B. (1998). Integrating global position estimation and position tracking for mobile robots: the dnamic markov localization approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems.*

Cassandra, A. R., Kaelbling, L. P., and Kurien, J. A. (1996). Acting under uncertainty: discrete bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems.*

Chen, C. and Wang, H. (2006). Appearance-based topological bayesian inference for loop-closing detection in a cross-country environment. *International Journal of Robotic Research*, 25(10):953–983.

Chen, S.-H. (1996). Elements of information theory: Tomas m. cover and joy a. thomas, (john wiley & sons, new york, ny, 1991). *Journal of Economic Dynamics and Control*, 20(5):819–824.

Connell, J. (1992). SSS: a hybrid architecture applied to robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2719–2724.

Connell, J. H. (1990). *Minimalist mobile robotics. A colony-style architecture for an artificial creature.* Academic Press, inc.

Cuadras, C. M. (1992). Some examples of distance based discrimination. *Biometrical Letters*, pages 3–20.

Cuadras, C. M. and Fortiana, J. (1995). A continuous metric scaling solution for a random variable. *Journal of Multivariate Analysis*, 32:1–14.

Cuadras, C. M. and Fortiana, J. (2000). The importance of geometry in multivariate analysis and some applications. In *Statistics for the 21st Century*, pages 93–108. Marcel Dekker, New York.

Cuadras, C. M., Fortiana, J., and Oliva, F. (1997). The proximity of an individual to a population with applications in discriminant analysis. *Journal of Classification*, 14:117–136.

Dennet, D. C. (1998). Cognitive wheels: the frame problem of AI. In *Brainchildren. Essays on Designing Minds*, pages 181–205. MIT Press.

Diard, J., Bessière, P., and Mazer, E. (2003). A survey of probabilistic models, using the bayesian programming methodology as a unifying framework. In *Proc. of the International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Singapore (SG).

Druid, A. and Hendler, J. A. (2000). *Robots for kids: exploring new technologies for learning*. Morgan Kauffman Publishers. Academic Press.

Duda, R. and Hart, P. E. (1972). Use of Hough transform to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

Eberset, C., Andersson, M., and Christensen, H. I. (2000). Vision-based door-traversal for autonomous mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 620–625.

Egido, V., Barber, R., Boada, M., and Salichs, M. A. (2004). A planner for topological navigation based on previous experiences. In *Proceedings of 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*.

Everett, H. R. (1995). *Sensors for mobile robots. Theory and applications*. A. K. Peters, Ltd.

Fox, D., Burgard, W., Dellaert, F., and Thrun, S. (1999a). Monte carlo localization: efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 343–349.

Fox, D., Burgard, W., and Thrun, S. (1998). Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(1):195–207.

Fox, D., Burgard, W., and Thrun, S. (1999b). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427.

Fox, D., Thrun, S., Dellaert, F., and Burgard, W. (2000). Particle filters for mobile robot localization. In *Sequential Monte Carlo methods in practice*, pages 470–498. Springer Verlag.

Fraundorfer, F., Engels, C., and Nistér, D. (2007). Topological mapping, localization and navigation using image collections. In *Intelligent Robots and Systems (IROS)*, pages 3872–3877.

Garrido, S., Moreno, L., Abderrahim, M., and Blanco, D. (2009). Fm2: A real-time sensor-based feedback controller for mobile robots. *International Journal of Robotics and Automation*, 24(1):48–65.

Gat, E. (1991). Integrating reaction and planning in a heterogeneous asynchronous architecture for mobile robot navigation. *SIGART Bulletin*, 4(2):46–50.

Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The Player/Stage project: tools for multi-robot and distributed sensor systems. In *Proc. of the International Conference on Advanced Robotics (ICAR)*, pages 317–323.

Gil, A., Reinoso, O., Vicente, A., Fernández, C., and Payá, L. (2005). Monte Carlo localization using SIFT features. *Lecture Notes in Computer Science*, 3522:623–630.

González, J. J. and Jiménez, J. A. (2009). La robótica como herramienta para la educación en ciencias e ingeniería. *Revista Iberoamericana de Informática Educativa*, (10):31–36.

Gonzalez, R. C. and Woods, R. E. (1993). *Digital image processing*. Addison Wesley.

González-Baños, H. H. and Latombe, J. C. (2002). Navigation strategies for exploring indoor environments. *International Journal of Robotics Research*, 21(10–11):829–848.

Gower, J. C. (1985). *Encyclopedia of Statistical Sciences*, volume 5, chapter Measures of similarity, dissimilarity and distance, pages 397–405. John Wiley & Sons, New York.

Greiner, R. and Isukapalli, R. (1994). Learning to select useful landmarks. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1251–1256. AAAI Press/MIT Press.

Grigorescu, C. and Petkov, N. (2003). Distance sets for shape filters and shape recognition. In *IEEE Trans. on Image Processing*, volume 12 (10), pages 1274–1286.

Haralick, R. and Shapiro, L. G. (1992). *Computer and Robotic Vision*. Addison-Wesley.

Harris, C. and Stephens, M. (1988). A combined corner and edge detection. In *Proceedings of The 4th Alvey Vision Conference*, pages 147–151.

Hempstalk, K., Frank, E., and Witten, I. H. (2008). One-class classification by combining density and class probability estimation. In *ECML PKDD '08: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, pages 505–519, Berlin, Heidelberg. Springer-Verlag.

Henthorn, R., Hobson, B., McGill, P., Sherman, A., and Smith, K. (2010). Mars benthic rover: In-situ rapid proto-testing on the monterey accelerated research system. In *OCEANS 2010*, pages 1 –7.

Hogg, D. W., Martin, F., and Resnick, M. (1991). Braitenberg creatures. *MIT Media Lab*, 13.

Horn, B. K. P. (1986). *Robot Vision*. MIT Press.

Horswill, I. (1994). *Specialization of perceptual processes*. PhD thesis, Massachusets Institute of Technology.

Howard, A. (2005). Multi-robot simultaneous localization and mapping using particle filters. In *In Robotics: Science and Systems*, pages 201–208.

Howard, A. and Kitchen, L. (2001). Navigation using natural landmarks. *Robotics and Autonomous Systems*, 26:99–115.

Irigoien, I. and Arenas, C. (2008). INCA: New statistic for estimating the number of clusters and identifying atypical units. *Statistics in Medicine*, 27(15):2948–2973.

Jennings, H. S. (1906). *Behavior of the lower organisms.* The Columbia university pres, New York.

Kalman, R. R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME journal of Basic Engineering*, 82:35–45.

Kaneko, A., Marino, M., and Fukushima, E. (2010). Humanitarian demining robot gryphon: New vision techniques and optimization methods. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 228 –233.

Khatib, O. (1996). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98.

Kimme, C., Ballard, D., and Sklansky, J. (1975). Finding circles by array accumulators. *Communications of the ACM*, 18(2):120–122.

Kortenkamp, D. and Bonasso, R. P., editors (1998). *Artificial intelligence and mobile robotics. Case studies of succesful robot systems.* MIT Press.

Kortenkamp, D., Huber, M., Cohen, C., Raschke, U., Koss, F., and Congdon, C. (1998). Integrating high speed obstacle avoidance, global path planning and vision sensing on a mobile robot. In *Mobile Robots and Artificial Intelligence*, pages 53–71. AAAI Press.

Kragic, D., Petersson, L., and Christensen, H. I. (2002). Visually guided manipulation tasks. *Robotics and Autonomous Systems*, 40(2-3):193–203.

Lan, Z. D. and Mohr, R. (1997). Robust location based partial correlation. Technical report, Institute National de Recherche en Informatique et en Automatique.

Langley, P., MacKusick, K. B., and Allen, J. A. (1991). A design for the ICARUS architecture. *SIGART Bulletin*, 4(2):104–109.

Lankenau, A. and Röfer, T. (2002). Mobile robot self-localization in large-scale environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1359–1364.

Latombe, J. C. (1991). *Robot motion planning.* Kluwer Academic.

Latombe, J. C. (1999). Motion planning: a journey of robots, molecules, digital actors and other artifacts. *International Journal of Robotics Research*, 18(11):1119–1128.

Lazkano, E. (2004). *Pautas en el desarrollo incremental de una arquitectura de control basada en el comportamiento para la navegación de robots en entornos semi-estructurados.* PhD thesis, Euskal Herriko Unibertsitatea/Basque Country University.

Lazkano, E., Astigarraga, A., Jauregi, E., Sierra, B., and Martínez-Otzeta, J. M. (2006a). Environment representation by behavior decomposition. In *Workshop de Agentes físicos*, pages 101–108.

Lazkano, E., Astigarraga, A., and Sierra, B. (2004a). Postrob: a behavior-based post deliverer. In *Towards Autonomous Robotics Systems (TAROS)*, volume 1. Springer-Verlag UK. ISSN 1744–8050.

Lazkano, E., Astigarraga, A., Sierra, B., and Rañó, I. (2004b). On the adequateness of emergency exit panel and corridor identification as pilot scheme for a mobile robot. In *Intelligent Autonomous Systems 8*, volume 1, pages 915–924.

Lazkano, E., Sierra, B., Astigarraga, A., and Martínez-Otzeta, J. M. (2006b). On the use of bayesian networks to develop behavior for mobile robots. *Robotics and Autonomous Systems*, 55(3):253–265.

Ledwich, L. and Williams, S. (2004). Reduced SIFT features for image retrieval and indoor localisation. In *Australian Conference on Robotics and Automation.*

Lee, D. (1996). *The Map-Building and Exploration of a Simple Sonar-Equipped Mobile Robot.* Cambridge University Press.

Leonard, J. J. and Durrant-Whyte, H. F. (1992). *Directed sonar sensing for mobile robot navigation.* Kluwer Academic Publishers, Cambrage, MA.

Levitt, T. S. and Lawton, D. T. (1990). Qualitative navigation for mobile robots. *Artificial Intelligence*, 44(3):305–360.

Li, W., Christensen, H. I., and Orebäck, A. (2004). An architecture for indoor navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1783–1788.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, Corfu, Greece.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–100.

Maes, P. (1989). The dynamic of action selection. In *Proceedings of the 1989 International Joint Conference on Artificial Intelligence, Detroit*, pages 991–997.

Mallot, H. A. and Franz, M. A. (2000). Biomimetic robot navigation. *Robotics and Autonomous System*, 30:133–153.

Manevitz, L. and Yousef, M. (2007). One-class document classification via neural networks. *Neurocomput.*, 70(7-9):1466–1481.

Manevitz, L. M. and Yousef, M. (2002). One-class SVMs for document classification. *J. Mach. Learn. Res.*, 2:139–154.

Matarić, M. (1990). A distributed model for mobile robot environment-learning and navigation. Master's thesis, MIT Artificial Intelligence Laboratory.

Matarić, M. (1992). A distributed model for mobile robot environment–learning and navigation. Technical Report MIT-AITR 1228, M.I.T.

Matarić, M. J. (2009). *The Robotics Primer*. MIT Press.

Mayr, O. (1975). *The origins of feedback control*. MIT Press.

McDonald, L. L., Lowe, V. W., Smidt, R. K., and Meister, K. A. (1976). A preliminary test for discriminant analysis based on small samples. *Biometrics*, 32:417–422.

Meyer, J.-A. (1997). From natural to artificial life: biomimetic mechanisms in animat design. *Robotics and Autonomous Systems*, 22:3–21.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630.

Minguez, J. and Montano, L. (2004). Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios. *IEEE transactions in Robotics and Automation*, 1(20):45 – 59.

Monasterio, I., Lazkano, E., Rañó, I., and Sierra, B. (2002). Learning to traverse doors using visual information. *Mathematics and Computers in Simulation*, 60:347–356.

Moravec, H. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical Report CMU-RI-TR-80-03.

Muñoz-Salinas, R., Aguirre, E., and García-Silvente, M. (2005). Detection of doors using a genetic visual fuzzy system for mobile robots. Technical report, University of Granada.

Murthy, S. K., Kasif, S., and Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–33. ftp://blaze.cs.jhu.edu/pub/oc1.

Nehmzow, U. (1999). *Mobile robotics: a practical introduction.* Springer.

Nehmzow, U. and McGonigle, B. (1993). Robot navigation by light.

Nehmzow, U. and Owen, C. (2000). Experiments with manchester's fourty two in unmodified large environments. *Robotics and Autonomous Systems*, 33:223–242.

Nourbakhsh, I. (1998). Dervish: an office-navigating robot. In Kortenkamp, D., Bonassi, R. P., and Murphy, R., editors, *Artificial Intelligence and Mobile Robots. Case Studies of Succesful Robot Systems*, pages 73–90. The AAAI Press. MIT Press.

Olson, E. (2009). Recognizing places using spectrally clustered local matches. *Robotics and Autonomous System*, 57:1157–1172.

Owen, C. and Nehmzow, U. (1998). Landmark-based navigation for a mobile robot. In *Simulation of Adaptive Behavior*.

Paz, L. M., Tardós, J. D., and Neira, J. (2008). Divide and conquer: EKF SLAM in O(n). *IEEE Transactions on Robotics*, 24(5).

Pfeifer, R. and Bongard, J. (2006). *How the body shapes the way we think. A new view of intelligence.* MIT Press.

Primdahl, K., Katz, I., Feinstein, O., Mok, Y., Dahlkamp, H., Stavens, D., Montemerlo, M., and Thrun, S. (2005). Change detection from multiple camera images extended to non-stationary cameras. In *Proceedings of Field and Service Robotics*, Port Douglas, Australia.

Rao, C. R. (1962). Use of discriminant and allied functions in multivariate analysis. *Sankhya-Serie A*, 24:149–154.

Rao, C. R. (1982). Diversity: its measurement, decomposition, apportionment and analysis. *Sankhyā. The Indian Journal of Statistics, Series A*, 44:1–22.

Rekleitis, I. M. (2004). A particle filter tutorial for mobile robot localization. Technical Report TR-CIM-04-02, Centre for Intelligent Machines, McGill University, 3480 University St., Montreal, Québec, CANADA H3A 2A7.

Roduner, C. and Rohs, M. (2006). Practical issues in physical sign recognition with mobile devices. In Strang, T., Cahill, V., and Quigley, A., editors, *Workshop on Pervasive Mobile Interaction Devices, (PERMID)*, pages 297–304, Dublin, Ireland.

Rosenblatt, J. K. (1995). DAMN: A distributed architecture for mobile navigation. In *Proc. of the AAAI Spring Symp. on Lessons Learned from Implememted Software Architectures for Physical Agents*, pages 167–178, Stanford, CA.

Roy, N. and Thrun, S. (1999). Coastal navigation with mobile robots. In *In Advances in Neural Processing Systems 12*.

Rusu, R. B., Meeussen, W., Chitta, S., and Beetz, M. (2009). Laser-based Perception for Door and Handle Identification. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, Munich, Germany. Best paper award.

Sánchez Martín, F. M., Jiménez Schlegl, P., Millán Rodríguez, F., Salvador Bayarri, J., Monllau Font, V., Palou Redorta, J., and Villavicencio Mavrich, H. (2007). Historia de la robótica: de Arquitas de Tarento al Robot da Vinci. (parte II). *Actas urológicas españolas*, (31(3)):185–196.

Sánchez-Yáñez, R. E., Kurmyshev, E. V., and Fernández, A. (2003). One-class texture classifier in the CCR feature space. *Pattern Recognition Letters*, 24(9-10):1503–1511.

Se, S., Lowe, D. G., and Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, 21(9):735–758.

Se, S., Lowe, D. G., and Little, J. J. (2005). Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21:364–375.

Searle, J. (1980). Mind, brains and programs. *Behavioral and brain sciences*, (3):417–458.

Seo, M. W., Kim, Y. J., and Lim, M. T. (2005). *Knowledge-based intelligent information and engineering systems (LNCS)*, volume 3684, chapter Door

Traversing for a Vision Based Mobile Robot using PCA, pages 525–531. Springer-Verlag.

Sethian, J. A. (1997). Level set methods: An act of violence - evolving interfaces in geometry, fluid mechanics, computer vision and materials sciences.

Siegwart, R. and Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*. MIT Press.

Skarbek, W. and Koschan, A. (1994). Colour image segmentation – a survey. Technical report, Polish Academy of Sciences.

Smith, S. M. and Brady, J. M. (1997). SUSAN: a new approach for low level image processing. *International Journal of Computer Vision*, 23(1):45–78.

Stachniss, C., Mozos, O. M., and Burgard, W. (2008). Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):205–227.

Stella, E., Cirelli, G., Attolico, G., and Distante, A. (1996). Sauro: An autonomous mobile vehicle for indoor environment. In *MELECON*, volume 2, pages 1145–1150.

Stoeter, S. and Papanikolopoulos, L. M. (2000). Real-time door detection in cluttered environments. *Proceedings of the IEEE International Symposioum on Intelligent Control*, pages 187–192.

Szelisky, R. (2010). *Computer Vision: Algorithms and Applications*. Springer Verlag.

Tamimi, H., Halawani, A., Burkhardt, H., and Zell, A. (2006). Appearance-based localization of mobile robots using local integral invariants. In *International Conference on Intelligent Autonomous Systems (IAS-9)*, pages 181–188, Tokyo, Japan.

Tardós, J. D., Neira, J., Newman, P. M., and Leonard, J. J. (2002). Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research*, 21(4):311–330.

Tax, D. (2001). *One-class classification; Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology.

Thorpe, C. F. (1984). Path relaxation: Path planning for a mobile robot. Technical Report CMU-RI-TR-84-5, The Robotics Institute. Carnagie-Mellon University.

Thrun, S. (1998a). Bayesian landmark learning for mobile robot navigation. *Machine Learning*, 33(1):41–76.

Thrun, S. (1998b). Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Thrun, S. (2002). Robotic mapping: a survey. Technical Report CMU-CS-02-111, School of Computer Science. Carnagie Mellon University.

Thrun, S. (2003). Robotic mapping: a survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring Artificial Intelligence in the New Millennium*, pages 1–35. Morgan Kaufmann.

Thrun, S., Buecken, A., Burgard, W., Fox, D., Froehlinghaus, T., Hennig, D., Hofmann, T., Krell, M., and Schmidt, T. (1998a). Map learning and high-speed navigation in RHINO. In Kortenkamp, D., Bonasso, R. P., and Murphy, R., editors, *Artificial Intelligence and Mobile Robots. Case Studies of Succesful Robot Systems*, pages 21–52. The AAAI Press. MIT Press.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99–141.

Thrun, S., Gutmann, J., Fox, D., Burgard, W., and Kuipers, B. (1998b). Integrating topological and metric maps for mobile robot navigation: a statistical approach. In *AAAI/IAAI*, pages 989–995.

Tomatis, N., Nourbakhsh, I., and Siegwart, R. (2001). Combining topological and metric: A natural integration for simultaneous localization and map building. In *Proceedings of the Fourth European Workshop on Advanced Mobile Robots*.

Trahanias, P. E., Velissaris, S., and Oraphanoudakis, S. C. (1999). Visual recognition of workspace landmarks for topological navigation. *Autonomous Robots*, 7:143–158.

Trullier, O. and Meyer, J. A. (1997). Biomimetic navigation models and strategies in animats. *AI Communications*, 10(2):79–92.

Trullier, O., Wiener, S. I., Berthoz, A., and Meyer, J. A. (1997). Biologically-based artificial navigation systems: review and prospects. *Progress in Neurobiology*, 51:483–544.

Turing, A. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460. http://www.oxy.edu/departments/cog-sci/courses/1998/cs101/texts/Computing-machinery.html.

Ulrich, I. and Borenstein, J. (2000). VFH: local obstacle avoidance with look-ahead verification. *IEEE Transactions in Robotics and Automation*.

Van Breemen, A. (2001). *Agent-based multi-controller systems: a design framework for complex control problems*. PhD thesis, University of Twente.

Walter, G. (1953). *The living brain*. Duckworth.

Wang, Q. and Lopes, L. S. (2005). *Emerging Solutions for Future Manufacturing Systems*, volume 159, chapter One-Class Learning for Human-Robot Interaction, pages 489–498. Springer Boston.

Webb, B. (2001). Can robots make good models of biological behavior? *Behavioral and Brain Sciences*, 24(6).

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2009). A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous System*, 57:1157–1172.

Yamauchi, B. and Beer, R. (1997). Spatial learning for navigation in dynamic environments. *Robotics and Autonomous Systems*, 26(3):496–505.

Yamauchi, B., Schultz, A., and Adams, W. (1998). Mobile robot exploration and map-building with continuous localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3715–3720.

Ye, W. and Zhong, Z. (2007). Robust people counting in crowded environment. In *Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics*, pages 1133–1137.

Yuen, H. K., Princen, J., Illingworth, J., and Kittler, J. (1990). Comparative study of Hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71–77.

Zanichelli, F. (1999). Topological maps and robust localization for autonomous navigation. In *IJCAI Workshop on Adaptive Spatial Representation on Dynamics Environments*.