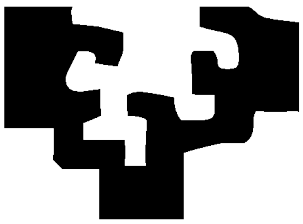
On Computational Intelligence Tools for Vision Based Navigation of Mobile Robots

By

Ivan Villaverde de la Nava

http://www.ehu.es/ccwintco

Dissertation presented to the Department of Computer Science
and Artificial Intelligence in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy



PhD Advisor:

Prof. Manuel Graña Romay

At
The University of the Basque Country
Donostia - San Sebastian
2009

Universidad del País Vasco Euskal Herriko Unibertsitatea

AUTORIZACION DEL/LA DIRECTOR/A DE TESIS

PARA SU PRESENTACION

Dr/a. _____con N.I.F._____

como Director/a de la Tesis Doctoral: _____

_____

_____

realizada en el Departamento _____

_____

por el Doctorando Don/ña. _____ ,

autorizo la presentación de la citada Tesis Doctoral, dado que reúne las condiciones necesarias para su defensa.

En _____a _____de _____de _____

EL/LA DIRECTOR/A DE LA TESIS

Fdo.: _____

Universidad del País Vasco Euskal Herriko Unibertsitatea

CONFORMIDAD DEL DEPARTAMENTO

El Consejo del Departamento de _____

_____

en reunión celebrada el día ____ de _____ de _____ ha acordado dar la conformidad a la admisión a trámite de presentación de la Tesis Doctoral titulada: _____

_____

_____

dirigida por el/la Dr/a. _____

_____

y presentada por Don/ña. _____

ante este Departamento.

En _____a _____de _____de _____

Vº Bº DIRECTOR/A DEL DEPARTAMENTO SECRETARIO/A DEL DEPARTAMENTO

Fdo.: _____ Fdo.: _____

Universidad del País Vasco Euskal Herriko Unibertsitatea

ACTA DE GRADO DE DOCTOR

ACTA DE DEFENSA DE TESIS DOCTORAL

DOCTORANDO DON/ÑA. _____

TITULO DE LA TESIS: _____

_____

_____

El Tribunal designado por la Subcomisión de Doctorado de la UPV/EHU para calificar la Tesis Doctoral arriba indicada y reunido en el día de la fecha, una vez efectuada la defensa por el doctorando y contestadas las objeciones y/o sugerencias que se le han formulado, ha otorgado por_____la calificación de:

unanimidad ó mayoría

En _____a _____de _____de _____

EL/LA PRESIDENTE/A, EL/LA SECRETARIO/A,

Fdo.: Fdo.:

Dr/a: _____ Dr/a: _____

VOCAL 1º, VOCAL 2º, VOCAL 3º,

Fdo.: Fdo.: Fdo.:

Dr/a: _____Dr/a:_____Dr/a:_____

EL/LA DOCTORANDO/A,

Fdo.: __

Agradecimientos

Hay una multitud de gente sin la que llegar hasta este punto me hubiese sido imposible y sin cuya ayuda esta tesis no hubiese llegado a término. Aunque me es imposible nombrarlos a todos personalmente, sí deseo mencionar, aunque sea de manera general, a aquellos que han contribuido especialmente a alcanzar este objetivo.

En primer lugar deseo agradecer al Prof. Manuel Graña Romay, mi director de Tesis, su apoyo y guía durante todo este tiempo y que haya sido capaz de orientarme para que siga este camino hasta el final.

A mis compañeros del Grupo de Inteligencia Computacional, por ser más que simples compañeros de trabajo y laboratorio.

A mi familia y amigos, por estar ahí y aguantarme incluso cuando soy inaguantable.

Finalmente, a todos los autores de ciencia ficción que en los últimos 150 años nos han inspirado con sus visiones de mundos futuros y sin los que probablemente ahora estaría haciendo algo completamente distinto.

A todos vosotros, muchas gracias.

On Computational Intelligence Tools for Vision Based Navigation of Mobile Robots

by
Ivan Villaverde de la Nava

Submitted to the Department of Computer Science and Artificial Intelligence on September 30, 2009, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

abstract

This PhD Thesis has dealt with two types of sensors, studying their usefulness for the development of navigation systems: optical cameras and 3D range finder cameras. Both were used to attack the problems of self-localization, egomotion and topological and metric map building. Good results have been obtained using Lattice Computing, Evolution Strategies and Artificial Neural Network techniques. Lattice Computing has been applied to self-localization in qualitative maps and to visual landmark detection using optical cameras. For metric localization tasks with 3D range finder cameras, Competitive Neural Networks and Evolution Strategies have been used to estimate the transformations between 3D views, which will provide the estimation of the robot's movement. Finally, we have performed a proof-of-concept physical experiment on the control of Linked Multi-Component Robotic Systems with visual feedback.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we will provide a general introduction of the PhD Thesis Report. Section 1.1 provides some historical background for the PhD Thesis work. With this background in mind, section 1.2 gives the main motivations behind the different parts of the work done. Section 1.3 states the objectives pursued with some degree of specificity. Section 1.4 refers the publications done while doing the research of this PhD Thesis, and some that are submitted or will be submitted shortly. Section 1.6 summarizes what we think of as the major contributions of the PhD Thesis work. Section 1.7 gives a guide to the structure of the PhD Thesis report in our hands.

1.1 Background

1.1.1 From Archytas' pigeon to Honda's Asimo: A short history of robotics

Nowadays, a few years into the 21st century, robotics has become for the general public no more than another of the ever evolving technologies that surrounds us every day. Decades of science-fiction media and extremely successful industrial robotics have taken robotic technologies out of the obscure experts-restricted-area of modern technology. Even though, this "popularization" of the technology has not completely removed the fascination and surprise from people, who still feels amazed every time that a new Japanese humanoid robot appears dancing in the TV. This fascination for the robotics roots very deep in the human nature. The desire to know and reproduce the

mechanics of life and animals have been always present in our minds since the very beginnings of civilization. Also, ancient Greek philosophers like Aristotle already imagined a world in which mechanical tools would ease or even prevent human work, thus ending any social injustice based on master-servant relationships [3]. The fusion of this two desires could be certainly in the origin of the robotics.

This origin can be traced back to the mechanical automatons of the antiquity. Driven by mainsprings, weights or water, built mostly for amusement and usually with little practical application, history is full of references of those automatons. Earliest references connect with mythical origins, like the golden handmaids of Hephaestus' workshop mentioned in the Iliad [52] or the mechanical animals of the Solomon's throne of the Jewish tradition. More factual references can be found from ancient Greek mathematicians such as the works of Archytas of Tarentum (3rd century B.C.) and his flying wooden pigeon [16], or Hero of Alexandria (1st century A.D.), creator of the first programmable automaton [103]. Muslim inventors took over from the Greeks in the middle ages, being Al-Jazari's automatic musicians band (c. 1206) credited as the first programmable humanoid automaton [31]. The Renaissance took back the interest in automatons to Europe, and most European courts received inventors and their machines. Automatons grew in complexity, from the Da Vinci's mechanical knight (c. 1495) [106] to the Vaucauson's "digesting duck" (1737), capable of imitate the movements of a duck and even to eat and defecate [90]. Eventually, steam started to be the power source for automatons, and the industrial revolution changed the orientation of those inventions to the more practical tools imagined by Aristotle (but not always with his expected results, though), being the automated weaving machines the ancestors of modern day industrial robots.

Industry automation saw an enormous development along the 19th century, but the modern popular concept of a robot as more than simply an "automatic machine" came in the first half of the 20th century from the science-fiction media. The word "robot" itself was coined in 1921 in Karel Čapek's *"R.U.R. (Rossum's Universal Robots)"* [22] and the concept was popularised in movies like Fritz Lang's "*Metropolis*" or the tales and novels by Isaac Asimov, who coined the term "robotics" in his short tale *"Liar!"*. Soon the real world caught up. *Televox*, world's first "robot", was built in 1926 from an automated electrical substation controller, and the Japanese robot Gakutensoku impressed audiences in 1928 with its movements and facial expressions. Those were followed by many other electro-mechanical devices

(a) (b) (c)

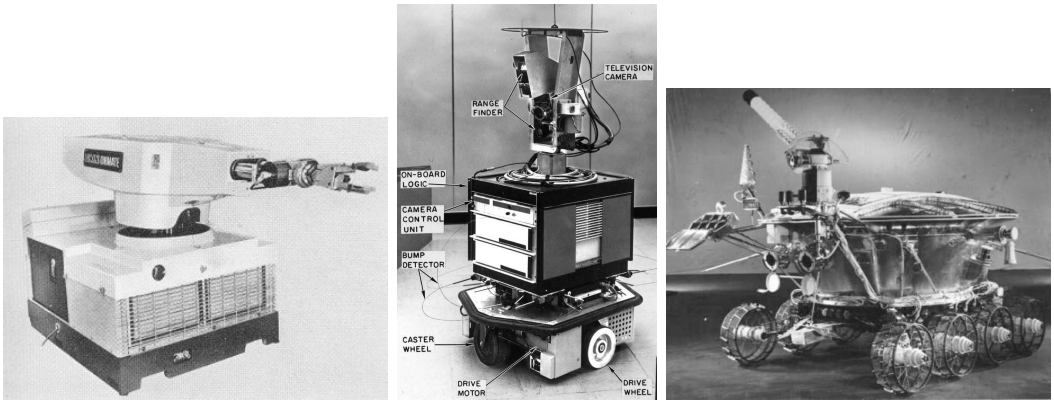
Figure 1.1: Examples of automatons: (1.1a) Al-Jazari's music band, (1.1b) da Vinci's knight and (1.1c) Vaucauson's digesting duck.

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

Figure 1.2: Electro-mechanical robots. (1.2a) Televox, (1.2b) Gakutensoku, (1.2c) Elektro and (1.2d) Elsie.

until the development of the electronics [102], including the popular *Elektro* (1938) or *Elsie* and *Elmer* (1948), the first autonomous robots. The first digital and programmable industrial robot, Devold's *Unimate*, was built in 1954 and installed in 1961 [78], leading to the 70's explosion on industrial robotics and later expansion to other areas. It can be said that intelligent robotics started properly with SRI's *Shakey the Robot*, the first robot able to reason about its actions and its surroundings [77].

Today, robots are used, or there are attempts to use them, in most of the human activities. Robots have substituted humans on most of the hard, repetitive and dangerous tasks on heavy industries. In some industrial sectors, like car industry, humans are almost relegated to mere supervision tasks.



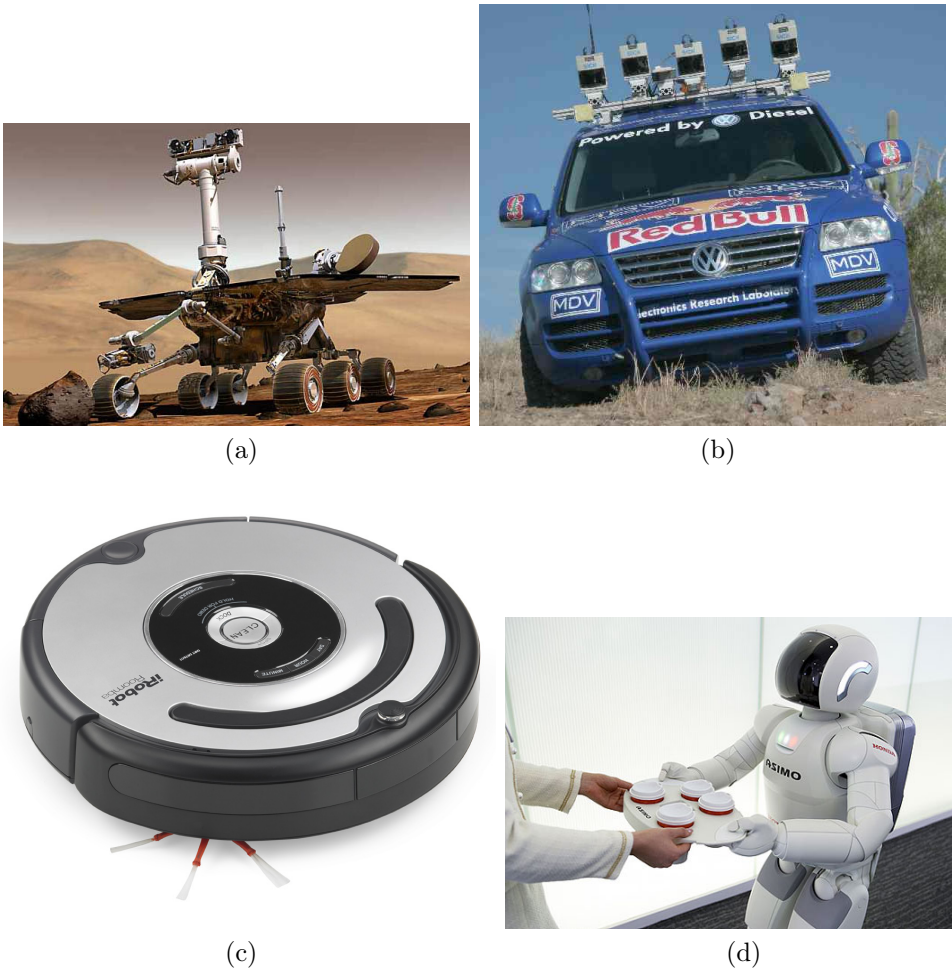
(a) (b) (c)

Figure 1.3: Early electronic robots. (1.3a) Unimate, (1.3b) Shakey the Robot and (1.3c) Lunokhod 1 lunar rover.

Robots explore the deeps of the sea [72]. Robots have been our first ambassadors to other worlds, from the soviet era automatic probes *Luna 9* (first probe to ever land in another celestial body, the Moon, in 1966 [74]) and *Venera 7* (first probe to land in another planet, landing in Venus in 1970 [75]), to the modern *Spirit* and *Opportunity* Mars rovers [73]. Robots are taking their place in our houses as waiters [71] or automatic vacuum cleaners [55]. Robots are being developed to assists the elderly [104] and help surgeons in the operating room [5]. Last developments serve to find and rescue victims of disasters [18], to fight fires [1], to drive cars [17] and to dispose dangerous wastes [84].

This scenario lead us to think of a future in which, as Aristotle imagined, humans are relieved from their work as more and more complex robots are introduced until they take over every work humans do not like to do themselves. But, quoting the physicist Michio Kaku, "*Our most advanced robots have the intelligence of a retarded cockroach*". As impressive as they can look, today's robots are simply evolved descendants of the ancient automatons, with limited capabilities, even the fully programmable ones, and mostly dedicated to repetitive tasks in static environments with very strictly controlled interaction with their surroundings.

In the current state of development in robotics there is a gap between the physical capabilities of the robots and their real capabilities. Most mechan-



(a) (b)

(c) (d)

Figure 1.4: Nowadays robots: (1.4a) Opportunity Mars rover. (1.4b) Stanford's Stanley, winner of the 2005 DARPA Grand Challenge. (1.4c) iRobot Roomba robotic vacuum cleaner. (1.4d) Honda Asimo.

ical problems of robots have been solved to some degree. Today there are robots, like Honda's Asimo [97], that can physically move like a person. However, they fail when they try to replicate more complex human behaviours, as they lack the necessary feedback and adaptation mechanisms to react to changes in their surroundings. There is still plenty of work to do in the mechanical aspects of robotics, but currently the most challenging research in robotics focus in the computational aspects. We have made robots that walk. Now we want robots that are able to walk among people. To interact with them. To recognise and learn new places. Robots need to perceive and interact with their environment, as living beings do, in order to be able to accomplish tasks more complex than "move this box between two fixed points". And is in this context in which this PhD Thesis works takes place. We have explored new ways of how robots can perceive their environment and use this information to learn it, in order to be able to interact with it.

1.1.2 Perceiving the world outside: how robots obtain information of their surroundings

As stated above, robots need to perceive their environment in order to be able to interact with it and to adapt to changes in it. In other words, robots need to "sense" the world. That means that robots not only need actuators, which allow them to move and carry out their task, but also a variety of sensors, which will allow them to know *how* can they move and carry out their task *inside an environment*.

Depending on the environment in which the robot is going to be placed and the task it is expected to perform, different types of sensors can be used. The most basic sensors simply measure the internal state of the robots themselves. In a typical industrial robotic arm, the operation of the robot is going to be limited to a fixed, well known, static and usually small area. The position of the robot in reference with the environment is know, so the robot does not need to sense the world outside, but it only has to know its own configuration. In the same way that a person is able to know the position of their arms and can turn on the lights of the room he has just entered without actually looking at the switch, a robot which knows its configuration can perform simple tasks in a known environment. In robotics, this type of information is usually provided by the direct measurement of part motion by the so-called encoders.

An encoder is basically a sensor that is attached to a moving part of the robot, generally a rotating part like motors or gears, and measures its rotation. To obtain this measurement, the moving part is "marked" at constant distances and the measurement is done by merely counting the number of those marks that pass in front of the sensor as the part moves. In a typical example of encoder these marks are set at the drive wheels of a robot. Counting the marks as it turns gives us the position of the wheel. Also, if the radius of the wheel is known, traveled distance, speed and acceleration are easily calculated. The estimation of the state of a robot by means of their self sensing encoders is known as *odometry*. This estimation is very sensitive to the precision and robustness of the sensors. For example, one wheel with 360 marks will have a resolution of 1°. That means that each measurement will have a mean error of 1/360th of the circumference of the wheel, which will accumulate over time. This accumulation of the encoder measurement error implies that even in the most controlled environments, robots that depend merely on odometry to operate require very high encoder resolutions and periodically frequent calibrations.

But encoders are not the only way to self sense the state of a robot. Gyroscopes, accelerometers, inclinometers or inertial sensors can also be used to estimate the state of the robot or to mitigate the effects of the error of the encoders. Usually those sensors are only present in the most advanced robots due its complexity and cost and, in any case, they can not provide information of the environment. Most other robots and tasks require of sensors to perceive the state of their environment. Other means of obtaining information about the robot status relying in some external reference are the compass, which can give the robot orientation, and, in open air (outdoor) environments, a GPS can obtain the robot spatial localization with great precision.

Robotics usually is inspired by Nature and tries to emulate the solutions that Nature has evolved over time to perform the same tasks robots try to do. It does by replicating mechanically their components to build robotic arms or legs. And it does it also in the way robots try to sense their external environment. The simplest form animals use for sensing their surroundings is the sense of touch. Even humans, when deprived of vision, use the touch to feel their way around. In robots, also, the simplest of those external sensor is the contact switch. This sensor is nothing more that a switch that closes a circuit when something pushes it. Typical examples would be the bumpers of a wheeled robot that tells the robot if it has collided with a wall or the

mechanical limit switch in the borders of the operation area of a robotic arm which indicate it when it has reached the limits of its working space. But, as in Nature, the simplest approach can evolve growing in complexity, and, therefore, more advanced "touch sensors" are being developed, bio-inspired in the sense that their design is inspired by the use that some animals like rats made of their whiskers [83].

Contact sensors have, after all, an obvious limitation: their reach is limited to the immediate surroundings of the robot, as they can sense only the part of the world that is in contact with the robot. This limitation can be a real issue. Starting with, contact sensors prevent any preplanning and only allow very limited reactive behaviours (i.e. robots can only take immediate decisions based on direct contact with obstacles). They can not avoid an obstacle from far away, but merely advance until they touch it and decide that it is an impossible to follow direction. Also, sensing things only when they are in touching reach can be just too late. Humans do not want that robots discover where they are... running over them. In the same way, a robot moving at a certain speed can sense too late the cliff it is headed to. Some remote sensing capability is, thus, required.

When sensing the environment, it can be measured in several ways and not only in a spatial way. Simplest forms of remote sensors consist in passive devices which measure one characteristic of the environment like its light intensity, temperature or noise level. Photoelectric cells were the very first external sensors mounted in autonomous robots. William G. Walter's "turtle robots" (*Elmer* and *Elsie* robots mentioned above) were equipped with light and contact sensors and were able to phototaxis (i.e. drive towards the direction of the light) in a way similar to some microorganisms and lower life-forms. Similarly, microphones can be used to "listen" the environment and drive a robot to the source of its commands, or drive it away from loud noises that can be interpreted as dangerous.

The most valuable information a robot can get from its environment is spatial. With little exceptions, the information a robot really needs to know is *where* things are in its surroundings. Robots which want to interact with their environment need to know where are the walls they can crash into, the doors they can pass through, the obstacles they have to avoid, the people they have to greet and the objects they have to take. In order to have this knowledge, the environment has to be modeled in some way. Even if the robot is provided with an *a priori* model, the robot needs to measure its surroundings and model those measurements to match them with this *a*

priori knowledge. But acquiring spatial information is not an easy task, so usually this modelling is usually necessarily limited (e.g.: a robot measures the distances in every direction around it at the height of its head).

Range sensors are a family of sensors that measure distances following the same principles used in sonar or radar, and are used pretty much in the same way some animals like bats or dolphins echolocate. They fall in the category of "active" sensors, since they emit a wave pattern (sinusoidal) in some physical medium (sound, light...) and measure the returned echoes of this wave created by the objects around in order to calculate the distances to them. Range sensors usually are only capable of measuring a distance in a line, or a very narrow cone, following the sensor orientation, and a reconstruction of the environment can be done only by integrating different measurements either obtained sequentially with the sensor oriented to different directions in time, or simultaneously by mounting several sensors oriented to different directions, the so-called *sensor ring*. Depending on the type of range sensor and the characteristics of the wave pattern and associated physical medium it uses, precision, range and speed of measurement can be very different, making each type suitable for different purposes.

IR (Infra-Red) range finders are the most basic of this kind of sensors. They consist in a pair of infrared LEDs, one of them configured as transmitter and the other as receiver. They use the rebounds of the infrared light emitted by the transmitter LED to measure distance to objects. Those rebounds are captured by the receiver LED, which converts them into a voltage proportional to the intensity of the light received. This intensity is also proportional to the distance to the object it rebounded from, so the estimation of the distance is immediate. Some models improve this estimation by means of triangulation techniques, knowing with precision the position of both transmitter and receiver LEDs. Anyway, those sensors are very limited in their capabilities. Their range is very short, in the order of tens of centimeters, with a very low precision. Also, as its measurement depends on the intensity of the reflected light, it is very sensitive to the reflectivity of the materials it rebounds from, and the angle of incidence of the irradiated light over the object's surface. Despite those problems, IR range finders are still widely used in robotics, due to its small size, price and power consumption, as close range secondary sensors, usually as proximity warning sensors.

The next type of range sensors, and probably the most used one until recently, are the ultrasonic sensors. Those sensors are based on SONAR (SOund Navigation And Ranging) technology and use sound to measure dis-

tances. The sensor emits a very high frequency ultrasonic wave and uses the Time-of-Flight (ToF) principle to estimate distance to objects, measuring the time it takes to the wave emitted to come back to the sensor after bounding. In contrast to the IR light emitted by the infra-red range finders, ultrasonic waves can not be focused completely on a point and they have a wide spread. The measurement provided by the ultrasonic sensors does not correspond only to the closest point in the directions the sensors are pointed at, but they receive echoes from surfaces localized within cone shaped areas of the environment, each one with the corresponding sensor in it's vertex. This implies that the spatial resolution of the measurement decreases greatly with the distance, as echoes from a given distance can be situated in any point in an arc spanning the width of the cone (e.g. in a sensor with a field of view of 10°, an echo at a distance of 20 cm can be from an object in any point into an arc of 3.5 cm. At 1 meter distance, the length of the arc would be 17.5 cm, an so on). Also, the sensor can get echoes from several sources (e.g. two separate obstacles in the width of the cone) or several rebounds from the same source (e.g. rebounds from a wall). Another consequence of this cone shaped field of view is that, when used as part of a sensor ring, different sensors can not be fired simultaneously. Instead, they have to be fired in a precise sequence in order to avoid receiving interferences from signals emitted by other sensors in the ring. Other sources of uncertainty came from the characteristics of the objects ultrasonic waves are incident into, like their shape or their capacity to absorb or reflect sound waves (e.g. sound-absorbing isolating materials versus hard flat surfaces). Anyway, ultrasonic sensors keep being the *de facto* standard range sensors, since they precision and range surpasses IR range finders, while being quite more low-priced and less bulky than laser range sensors.

Laser range finders are the most precise and have the longest reach of all the range sensors used in robotics. The ones used for LiDAR (Light Detection and Ranging) usually use the same ToF principle as the ultrasonic sensors, but sending a laser pulse instead a ultrasonic wave. They have a very large range which, depending on the application, can be from hundreds of meters up to several kilometers, and they are even used to measure the distance to objects in space like artificial satellites or even the Moon. The laser beam is very narrow, so it does not have the same spatial resolution problems of the ultrasonic sensors. Laser beams can be affected by atmospheric conditions like temperature gradients or hot air bubbles, but those problems arise at distances over hundred of meters. Robotics oriented laser range finders

are usually designed to operate at ranges of tens of meters, in which those problems have very little measurable effect in their precision. Their effective range will be determined mainly by the power of the laser used and their precision by the speed of the hardware used to measure the time of the rebounds, due the extremely high speed of light beams. For other applications in which the highest precision is required other techniques like triangulation or multiple frequency phase shift can be used. The typical laser rig used in robotics consists in a laser range finder and a rotating mirror which points the beam at different orientations. Each full rotation of the mirror provides measurements at specified angle intervals in a wide field of view of more than 180° and a resolution of less than 1°, with scan rates of around 30-50 Hz. As they ranging capacities are excellent, drawbacks of those systems for mobile robotics come from their bulky size (usually weighting several kilograms), high power consumptions and very high price (in the order of several thousand euros). Recently new highly mobile robotics oriented laser range finder rigs have been started to be released, offering lightweight devices but with much less range, and still at high prices.

Finally, the last widely used type of sensors are optical sensors: video cameras mounted on a robot which capture images of the robot's surroundings. The simplest installation is a front faced camera that records the scene that there is just in front of the robot, and the most common one is with the camera mounted on a Pan-Tilt unit that enables the camera to "look around". Different ways of mounting and combining cameras with other elements provide different optical information. Omnidirectional cameras are mounted vertically with a conic mirror that allows them to obtain an image covering 360° around the robot. Binocular and trinocular rigs provide stereo vision that allows obtaining depth information from the images. Recently, some IR cameras which combine characteristics of traditional optical cameras and range sensors have been introduced. Since digital camera technology is widespread and almost any commercial camera is suitable to be mounted on a mobile robot, optical cameras are cheap, small and widely available, and are of very common use in mobile robotics.

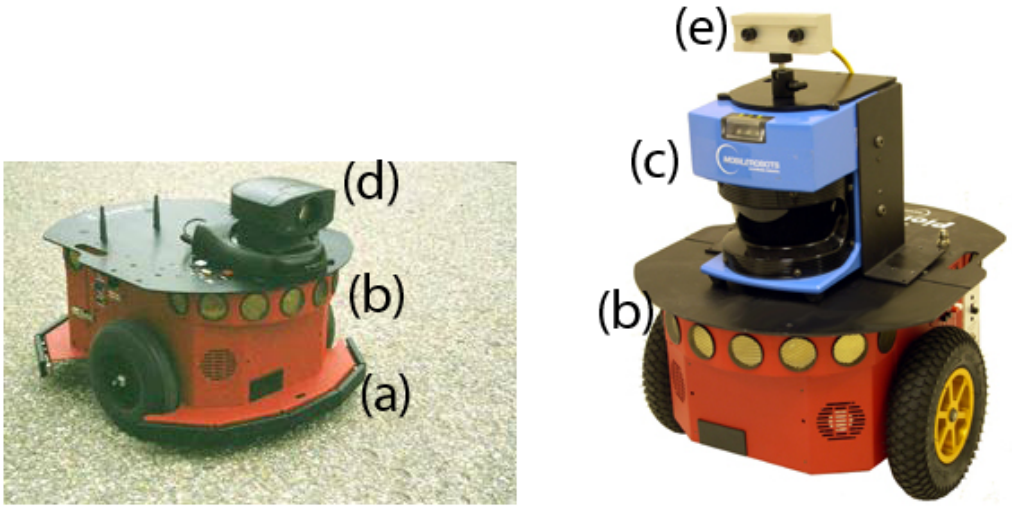


Figure 1.5: Mobile Robots' Pioneer DX robots showing a variety of sensors. (a) Bumpers, (b) ultrasonic sensor ring, (c) laser range finder and optical cameras on a (d) pan-tilt unit and (e) binocular stereo rig.

1.1.3 Getting an image of the world: computer vision in mobile robotics

It has been stated that robotics try to emulate Nature. In Nature visual perception is a privileged way for animals to obtain information from its surroundings. Almost any superior animal (except the ones living in environments in complete darkness, without any source of light) depends on vision. Even the ones which have developed alternative ways of sensing the world, like bats or dolphins, still made use of their eyes to perceive it in some way. Millions of years of evolution have refined a visual system capable of obtaining huge quantities of useful information of the surroundings like depth, color, shape or motion.

But, although for us vision is a very natural way of perceiving our environment, the processing required for extracting that useful information is far from trivial. The way visual information is processed in animal brains is largely unknown, and research in the area has not been extensive until late 70's, when computers started to spread, allowing the processing of the large quantities of data required for those studies and opening a new field of research: computer vision.

"Computer vision is the science that develops the theoretical and algorithmic basis by which useful information about the world can be automatically extracted and analyzed from an observed image" [51], where image is defined as the projection of an object or scene, with the goal of making useful decisions about those real physical objects and scenes [101]. Although initially applied only to optical images, computer vision techniques have also been extended to the processing of more complex types of images, like volumes obtained from MRI or hyperspectral images, and other image-like data not obtained with optical systems. Also, although it would seem there is only a problem to solve in computer vision (i.e. "to see"), there are actually a large amount of different problems to solve, depending on the kind of information we want to obtain from the images. Since no global solution have been found for "the vision problem", individual computer vision problems like object recognition, motion estimation or scene reconstruction have to be solved with specific methods and techniques. This abundance of data types, problems and techniques to solve them makes that the boundaries of computer vision are not clearly established and they blur and overlap with other research areas like signal and image processing or pattern recognition, up to the extent of not being clear if they are actually different fields or just an specific area inside a wider research field, as most of the methods used in them are common.

Despite the variety of methods and problems, most computer vision applications are composed of several processing steps, from the raw image data acquisition to the most higher-level processing:

- **Image acquisition:** The process of obtaining a digital image from the sensors. Depending on the kind of sensor used it can have the form of a 2D array or a 3D volume, each data unit representing light intensities in several (gray-scale cameras) or one (RGB and hyperspectral cameras) wavelengths, distances (LiDAR, RADAR), nuclear magnetic resonance (MRI) or other types of data. This task is mainly hardware related.

- **Preprocessing:** Raw image data usually has to be processed in order to be possible to apply other methods to extract useful information from it. The acquisition hardware used to capture the image data can introduce noise or artifacts in the image which have to be cleaned to avoid interference with other processes. Examples of this problem can be the specular light reflections or chromatic noise in digital optical cameras, inhomogeneities in MRI or pixels excited by cosmic rays in

astronomical imaging. Other required preprocessing can be oriented to enhancing images to ease further processing, like contrast or exposition correction.

- **Segmentation:** It is the process of dividing the image into its constituent regions or objects [36] in order to reduce the regions which require further processing or to identify desired elements in the image. Segmentation is one of the most crucial tasks in image processing, since its precision can determine the success of further methods applied to segmented regions.

- **Feature extraction:** To obtain the useful information from the images. Desired features can be of very different types, including shapes, lines, edges, invariant points, corners, color regions, textures, optical flow, etc.

- **High-level processing:** The use of extracted features to perform an specific task. Typical tasks could be face detection, object detection and classification, motion estimation or distance estimation.

The application of computer vision to robotics is usually known as *Robot Vision*. Robot vision can be applied to most of the tasks in which a robot has to interact with its environment. There are, however, several task in which most of the research have been focused. Most of those representative tasks are specific of mobile robotics and try to allow them to recognize, learn and move inside an environment. In any case, visual data processing methods and techniques are the same as that for other computer vision applications, so the differences with other applications are mainly in the high-level processing. Some typical robot vision tasks are:

- **Tracking:** Detection of objects and continuously measure their position in the sensor's reference system. Used to detect and track possible obstacles or find task goals.

- **Visual servoing:** The direct embedding of visual error information into the control feedback loop, in order to move a robot or an actuator towards a tracked target or keep a relative position with respect to it. Typical examples could be moving a robotic arm to pick up an object or drive a robot towards a door.

- **Ego-motion:** It is the estimation of the relative camera spatial movement between two consecutive images. Integrating several of those estimations the path followed by the movement of the robot can be reconstructed.

- **Self-Localization:** Estimation of the position of the robot inside an environment using its own sensorial information. In the case of visual information some image features or the whole images may be used as landmarks. Each landmark has a known corresponding spatial position, which allows the robot which has detected it to reference its own position in a world model. This landmark spatial position implies an *a priori* knowledge of the environment in the form of a map. Localization is not always a continuous process, but sometimes it is performed instead only at specific time steps, keeping track of the position of the robot between steps by dead-reckoning techniques like odometry.

- **Map building:** It is the process of building a model of the environment the robot is moving along. It implies the detection of landmarks and the correct estimation of their position in order to integrate them with previous measurements. Maps can take several forms, like occupancy grids, metrical or topological maps. Map building can be an independent task, but when it is performed simultaneously with localization the resulting process is known as *Simultaneous Localization and Mapping* (SLAM).

- **Navigation:** The ability of a robot to move inside an environment with a specific purpose [13], guided by visual information. It makes use of other robot vision techniques in order to perform its entrusted task. The robot needs to locate itself into the environment and to have a map of it in order to perform path planning to reach its goal. Obstacles in its path have to be detected and tracked in order to avoid them, and visual servoing can be used once the final goal is detected.

As can be appreciated from the above task definitions, robot vision applications are not completely independent and usually relate and depend on others. For example, the task of self-localization relies on the detection and tracking of visual landmarks, so basically uses the same methods that tracking applications use. Robot vision applications usually have to repeat a task

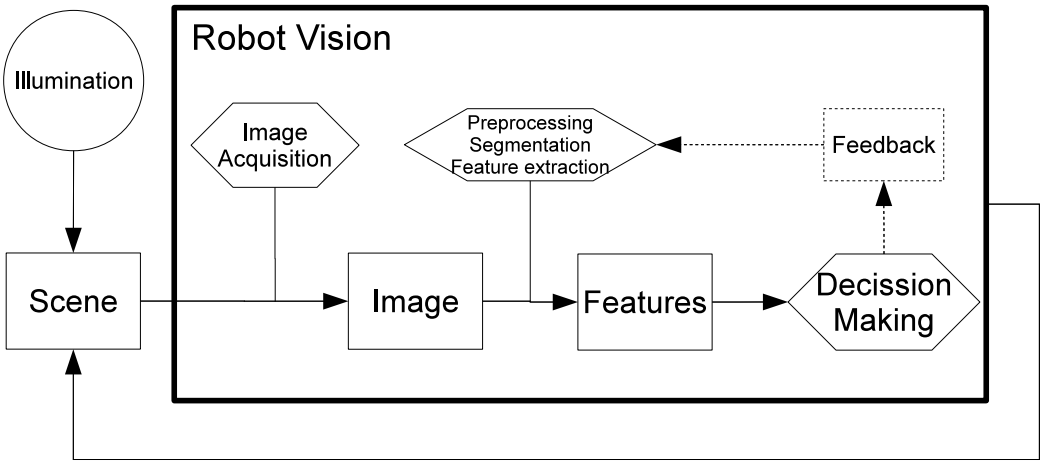


Figure 1.6: Robot Vision Cycle.

in a closed cycle [53], which can also have some feedbacks to the next iteration, like definition of regions of interests for tracking tasks or dynamic color thresholds for segmentation (figure 1.6).

Robot vision is a wide research field, covering a great number of applications and techniques. The works presented here center in the more general navigation problem, focusing mainly on applying new Computational Intelligence based Artificial Vision techniques to the localization and map-building tasks.

1.2 Motivation and general orientation of the PhD work

The works in this PhD Thesis have been devoted to explore the use of innovative Computational Intelligence techniques for vision based localization and mapping for mobile robots. In particular, the two main computational approaches proposed and tested during this PhD works were

- The ones based on Lattice Computing [37], in the form of several applications of Lattice Associative Memories, and
- The ones based on Hybrid Systems combining Competitive Neural Networks and Evolution Strategies.

We have also proposed and realized a proof-of-concept physical experience on the vision based control of a Linked Multi-Component Robotic System (MCRS), where the innovation is not in the image processing but in the kind of system realized. Before going into detailed comments on the motivation of each of the approaches pursued during this PhD works, some words about the general orientation and methodology of the work must be said. We have always tried to bring the ideas to their physical realizations on the available robotic platforms, which will be described elsewhere, therefore there is a long and unrewarding amount of work behind each experiment reported in the following pages. There are additional obstacles which are common place in mobile robotics that have been also present in the development of the Thesis works:

- The lack of reproducibility of the results because of the many uncontrollable factors that converge in each experiment.
- The difficulty of measuring the behavior of the robots.
- The hidden efforts invested in the maintenance of the equipment and the setting of the experimental fields.
- The need to perform the experiments at night hours or profiting from exceptional circumstances, like the precise moment when a classroom will be emptied because of the general refurbishing of the Facultad de Informática building.

From a methodological point of view, we have shifted from the early attempts to obtain on-line real time results by implementing and installing our software on the robot's on board computer to the realization of computational experiments on field gathered data. The main reason for that shift is the need to factor out random effects of the environment in our evaluations of the algorithms. That is, we needed to be able to reproduce the same results and to identify the variations due to changes in our algorithms, in order to be able to improve and debug them. As a result of our efforts, we have also produced some collections of sensor and odometry data that we have made public for two reasons:

(1) to contribute to the community pool of data for validation of algorithms, and

(2) to allow the independent reproducibility of our own results by other research groups.

In some cases, like the multi-robot transportation of a hose in chapter 4, we think that the successful realization of the experiment is by itself an interesting contribution, although it can be difficult to reproduce by other research groups.

1.2.1 Lattice Computing approaches to localization and mapping

In recent years, the Lattice Theory has been identified as a central concept for a whole family of methods and applications in Computational Intelligence [56, 57]. One of those Lattice Computing based methods are the Lattice Associative Memories (LAM), first introduced in [95, 91] as Morphological Associative Memories. LAMs were initially proposed for the storage of binary and gray patterns, with the aim of recovering the original clean image from noisy copies. This pattern storage and retrieval capacity could also be used in a robotic mapping context as a way of storing individual views that identify map positions. This approach was first explored in [85, 87], approach we continue in the works reported in section 2.3. LAMs are also been used for spectral unmixing of hyperspectral images [42, 43], leading to the use of the convex coordinates produced by the unmixing process as features for classification purposes, approach we apply for map building and localization in the works reported in section 2.4. In the hyperspectral image processing context, the idea of endmember is that they represent instances of pure elements. In a robotic mapping context, those endmembers could correspond to the distinct views which could be used as landmarks. This is the approach we propose and test in section 2.5.

1.2.2 Localization from 3D imaging

Until recently, acquisition of 3D measures in an environment was restricted to the range sensors described earlier or to large and complex LiDAR rigs used for 3D object reconstruction or geodesic surveying, which were not suitable for a standard mobile robotic platform. However, a new type of sensor able to acquire this type of information has been added to the spectra of sensors for mobile robots. The recent market introduction of lightweight Time of Flight

(ToF) 3D cameras [79] has opened a broad new spectrum of possibilities. Those cameras blend some characteristics of traditional range sensors with those of video cameras, providing depth information covering a wide field of view not restricted to a narrow line or cone. The data generated by them can be processed both with artificial vision techniques and other techniques used with laser range finders or LIDAR. Following the main thread of this PhD Thesis works, we wanted to try Computational Intelligence approaches which made use of the 3D measuring this kind of cameras provide for applications to the Simultaneous Localization and Mapping (SLAM) paradigm. As a first step, we have developed an hybrid neuro-evolutionary egomotion algorithm which estimates the robot's trajectory from the measures of the ToF camera, which is reported in chapter 3.

1.2.3 Multi-robot visual control

Multi-robot systems have surged as a way to fulfil more efficiently a task by cooperation between several robots. One of the objectives that multi-robot systems try to face is the operation in big, unstructured and very dynamic environments in which a multi-robot system possesses several advantages from a single robot. Also, in those environments, like shipyards or large civil engineering constructions, a typical task to be performed is the transportation of fluids with hoses. A multi-robot system designed to fulfil those tasks, in the form of a robotic hose transportation system, is the paradigmatic case of Linked System [26], with several robots attached or grabbing the hose they have to tow. Recently, the efforts in our research group have been focused towards the study of one multi-robot systems designed to perform that task. Following the main thread of this PhD Thesis, we have realized a vision based control for one such systems, opening a wide new field for further research.

1.3 Objectives

The stated goal of this PhD Thesis works is to explore the use of innovative Computational Intelligence techniques for vision based localization and mapping for mobile robots. Beyond this general objective statement, there are several more specific objectives that we wanted to fulfil. It should be noted that those specific objectives were not completely established at the start of this PhD Thesis, as we were exploring some recently, at the time,

arisen paradigms and new possible objectives emerged from the results of the previously explored ones, which resulted interesting enough to be also explored.

1.3.1 Operational objectives

As other robotics related researches, the works in this PhD Thesis required of the establishing and maintenance of a complex experimental environment in order to validate the proposed approaches. This experimental environment involves different robotic platforms and software tools which operation needs to be learned. There were, thus, several operational objectives that, although scientifically unproductive, should be fulfilled before and throughout the PhD Thesis:

- Learn how to operate the different robotic platforms.
- Learn how to operate the data acquisition hardware.
- Learn and develop the appropriate software tools to implement proposed approaches.
- Design adequate experiments to validate proposed approaches.
- Acquire adequate datasets to allow reproducibility of the results produced.

1.3.2 Scientific objectives

The specific scientific objectives of the works in this PhD Thesis relate to each of the Computational Intelligence approaches we wanted to explore:

- Test the capacity of LAMs for view storing and recognition through retrieval in a real robot implementation.
- Test the usefulness of the convex coordinates extracted with LAMs as feature vectors for view classification in a robotic mapping context.
- Test the usefulness of the endmembers induced with LAMs as landmarks in an SLAM context, developing the adequate tools for its on-line use.

- Develop an hybrid approach to the use of 3D data provided by innovative 3D ToF cameras for ego-motion estimation.
- Demonstrate a physical realization of vision based control for a multi-robot linked system in the form of a hose transportation system.

1.4 Publications resulted from this PhD Thesis works

The works in this PhD Thesis have been submitted to several conferences and journals for peer review. This has resulted in the following publications:

Related to Lattice Computing approaches to localization and mapping:

1. Ivan Villaverde, Sergio Ibáñez, Francisco Xabier Albizuri and Manuel Graña. Morphological Neural Networks for real-time vision based self localization. In Ajith Abrham, Y. Dote, T. Furuhashi, M. Köpen, A. Ohuchi, and Y. Ohsawa, editors, Soft Computing as transdisciplinary Science and Techonology, Proc. WSTST'05, volume 29/2005 of Advances in Soft Computing, pages 70–79. Springer-Verlag, 2005. ISBN 3-540-25055-7. [123]

2. Ivan Villaverde, Manuel Graña and Alicia D'Anjou. Morphological Neural Networks for localization and mapping. In Proceedings of the IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 06), pages 9–14. IEEE Press, 2006. ISBN 1-4244-0245-X. [120]

3. Ivan Villaverde, Manuel Graña and Alicia D'Anjou. Morphological Neural Networks and vision based mobile robot navigation. In Artificial Neural Networks - ICANN 2006, volume 4131 of Lecture Notes in Computer Science, pages 878–887. Springer-Verlag, 2006. ISBN 3-540-38625-4. [119]

4. Manuel Graña, Ivan Villaverde, Ramón Moreno and Francisco Xabier Albizuri. Computational Intelligence Based on Lattice Theory, chapter Convex Coordinates From Lattice Independent Sets for Visual Pattern

Recognition, pages 99–126. Springer-Verlag, 2007. ISBN 978-3-540-72686-9. [46]

5. Ivan Villaverde, Manuel Graña and Alicia D'Anjou. Morphological Independence for landmark detection in vision based SLAM. In F. Sandoval, A. Prieto, J. Cabestany, and M. Graña, editors, Proc. of IWANN 2007, volume 4507 of Lecture Notes in Computer Science, pages 847–854. Springer-Verlag, 2007. ISBN 3-540-73006-0. [121]

6. Ivan Villaverde, Manuel Graña and Jose Luis Jiménez. Lattice Independence and vision based mobile robot navigation. In Bruno Apolloni, Robert J. Howlett, and Lakhmi Jain, editors, Knowledge-Based Intelligent Information and Engineering Systems, KES 2007, volume 4693/2009 of Lecture Notes in Artificial Inteligence, pages 1196–1203. Springer-Verlag, 2007. ISBN 978-3-540-74826-7. [122]

7. Ivan Villaverde, Alicia D'Anjou and Manuel Graña. Morphological Neural Networks and vision based simultaneous localization and mapping. Integrated Computer-Aided Engineering, 14(4)(14):355–363, 2007. IOS Press. [114]

8. Ivan Villaverde and Manuel Graña. A Lattice neurocomputing approach to visual landmark identification for mobile robot navigation. Neural Processing Letters (in preparation, stimated submission November 2009).

Related to 3D localization:

1. Ivan Villaverde. 3D Camera for Mobile Robot SLAM. Ramón Moreno Jiménez, Miguel A. Veganzones, Maria Teresa García-Sebastián (Eds.) Actas de las I Jornadas de Inteligencia Computacional - JIC'07, pages 248–253, Basque Country University Editorial (2007); ISBN 978-84-9860-019-3 [113]

2. Ivan Villaverde, Zelmar Echegoyen and Manuel Graña. Neuro-evolutive system for ego-motion estimation with a 3D camera. In M. Köppen, editor, Advances in Neuro-Information Processing, volume 5506/2009 of Lecture Notes in Computer Sciences, pages 1021–1028. Springer-Verlag, 2009. [116]

3. Ivan Villaverde and Manuel Graña. A hybrid intelligent system for robot ego-motion estimation with a 3D camera. In Emilio Corchado, Ajith Abraham, and Witold Pedrycz, editors, Hybrid Artificial Intel ligence Systems, volume 5271 of Lecture Notes in Artificial Inteligence, pages 657–664. Springer-Verlag, 2008. [117]

4. Ivan Villaverde, Zelmar Echegoyen and Manuel Graña. Neuro-evolutive system for ego-motion estimation with a 3D camera. Australian Journal of Intelligent Information Systems, 10(1):59–70, 2008. ISSN 1321-2133. [115]

5. Ivan Villaverde and Manuel Graña. An improved evolutionary approach for egomotion estimation with a 3D TOF camera. In J. Mira et al., editors, Bioinspired Applications in Artificial and Natural Computation, volume 5602/2009 of Lecture Notes in Computer Science, pages 390–398. Springer, 2009. ISBN 978-3-642-02266-1. [118]

6. Ivan Villaverde and Manuel Graña. Neuro-Evolutionary mobile robot egomotion estimation with a 3D ToF camera. Neural Computing and Applications (submitted).

Related to multi-robot visual control:

1. Zelmar Echegoyen, Alicia D'Anjou, Ivan Villaverde and Manuel Graña. Towards the adaptive control of a multirobot system for an elastic hose. In Advances in Neuro-Information Processing, volume 5506/2009 of Lecture Notes in Computer Science, pages 1045–1052. Springer, 2009. [28]

2. Zelmar Echegoyen, Ivan Villaverde, Ramon Moreno, Manuel Graña, Alicia d'Anjou. Linked mobile robot control: the hose manipulation problem. Robotics and Autonomous System (in preparation, estimated submission November 2009).

Collaborations related with issues of the PhD Thesis:

1. Manuel Graña, Maite García-Sebastián, Ivan Villaverde and Elsa Fernandez. Proceedings of the Lattice-Based Modeling Workshop, in conjunction with The Sixth International Conference on Concept Lattices

and Their Applications, chapter An approach from Lattice Computing to fMRI analysis, pages 33–44. 2008. ISBN 978-80-244-2112-4 [41]

2. Manuel Graña, Ivan Villaverde, Jose Manuel López Guede and Borja Fernández. Review of hybridizations of Kalman filters with fuzzy and neural computing for mobile robot navigation. In E. Corchado, E. Oja X. Wu, A. Herrero, and B. Baruque, editors, Hybrid Artificial Intelligence Systems, volume 5572/2009 of Lecture Notes in Computer Science, pages 121–128. Springer, 2009. [44]

3. Manuel Graña, Ivan Villaverde, Jose Orlando Maldonado and Carmen Hernandez. Two Lattice Computing approaches for the unsupervised segmentation of hyperspectral images. Neurocomputing, 72(10-12):2111– 2120, 2009. [45]

1.5 Research projects

During the PhD Thesis works the candidate has been involved in several research projects focused on vision based mobile robotics, and has also collaborated in other projects where some of the techniques employed in some parts of the PhD Thesis works were also applied.

1. Estudio de métodos de visión artificial para navegación de sistemas autónomos. Universidad del País Vasco (UPV-EHU), 2003. Duración: 9 meses. Referencia: TIC2000-0739-C4-02.

2. Sistemas GIS y clasificación de imágenes hiperespectrales no supervisadas. Universidad del País Vasco (UPV-EHU), 2004. Duración: 12 meses.

3. Percepcion artificial y control de caos para robótica modular en entornos dinámicos y no estructurados (McRobs), Ministerio de Educación y Ciencia, 2006. Investigador Principal: Manuel Graña Romay. Entidades colaboradoras: Ciencias de la Computación e Inteligencia Artificial (UPV/EHU), Universidad de A Coruña, Inteligencia Artificial (UPM). Duración: 36 meses. Referencia: DPI2006-15346-C03-03.

1.6 Contributions of the PhD Thesis

The works carried out during the course of this PhD Thesis have resulted in several original contributions to the state of the art in vision based robotics. Those contributions are, in the same order in which they are presented in this report:

- Lattice Associative Memories (LAM) for visual mapping and localization: We proposed the use of LAMs for storing and identifying visual landmarks. The proposed approach was implemented and tested on a real robotic platform, and it proved to be viable for real-time self-localization even in a mildly unstructured environment.

- LAMs for feature extraction in landmark recognition: We proposed a Lattice approach using LAMs to feature extraction, using the extracted Convex Coordinates from an induced endmember set as feature vectors to identify views as landmarks. Results obtained are comparable to PCA approaches, with stronger dimensionality reduction.

- LAMs for unsupervised landmark selection in mapping: We proposed to use as visual landmarks the endmembers extracted by the Endmember Induction Heuristic Algorithm (EIHA). The approach proposed was able to obtain a good partition of the space, placing landmarks well distributed along mapped paths.

- Neuro-Evolutionary system for egomotion estimation using 3D ToF cameras: We proposed an approach for using 3D information from new ToF cameras for mobile robot ego-motion estimation. The acquired 3D point cloud was fit using Competitive Neural Networks. A registration technique based on an Evolution Strategy was developed for motion estimation. The ES was compared with other well known registration algorithms. The system's performance was comparable to the robot's odometry.

- In the field of multi-robot systems we have demonstrated the physical realization of the vision based control of a multi-robot system manipulating a hose, which falls in the class of linked multi-component robot systems, opening some new avenues for experimentation.

- Several experimental datasets have been compiled and made available to the robotics community:

  - Optical images dataset: A compilation of optical images obtained by a mobile robot's camera while performing several walks along the corridors of our building.
  - 3D ToF camera dataset: A compilation of 3D images covering exhaustively two rooms of our building, recorded with a 3D ToF camera mounted on a mobile robot.

1.7 Structure of the PhD Thesis report

This PhD Thesis report is structured following the different approaches and problems which have been explored along the works of the PhD Thesis. The main contents are divided in three chapters, reporting in each one a Computational Intelligence approach to solve a different specific problem of the more general mobile robot navigation problem. Several appendices provide additional information with the theoretical foundations of the approaches proposed, as long as descriptions of some side products of the works carried out that need to be documented. The contents of each chapter in the PhD Thesis are briefly described below:

- Chapter 2 is devoted to the application of Lattice Computing techniques based on Lattice Associative Memories (LAM) to the problems of robot self-localization and map building, both off-line and following the SLAM paradigm. The chapter has three main sections in which different applications of the LAM are proposed and experimentally validated:
  - Lattice Heteroassociative Memories (LHAM) for visual mapping and localization. LHAM are used to build a map, storing views as landmarks. LHAM Recall is used as the landmark recognition tool.
  - Lattice Autoassociative Memories (LAAMs) for feature extraction in landmark recognition. LAAMs are used to compute the convex coordinates from the endmembers of the images the robot acquires. Those convex coordinates are used as feature vectors for landmark recognition as a classification problem.
  - LAAMs for unsupervised landmark selection for topological SLAM. LAAMs are used to select on-line the most adequate views to be

used as landmarks in a topological SLAM system.

- Chapter 3 is devoted to the application of hybridization techniques, combining a Competitive Neural Network and an Evolution Strategy, to the egomotion of a mobile robot using the 3D data acquired by a ToF camera. A description of the system is presented, describing the purpose of the different method used. Some experimental results are given, along with comparisons with other techniques found in the literature.

- Chapter 4 is devoted to the physical realization of visual control for a multi-robot hose manipulation system. An experiment with our first attempts to build a vision based multi-robot system for that purpose is reported.

- Appendix A gathers the formal definitions of the Linear Mixing Model, Lattice Associative Memories and several endmember induction algorithms which are the theoretical foundations of the approaches reported in chapter 2.

- Appendix B gathers the formal definitions of the Computational Intelligence tools used in chapter 3.

- Appendix C contains a description of the hardware and software tools used in this PhD Thesis works, along with descriptions of the particular configurations of the experimental settings of the tests reported in chapters 2, 3 and 4.

- Appendix D contains a description of the optical image datasets recorded and used in chapter 2.

- Appendix E contains a description of the datasets recorded with the 3D ToF camera and used in chapter 3.

1.7.1 About the conclusions sections

Each of the main chapters in this PhD Thesis report represents a different track of research, ending with a particular conclusions section. Therefore, there is no specific chapter devoted to conclusions for the whole report.

Chapter 2

Lattice Computing approaches to localization and mapping

In this chapter we report the works done applying Lattice Computing approaches to visual localization and mapping tasks for mobile robots. We follow the historical path of our attempts to apply Lattice Associative Memories (LAM) (early called Morphological Associative Memories) to the tasks of self-localization and map building using visual information provided by the on-board video camera. The work is mainly of an experimental nature, as the theoretical foundations have been developed outside the scope of this PhD Thesis, which is more focused on physical realizations and applications of the Lattice Computation ideas.

The structure of this chapter is as follows: In section 2.1 a brief review of approaches to vision based navigation and the motivation of the approach taken is given. In section 2.2 we recall the tools and robots used in the experiments. The following three sections, 2.3, 2.4 and 2.5, present the successive approaches that we have tested during this PhD works. Each one of those sections will contain a brief introduction to the specific problem which is faced, a description of the approach and some experimental validation. Finally some conclusions and final remarks will be given in section 2.6. Additionally, appendix A contains the theoretical basis of the tools used, namely the Linear Mixing Model, Lattice Associative Memories and the endmember induction algorithms used in the experiments.

2.1 Background and motivation

As said in the introduction, navigation is the ability of an agent to move around its environment with a specific purpose [13]. It implies some knowledge of the environment, be it topological or not. A necessary ability for navigation is self-localization: the capacity of the robot to ascertain, more or less accurately, "where it is" from the information provided by its sensors. This knowledge makes possible other navigation related tasks like path planning. The basic self-localization procedure is odometry: self-sensing and keeping track of motion commands. However the uncertainties related to the environment and the robot internal status call for sensor based external confirmation of the position internal estimation.

Self-localization based on low range external sensors has been formulated in a probabilistic framework [32]. Vision based [21] and mixed [80, 96] systems are proposed to increase the sensing range and robustness. Visual self-localization methods usually are based on landmark recognition [81, 4, 11, 64, 67, 47, 99, 89]. For instance, the system described in [4] computes for each stored view a graph model representation of salient points in the image as measured by the information content of a neighborhood of the point in a gradient image. To recognize the view, the salient points in the current image are compared to the stored models. Model matching as performed in [4] is not invariant or robust against translations and rotations. Therefore each view is only recognized when the robot is within a small neighborhood of the physical position and orientation where the landmark was detected originally. Recognition in this case is not continuous, unless the stored views built up a dense map.

For the development of truly autonomous mobile robots, SLAM (Simultaneous Localization And Mapping) is a key problem for autonomous navigation [2, 23, 110], and it is nowadays one of the most alive research subjects in the robotics research community. SLAM can be defined as the ability of an agent to, starting from an unknown position in an unknown environment, build up a map of this environment and, simultaneously, self-locate inside that map. SLAM requires both automatic landmark selection while wandering [68] and robust association between observed and stored data [76].

From the computational point of view, Kalman filters are often used as basis for SLAM algorithms, but some works use Expectation Maximization techniques [109]. Maps are built using hierarchical schemes in [29, 33]. Kalman filter simplifications are proposed in [48, 49] in order to get real time

performance in outdoor environments. In [111] an extended Kalman filter is mixed with a topological map. Generalized Voronoi Graph (GVG) is used in [14] for localization and generation of a control law that guarantees complete exploration of free space.

SLAM approaches have been developed based on a wide variety of sensors, from the classical ultrasonic sensor [23] to the more modern and precise laser range finder [29, 109, 111, 12, 66]. Also, a wide variety of vision based sensors have been used, using single [19], stereo [82] or trinocular [76] vision systems. However, until recently the application of visual techniques in SLAM has been compromised by the requirements of real time operation, that are not easily meet by the computationally expensive computer vision methods.

Because of the computational cost of robust image segmentation and feature extraction processes, there is a trend towards the use of the image as a whole or after some (linear) transformation that does not impose great computational requirements. Such approaches are called "appearance based". In this chapter we report our attempts to follow this philosophy using Lattice Heteroassociative Memories (LHAM) and Lattice Autoassociative Memories (LAAM) as the main computational tools. We will try to use the images with minimal processing for the tasks of detemining the actual position of the robot and of building a topological map of the environment.

2.2 Experimental settings

Across the chapter works we have used the same kind of mobile robotic platform, the Mobile Robots' Pioneer platform, in two of its versions: a Pioneer 2 DXE and a Pioneer 3 DX. Both robots are a highly mobile platform, with a inboard computer, a PTZ controlled camera, sonar and other sensors. Some of the works have given as a side-product collections of images and data presented in the appendices D and E, which we have made public thorough the group's wiki. A detailed description of the resources used and the experimental settings can be found in appendix C.

2.3 LHAM for visual mapping and localization

The initial works of this PhD Thesis were a continuation of the initial work of [85, 87], where the use of Lattice Associative Memories (called Morphological

Associative Memories at that time) to solve the self-localization problem was proposed for the first time. Experimental results reported there [85, 87] on a set of images recorded from a robot did show the feasibility of the approach. We followed a brute force approach trying to use the LAM as an storage device for the views as the robot was advancing. The essence of the approach followed in [85, 87] was that the robot would recognize the view obtained in a spatial position while it is located near it. Therefore map building proceeds as follows: the first view is memorized, the robot advances and while the same (some) view is recognized it does not store any new view. When the system does not recognize the current view, a new input-output pair it is stored in the LHAM. The approach is holistic, because it treats the image as a unit, and appearance based, because it does not try to analyze it or extract further information from the camera image. However it has a limited capacity given by the predefined size of the output binary vector.

In the works of this PhD Thesis, a similar approach has been implemented on a Mobile Robots Inc. (formerly ActivMedia) Pioneer 2 DX mobile robotic platform. In this section some recognition results obtained in real time operation of the robot are presented. The real time operation requires more robustness than the off-line experiments, therefore the approach in [85, 87] has been modified. Dual LHAM's are introduced in order to reduce confusion between landmark views. A modular incremental approach is used, that assigns a couple of LHAM to each stored image. Besides, dual binary images are obtained to speed up the computation process, and they are computed in a way that enhances the robustness of the recognition by the corresponding LHAM storing them.

2.3.1 Description of the approach

The stated goal is to recognize, with some degree of robustness, several predetermined robot placements and orientations based on the recognition of the visual information captured by the robot. We can associate to each position an area of the physical environment where the robot recognizes this position, like in [4, 64]. The robot is supposed to wander looking forward, taking images at a steady rate. Each image corresponds to a view of the world, characterized by a physical position and orientation. Images are analyzed continuously and when a scene is recognized a certain spatial position is assumed for the robot. Robustness must cope with some variations in lighting and small rotations and translations of the camera due to the uncertainty of

the robot position, which, in its turn, is due to the uncertainties in the motion of the robot. The resulting set of positions and stored views constitute a topological map, one that may serve the robot to identify its location as a discrete position in space, assuming some uncertainty, but that does not allow to give metric information about the relation between the robot and other parts of the environment.

In this approach to visual self-localization, the focus is put on LHAMs because its size grows only linearly with the size of the input patterns, while the LAAM size grows quadratically with the size of the input patterns, which makes them not very useful for real time processing of moderate size images. The sensitivity of the LHAM to specific kinds of noise has been established [95, 91, 105], being the M memory is sensitive to erosion (which corresponds to darkening a grayscale image) while the W memory is sensitive to dilation (which corresponds to brightening the image). That means that the M memory will produce perfect recall of the desired output if the input is perturbed only by dilative noise, and will not when the input is perturbed even partially by erosive noise. The dual statement is true for the W memory. [1]

In [85, 87] it was proposed to apply structured erosions and dilations to the input patterns to increase the robustness of the recalling process. To build up a robust M memory the eroded input patterns are stored to ensure that most of the test patterns will be dilations of the corresponding stored ones. Besides, it is proposed to perform a dilation of the test pattern prior to test the recall of the memory. Again, the dual operations are applied when dealing with the W memory.

We profit on the duality of the LHAM memories to increase the robustness of our approach as follows: we use two dual LHAM to store two binary images obtained from the scene to be stored. The thresholds that define these binary dual images are set in order to increase their robustness to noise, having implicit dual erosive and dilative effects. In other words, these dual binary images are equivalent to an erosion and a dilation of a single binary image, although not a proper morphological erosion or dilation because there is no structural element.

In order to develop a system that could produce successful real-time results, it was necessary to take into account various problems. The most

[1] Erosive noise means that the noisy pattern is below the original pattern. Dilative noise means that the noisy pattern is above the original pattern. Common additive noise is both erosive and dilative, hence the difficulties of LAM to deal with the common sources of noise.

important ones were the limitation in the size of the LHAM that store the landmark images, the reduction of the computational cost, the image preprocessing and landmark selection, and the robustness against displacements and illumination changes. Each one will be discussed in turn below.

LHAM size limitation

The number of views that will be stored when building a map usually is not defined beforehand: it will depend on the speed of the robot motion, the power of the on-board computer, and the experimental path. In the original approach [85, 87], each view is encoded with an output vector selected from a set of orthogonal binary vectors. This implies that the size of the LHAM would have to be set prior to any processing. Therefore, either we decide on a maximum number of views that can be stored or we resort to a modular approach that would allow adding as many views as desired.

The proposed, implemented and tested method consists in building up a set of memories, each one storing a unique landmark image, whose output is a scalar binary variable taking $\{0, 1\}$ values. In this way, we obtain a system of unrestricted capacity that, instead of giving only one answer encoding the position, gives an affirmative or negative answer for each of the stored positions. This structure is equivalent to a single LHAM with orthogonal binary output vectors, if we decide that no position has been recognized when more than one module gives a positive response.

This kind of decomposition of the problem is not very original. It does follow the same philosophy of classifiers built and trained independently with a likelihood measure. An example of this kind of systems are the speech recognition processes, where words are modeled by independent Hidden Markov Models (HMM). Trained is performed independently for each HMM maximizing the likelihood of the correct response. In operation, the decision of which word has been recognized goes to the HMM with the maximum response. In our approach we have two disadvantages: (1) Output is binary, therefore we can not act as if the output is some kind of likelihood measure. (2) we do not have any means to break ties, so when several units give a positive response we can not select one in an optimal sense.

Preprocessing and Landmark selection

To obtain real-time performance of the self-localization process, we decide to work with binary images instead of the grayscale originals, because the binary images are easier and faster to process, and the LHAM with binary inputs were better understood at the time. Also, it was decided to use dual M and W memories simultaneously to enhance the accuracy of the detection. The goal is to reduce the number of conflicting recognitions. It was noticed that the M memories recognize the views by their dark spots, while the W memories do it by their bright spots. In terms of the basic results for LAM, M memories are robust to erosive noise, therefore dark spots are natural kernels for robust recognition. Dually, W memories are robust to dilative noise, therefore the white spots are the natural kernels for robust recognition. The size of these spots (blobs) are important to ensure the robust recognition against several noise sources. The images are processed as follows in order to enhance their recognition by the dual memories.

Given an input image $X = \{x(i,j)\}$ and thresholds θ_1 y θ_2, the dual binary (0 = black, 1 = white) images are processed applying the thresholds as follows:

$$X_1 = \{x_1(i,j) = 1 | x(i,j) > \theta_1\}; \quad X_2 = \{x_2(i,j) = 1 | x(i,j) > \theta_2\}, \quad (2.1)$$

where thresholds θ_1 y θ_2 are computed as follows:

$$\theta_1 = \bar{x} - S; \quad \theta_2 = \bar{x} + S, \quad (2.2)$$

where $\bar{x}$ is the arithmetic mean of the intensity of image X and S is its standard deviation. This selection of the thresholds ensures that the binary images have a significant amount useful data. Image X_1 detects the darkest spots of the image as black regions, while X_2 detects the lighter spots as white regions. According to the approach developed in [85, 87] X_1 will be stored in a $M_{X_1 1}$ memory, while X_2 will be stored in a $W_{X_2 1}$ memory. This method doubles the computational cost. However, the significant improvement in the results makes acceptable this excess of cost.

When an input test image is presented, both memories must give their recognition to declare the view recognized as the stored scene. The current view is declared recognized if there is one and only one positive response from the collection of scenes memorized in the current set of pairs of dual

LHAM memories $\left\{\left(M_{X_1^k 1}, W_{X_2^k 1}\right); k = 1, .., M\right\}$. If the current view is not recognized because there is not a positive response from any of the LHAM couples, then it becomes a new stored scene $\left(M_{X_1^{M+1} 1}, W_{X_2^{M+1} 1}\right)$.

Since variable size maps are the most likely result of this approach, checking each input image against each of the stored landmark images to determine the current location, the cost of the self-localization will increase with the size of the map. For a big size map, the real-time constraint may be compromised. To alleviate this problem, we test each input image only with LHAM storing images that have similar luminosity, that is, the ones that have similar average intensity. That way we obtain a steady 2.5-3.5 fps frame rate processing on a K6-II processor at 400MHz which is the CPU of the old Pioneer's on board computer.

Robustness

There are two main situations that can cause missing the recognition of a view stored from a given position when the robot is again in a nearby position:

1. There have happened illumination changes in the environment.
2. There are physical displacements from position where the stored view was taken.

The first problem, illumination changes, is already mitigated by the dynamic definition of the thresholds defined in equation 2.2, it is expected that the resulting binary images will remain also invariant to some degree of illumination changes.

When facing the second problem, two contradictory requirements collide regarding the robustness of the view recognition to position uncertainty. Recognition must be sensitive enough to discriminate between views from well-separated physical positions. At the same time, recognition must be robust to allow the identification of a position from close-by positions and illumination conditions. To ensure this kind of robustness the system profits from the LHAM's sensitivity to erosions and dilations. The erosion and dilation implied by the dual thresholds have the effect of extending the regions of the image that act as anchors for the recognition. This has a double effect: in one hand it increases robustness against translation and rotation relative to the reference position, and, on the other hand, it increases the probability of

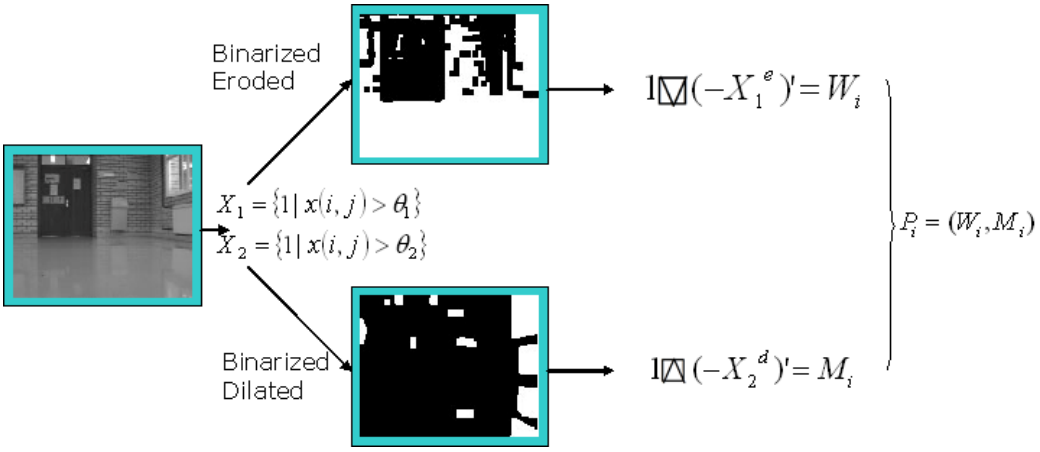


Figure 2.1: Map landmark image storage for a given robot position.

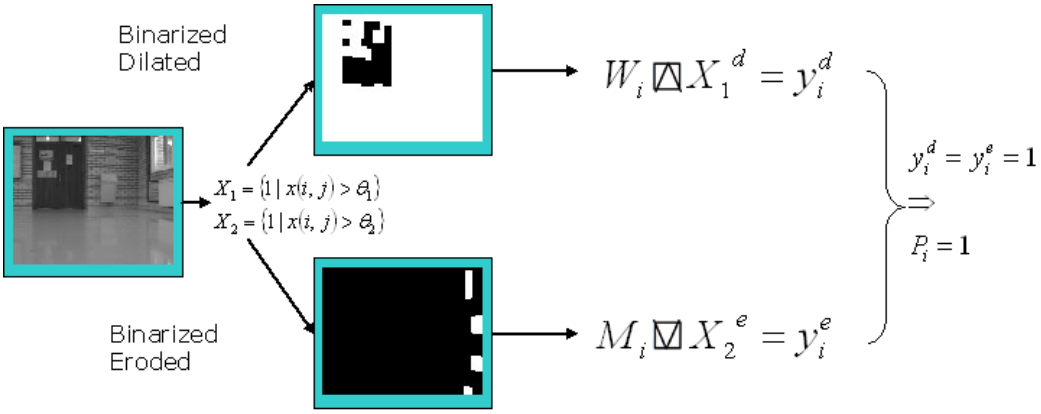
conflicting recognition. Conflicting recognition is partially solved by the use of dual memories discussed before. To erase small regions, which are unlikely to be matched, we apply a morphological closing and opening filters on the input binary images before they are compared with the stored memories.

Map creation process and self-localization

In summary, the topological map creation involves two dual processes on the image taken from each map position.

- First, to generate the dark and bright binary image landmarks to be stored by the dual LHAM modules, input grayscale images will be binarized using the thresholds defined in equation 2.1.
- Next, those binary images will be subject to the morphological operations discussed before.
- Finally, the closed dark and opened bright image will be used to build a couple of memories $\left(M_{X_1^k 1}, W_{X_2^k 1}\right)$ (figure 2.1).

During the self-localization process, each new image view is compared with each pair of dual memories. Each input image will go through processes dual to those performed for the memory building: The input image is binarized to emphasize the dark (bright) elements and the resulting binary image is opened (closed), and then tested with the W (M) memory. If the



$$P_1 \oplus P_2 \oplus .. \oplus P_i \oplus .. \oplus P_n = 1 \Rightarrow OK$$

Figure 2.2: Map position recognition. P_i denotes the response from the i-th position and $\oplus$ is the XOR binary operator.

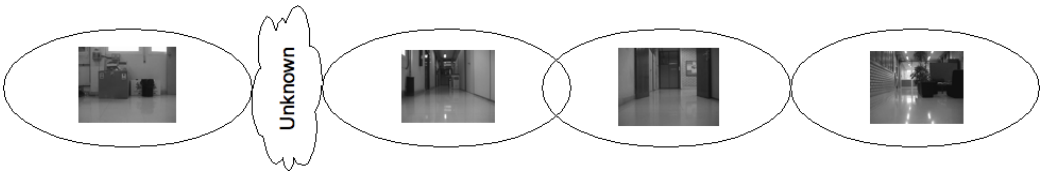


Figure 2.3: A non exhaustive map may have gaps of unknown response by the system.

response from both memories is equal to 1, the localization on that position is considered as positive (figure 2.2).

We build up a topological map taking reference positions at fixed distances, measured by odometry. Due the discrete LHAM binary response, the system may not give positive recognition when the robot is placed in positions that lie between positions whose views have been stored. The system could detect when the robot enters and departs the neighbourhood area of a stored position. The topological map will be composed, then, of a series of positions, each identified by an image which models a region around to the position where the image was taken. The map is not exhaustive and the represented regions could be adjacent, overlap or have gaps between them, as is shown in figure 2.3.

2.3.2 Experimental validation

The processes above explained were implemented on a Pioneer 2 DX robot, to perform real life experimental evaluation of the proposed approach on a real robot. The details of the implementation can be found in the appendix C. Each experiment consists in the definition of a path through our laboratory, which must be covered by the robot several times, at a speed of 250 mm/sec. In the first walk, the robot generates a map as described before. In the next walks, the robot tries to perform the self-localization based on the map generated in the first walk without any stop in its movement.

Map positions are consecutive and numbered increasingly from position 1. The visualization of the empirical results consists in the plot of the sequence of recognized positions versus time. In this representation, localization errors will be easily perceptible at first sight: a correct localization process will show a staircase growing graph, where the localization errors appear as sudden peaks and valleys.

For each view the output of the localization system can be one of four possible results:

1. It recognizes one unique position, which corresponds to the correct one.
2. It recognizes one unique wrong position.
3. It recognizes multiple positions.
4. It does not recognize any position.

The fourth case happens mostly when the robot is moving between two mapped positions. The second and third cases will be considered as localization errors and, finally, the first case will be considered as a correct localization. Below we present the results obtained on two experimental paths. In those examples, map position views were taken with a constant separation of 60 cm, as measured by odometry. The use of odometry already introduces noise in the map building process.

The first experiment consists of a straight path 3 meters long, travelled in both directions, inside the laboratory (figure 2.4). A 10 positions map was generated. Figure 2.5 shows some images corresponding to views taken by the robot while traversing the path. Notice that there is no structure in the environment, which is a relatively chaotic working laboratory environment with boxes, tables and chairs in the robot's line of sight.

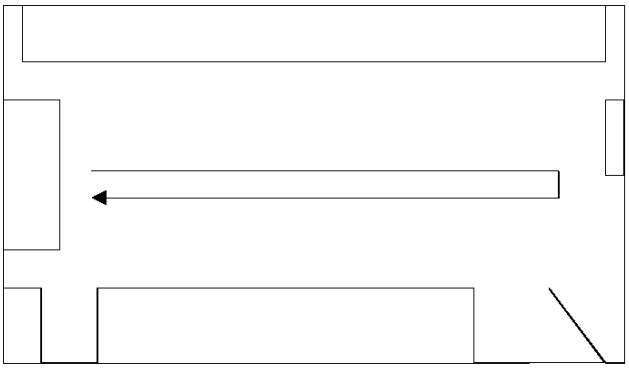


Figure 2.4: First experimental path over the plan of the laboratory.

Figure 2.5: Sample images as taken by the robot in one traversal of the first experimental path.

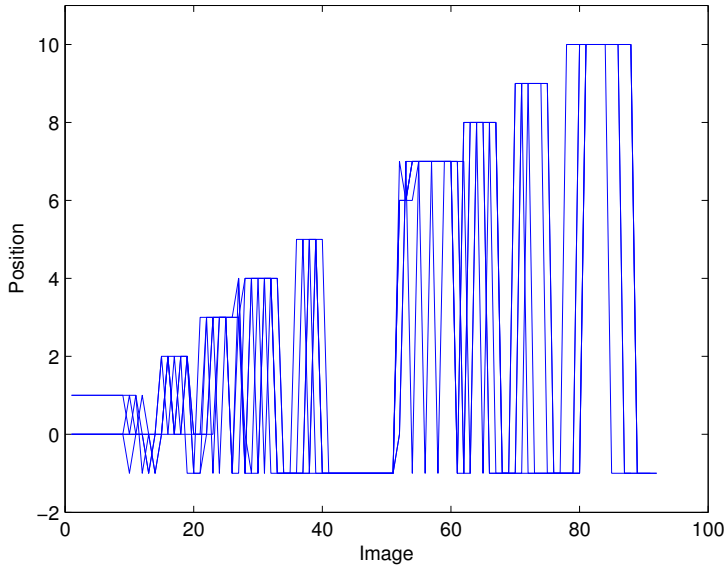


Figure 2.6: First test results graph.

In the graph of figure 2.6 overlaid plotting results from 5 test wanderings on the experimental path are shown. The values plotted in the figure have the following interpretation:

- A value of -1 means that the view was not associated with any stored position.
- A value of 0 means that the view was associated with more than one stored position.
- Remaining values are identification of the recognized stored map positions.

The growing staircase graph in figure 2.6 must be interpreted as a notable success in the localization. The wide zone without localization in the middle of the path (images 40 to 50) corresponds with the physical turning point of the robot's path. The clear shape of the overlaid plot demonstrates that the system robustly recognizes the same stored positions in the neighbourhood of the same physical test positions in all 5 attempts.

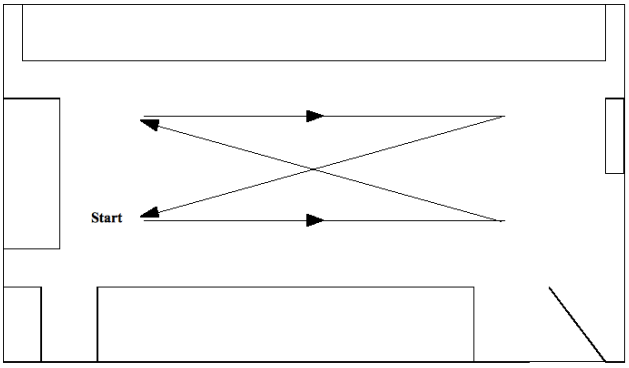


Figure 2.7: Second experimental path over the plan of the laboratory.

Average correct recognition of the mapped positions in the 5 wanderings is in the 85%-90% range, with a short interval without localizations when the robot is moving between them. Wrong localization in the worst case was a 10% of the recognition trials, and was in the 1%-5% range in the remaining cases. Worst case multiple localization errors was 24%, maintaining around 10% in the rest. The frame rate was of about 3 fps.

It can be observed that there is a concentration of both multiple localizations and wrong localizations at the start of the path. This is explained by a special characteristic of the experimental environment: at the start of the path the robot is headed towards a smooth and featureless wall, therefore the views from the initial positions are almost the same.

For the second experiment a longer path was defined. In this experiment, the generated map was of greater size than in the first experiment, and the localization more difficult. This path is formed by an X on the laboratory's floor. The cross is made of two arms of 3.6 meters long and two horizontal lines of 3 meters long (figures 2.7 and 2.8). The map is formed by 22 positions. In figure 2.9 we can see the graph obtained by overlaying the results of the 5 test wanderings. As before, the growing staircase shape of the graph indicates the general good behaviour of the localization system.

In this experiment, mapped positions where recognized with an average 90% of success, with localization errors concentrated at the start and the middle of the path. Wrong localization range between 1%-3% in all passes, while multiple localization are about 10%. As in the first experiment most of the localization errors appear at the beginning of the path. The frame rate was around 2.6 fps, despite that the map's size was double than before.

Figure 2.8: Sample images as taken by the robot in one traversal of the second experimental path.

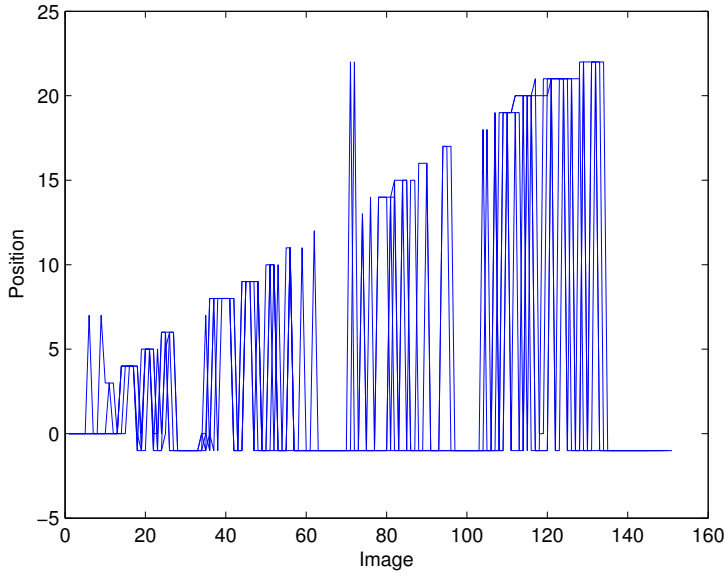


Figure 2.9: Second test results graph.

2.4 LAMs for feature extraction in landmark recognition

The approach tested in this section can be summarize as follows: we try to perform the visual recognition of designed landmark positions by a supervisedly built classifier. We test two ways to compute feature vectors from the images: the convex coordinate representation and the Principal Component Analysis (PCA) [35]. Our innovative approach to characterize the images, which the robot captures along its movement, consists in the computation of the coordinates of the image data points relative to the vertices of a convex region that covers all or most of the points that represent the images in high-dimensional space: the *convex coordinates*. These convex coordinates are the result of the linear unmixing relative to the vertices of this convex region. Therefore the dimensionality reduction depends on the degree of detail of the definition of this convex region: the number of vertices that describe it. We derived this approach from the unsupervised analysis of hyperspectral images [40, 43]through the "spectral unmixing" [59] model. Therefore the use of the name endmember for the convex set vertices. We use results from the field of LAM to induce these endmembers from the data. In [38, 39] the convex coordinates of the pixel spectra were used as feature vectors for supervised classification, with surprising good results. Here we try to extend this approach to the visual recognition of landmark views for mobile robot self-localization.

2.4.1 Description of the approach

The proposal contained in this section is a process with the following steps:

1. Extraction of the endmembers of the data sample. They provide an approximation to the vertices of the minimal convex polytope that covers the data sample.

2. Feature extraction: the sample data points are represented as linear combination of the extremal points, the coefficients are the convex coordinates.

3. Some distinguished or landmark positions to be recognized are selected by hand. The convex coordinates of the views assigned to them are

the training set for a classifier that learns to recognize them. In this section, the classifier used is a k-NN classifier.

4. The robot will identify its actual spatial position with the landmark position whose view is recognized.

First step consists in the induction of the endmembers from the data sample formed by the set of images captured by a robot along its travelled path. Those induced endmembers will be, in the second step, the basis for a linear unmixing of the image data. This linear unmixing will give as a result a vector or convex coordinates which will be used as feature vector for the images. We use for the endmembers induction the Endmember Induction Heuristic Algorithm (EIHA) [45] described in the Appendix A.4.

Map building and localization

This approach requires the full image data set that "describes" the path which is going to be mapped, which has to be recorded in a training step and processed afterwards. It is an off-line mapping algorithm. This image data set will be composed of a sequence of optical images taken at regular spaces all along the path followed by the robot.

From this training data set several positions are selected to act as landmarks. Selection of those positions can follow any arbitrary pattern, like taking positions at fixed distances or selecting positions of practical relevance. The optical views from these position are transformed into the convex coordinates computed using the endmembers extracted from the whole image data set of the path by the EIHA described in Appendix A.4. We assume that the selected positions divide the path into segments or regions, being the representative (s) (sometimes the center) of each region one or some of the positions previously selected. These path regions correspond spatial regions where the reference landmark views are expected to be smoothly recognized[2]. Ideally these regions are adjacent and dense. Figure 2.10 illustrates this idea, where each ellipsoid represents a region of the path that is smoothly recognized. In the validation experiments below, we will, in fact, use this kind of regions as the ground truth for the classifier. The images shown in each elipsoid region are the ones selected as representatives of the region in order

[2]Smooth recognition means that small displacements do not modify catastrophically the recognition.

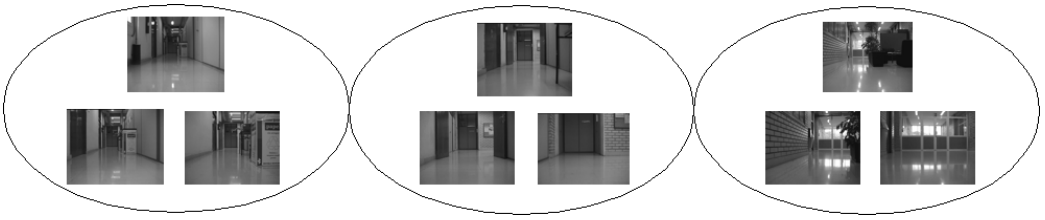


Figure 2.10: Connected regions map.

to build the classifier that will determine that an input image corresponds to a position inside this region. Map building includes the construction of the feature vector classifier using these images as the training set.

The robot self-localization process will be, thus, produced by the classification of new acquired images in one of the regions previously defined using the stored images as representatives of the region. For an input image the process is as follows:

1. Perform the linear unmixing of the image with the basis of endmembers computed from the training path image collection. The convex coordinates are the feature vector.

2. Classify the image feature vector with the classifier trained with the feature vectors of the landmark representative images.

3. When performing validation, count as success if the actual robot position falls in the region defined by the landmark classifier.

This mapping approach produces a topological map, because no metric information is stored or retrieved form the map, and the robot only uses relative, non-precise positioning.

2.4.2 Experimental validation

The experimental setup is as follows. First, a path was defined from the research group's laboratory to the stairs hall on the 3rd floor of our building. The mobile robot platform was guided manually six times following this path. In each of those trips, the odometry was recorded and the images taken from the camera, at an average of 10 frames per second, were also recorded. This process is described in detail in appendices C and D.

In the computational experiments that follow, the first trip was used to train the system parameters, and the five remaining trips were used as test sequences, simulating a real trip after the map building process. The task to perform is to recognize a hand selected set of spatial landmark positions given their respective views of the world as taken by the robot's camera. The landmark positions were selected on the floor plane, selecting places of practical relevancy, like doors to other laboratories, and the corresponding landmark views were extracted from the first trip image sequence based on odometry readings. Figure 2.11 shows the selected landmark positions as well as the paths traversed by the robot gathering the experimental data, and figure2.12 shows the landmark views as extracted from the image sequence.

Classes of images are identified for each of the selected landmark position, assigning the images in the sequences to the closest landmark map position according to its corresponding robot odometry reading. This image labelling is the ground truth for the ensuing processes. Therefore the task becomes the classification of the images into one of the map classes. The classification was done using a k-NN classifier. Each class is composed of images from the training sequence taken in robot path positions before and after the landmark map position for which it is the closest map position.

From the image sequence of the first trip a PCA transformation consisting of the first 230 eigenvectors was computed. All the following computations were done on the PCA coefficients of the images. The EIHA was applied several times to the PCA coefficients of the different image sequences. The noise tolerance parameter with best results was set, after some tuning, to $\alpha = 5$. An instance of the endmembers selected with this value is given in figure 2.13. Despite their similarity to the collection of images shown in figure 2.12, these are obtained from a completely unsupervised process, while those in figure 2.12 correspond to a human made landmark selection. An interesting question raised from those results was whether this approach could be used as an automatic landmark position identification whose landmark views correspond to the extrema found by the EIHA. This question guided the works presented in next section 2.5.

As the different EIHA runs may give different results, we give data from several repetitions of the algorithm in the tables below. Initially, for the test trips each image was classified on the map classes using 1-NN. Table 2.1 shows the landmark recognition results with our convex features, while results using PCA as features for classification are shown in table 2.2. Our convex feature approach provided a big reduction on the number of features used with some

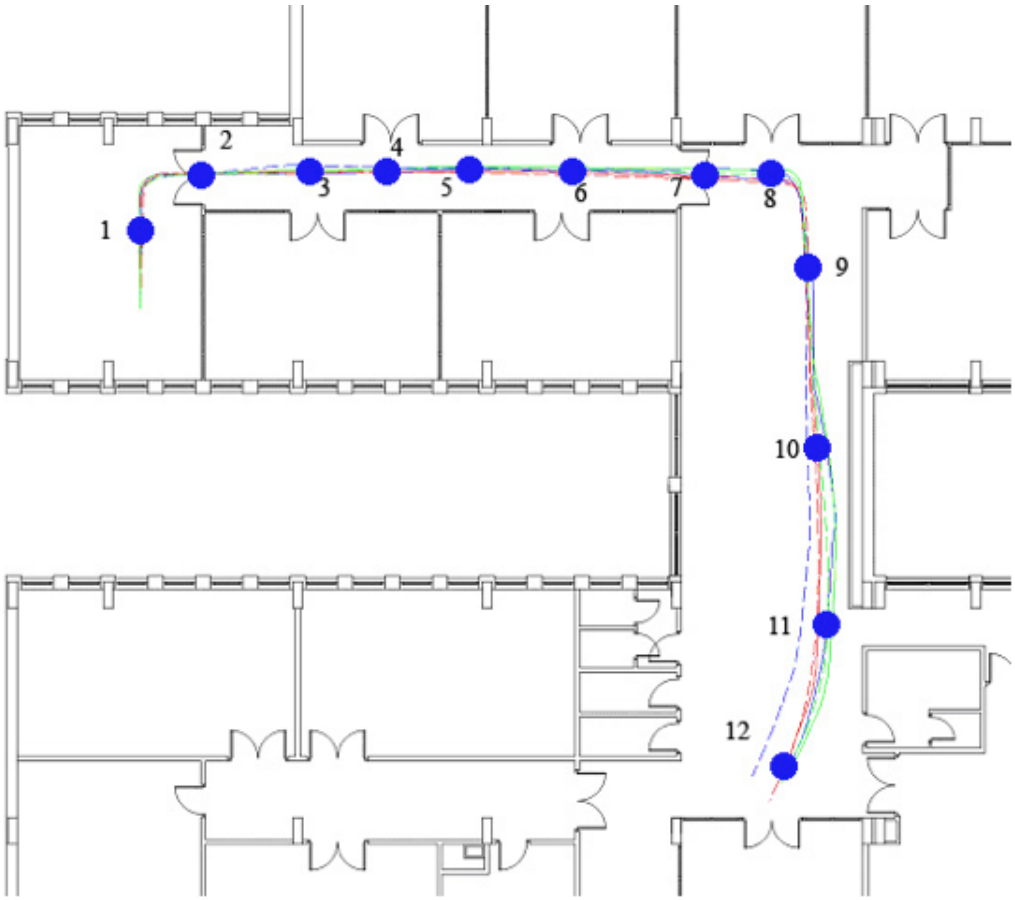


Figure 2.11: Reference positions for region partitioning. (1) Laboratory. (2) Laboratory door. (3) Laboratory door. (4) Laboratory door. (5) Printers. (6) Laboratory doors. (7) Corridor door. (8) Laboratory door. (9) Router closet. (10) Couch & coffee. (11) Crossing. (12) Hall.

Figure 2.12: The landmark views corresponding to the positions selected to build up the map.

Figure 2.13: The views corresponding to the endmembers selected in one instance of the execution of the EIHA.

improvement sometimes and comparable global average recognition results.

Ensuing computational experiments were done using 3-NN classifiers, with noticeable improvements in the results. Table 2.4 and table 2.5 show the classification success ratios for each of the sequences on the convex coordinates computed from the endmembers found by the algorithm after each run, with different values of α. The last column shows the number of extrema found at each run. Notice that the success ratio decreases as the image sequence is farther in time from the initial one used for training. Notice also that the number of extrema is in the order of 10 for the best results, meaning a dimension reduction from 230 to O(10). The Av. column shows the average success for each image sequence, including the training image sequence. For comparison we perform some further dimension reduction on the PCA coefficient vectors selecting the most significant eigenvectors for the transformation. The average results of the image classification are given in table 2.3, the average being computed as in the last column of the other tables.

Table 2.1: Landmark recognition success rate based on the convex coordinates representation of the navigation images for several runs of the EIHA with $\alpha = 5$ and using 1-NN.

Run	Train	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5	Av.	#endmember
1	0.76	0.62	0.59	0.55	0.56	0.49	0.595	14
2	0.78	0.63	0.62	0.54	0.56	0.44	0.595	13
3	0.78	0.62	0.63	0.57	0.61	0.49	0.617	12
4	0.75	0.62	0.60	0.56	0.60	0.48	0.602	12
5	0.73	0.61	0.61	0.55	0.54	0.50	0.59	14
6	0.75	0.63	0.59	0.55	0.59	0.47	0.597	13
7	0.78	0.65	0.64	0.53	0.56	0.48	0.607	12
8	0.75	0.63	0.59	0.51	0.51	0.49	0.58	11
9	0.74	0.63	0.61	0.54	0.57	0.46	0.592	10
10	0.73	0.58	0.58	0.51	0.53	0.43	0.56	11
Av.	0.755	0.622	0.606	0.541	0.563	0.473	0.594	

The approach to endmember extraction described in the algorithm A.3 described in appendix A.6 was also tested. Table 2.6 shows the results of the convex coordinate approach with varying number of endmembers selected applying the algorithm A.3. The best results are obtained when selecting 10 and 15 extrema, which give results comparable to the PCA. Contrary to the

Table 2.2: Landmark recognition averaged success rates based on the PCA representation of the navigation images for several sets of eigenvectors selected, using 1-NN.

PCA 197	PCA 150	PCA 100	PCA 50	PCA 30	PCA 10
0.61	0.59	0.60	0.61	0.62	0.61

Table 2.3: Landmark recognition success rate based on the PCA representation of the navigation images for several sets of eigenvectors selected, using 3-NN.

Set	Train	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5	Av.
PCA 197	0.95	0.84	0.76	0.65	0.79	0.77	0.793
PCA 150	0.95	0.84	0.76	0.65	0.78	0.76	0.79
PCA 100	0.95	0.85	0.76	0.65	0.79	0.76	0.793
PCA 50	0.95	0.85	0.76	0.65	0.78	0.78	0.795
PCA 30	0.96	0.87	0.77	0.64	0.78	0.78	0.8
PCA 10	0.96	0.86	0.78	0.66	0.76	0.73	0.792
Av.	0.953	0.852	0.765	0.65	0.78	0.763	0.794

Table 2.4: Landmark recognition success rate based on the convex coordinates representation of the navigation images for several runs of the EIHA with $\alpha = 6$ and using 3-NN.

Run	Train	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5	Av.	#endmember
1	0.92	0.78	0.74	0.66	0.68	0.62	0.733	8
2	0.95	0.77	0.73	0.74	0.72	0.62	0.755	7
3	0.95	0.83	0.73	0.67	0.72	0.64	0.757	8
4	0.94	0.78	0.71	0.67	0.69	0.64	0.738	7
5	0.93	0.76	0.71	0.65	0.67	0.62	0.723	7
6	0.93	0.77	0.72	0.69	0.68	0.57	0.727	8
7	0.95	0.78	0.70	0.61	0.66	0.62	0.72	8
8	0.95	0.78	0.69	0.58	0.69	0.62	0.718	8
9	0.93	0.80	0.73	0.70	0.73	0.63	0.753	8
10	0.94	0.80	0.73	0.65	0.69	0.67	0.747	9
Av.	0.939	0.785	0.719	0.662	0.693	0.625	0.737	

Table 2.5: Landmark recognition success rate based on the convex coordinates representation of the navigation images for several runs of the EIHA with $\alpha = 5$ and using 3-NN.

Run	Train	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5	Av.	#endmember
1	0.94	0.81	0.76	0.72	0.73	0.67	0.772	13
2	0.94	0.85	0.77	0.69	0.78	0.71	0.79	14
3	0.94	0.84	0.75	0.70	0.75	0.74	0.787	13
4	0.94	0.83	0.71	0.63	0.73	0.67	0.752	14
5	0.94	0.85	0.79	0.69	0.78	0.72	0.795	12
6	0.93	0.80	0.70	0.67	0.69	0.70	0.748	12
7	0.94	0.83	0.71	0.59	0.70	0.66	0.738	12
8	0.93	0.82	0.76	0.69	0.74	0.66	0.767	12
9	0.94	0.79	0.73	0.64	0.70	0.63	0.738	14
10	0.92	0.79	0.70	0.63	0.65	0.60	0.715	12
Av.	0.936	0.821	0.738	0.665	0.725	0.676	0.76	

EIHA, this algorithm does not have any random component, so there is no need to repeat the computation. Also the number of extrema is a parameter of the algorithm. The results with both algorithms are very similar.

The comparison of the results in tables 2.4, 2.5, 2.6 and 2.3 shows that there is at least an instance of the convex coordinates features which improves the best results of the PCA, and that the convex coordinates are comparable or improve the results of the PCA with a the same or much stronger dimensionality reduction. Although these results are much less spectacular than the ones reported in [38, 39], they confirm the potential usefulness of convex coordinates as a feature selection algorithm.

2.5 LAMs for unsupervised landmark selection for SLAM

In this section we try to approach the problem of visual SLAM as a coupled problem of automatic visual landmark extraction and classification of the robot camera views. The approach is non metric in the sense that precise spatial information is not extracted from the images. As in the previous section, the result of the process is a topological map where nodes are given

Table 2.6: Landmark recognition success rate based on the convex coordinates representation of the navigation images for several numbers of endmembers extracted and using 3-NN and the algorithm A.3.

#endmember	Train	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5	Av.
5	0.96	0.79	0.74	0.64	0.71	0.61	0.742
10	0.96	0.80	0.76	0.61	0.80	0.72	0.775
15	0.96	0.80	0.74	0.66	0.79	0.69	0.773
20	0.96	0.80	0.76	0.65	0.81	0.67	0.775
25	0.96	0.78	0.72	0.62	0.74	0.68	0.75
30	0.96	0.81	0.73	0.60	0.75	0.69	0.757
Av.	0.96	0.797	0.742	0.63	0.767	0.677	0.762

by the landmark images and the transitions between nodes happen when a landmark view is either discovered or recognized. This approach is a step ahead towards autonomous systems relative to the works presented in the previous section.

In this new approach, we will not compute the convex coordinates for two reasons. First, the main interest here is the usefulness of the endmembers obtained by this approach as landmarks for SLAM. Landmarks are the nodes of a qualitative map which is constructed as the robot moves on, as in previous sections. Once the landmarks are identified, the localization of the robot is performed computing the distance from the present view to the landmark views. This distance is computed on the raw image feature space. The second reason is that because of the on-line nature of the SLAM process it is not possible to use statistical information or information that could be obtained from a representative data sample. That means that we can not use PCA or convex coordinate features, because re-estimation will modify the representation of the images and invalidate previous decisions about the landmark selection. The on-line condition also forces modifications in the algorithm for endmember induction from the data, which are specified in section A.5 of Appendix A.

2.5.1 Experimental validation

The stated goal is to automatically detect from the visual data stream a set of views that can be considered as landmarks for visual SLAM. To have

some experimental setting where the experiments would be reproducible and the effects of algorithm design could be assessed, the works have been done on the same recorded image sequences than in 2.4, described in appendix D. The robot platform used is a Pioneer 2 DX, as in previous works, with detailed description of the experimental settings in appendix C. In more general terms, the experimental setup is as follows:

First, five paths on different floors of our building were selected. The mobile robot platform was guided manually six times following this path. During each of those wanderings, the robot recorded the odometry and the images taken from its camera, at an average of 1 frames each 5-6 cm. Figure 2.14 shows the first wandering on each path overlaid to the floor plan. Second and third paths correspond to traveling in the opposite direction of the same path. Fourth and fifth path are also opposite direction traveling.

Although odometry introduces some errors, we can associate for validation purposes the selected views and an area of influence where the robot would recognize this position. This association may be used as the ground truth for quantitative evaluations. Usually the views that must be assigned to a given landmark view are the ones corresponding to positions closer to the landmark position according to the euclidean distance between positions. This introduces some bias in the validation, because the landmarks were not selected trying to minimize this classification error. Classification results interpretation must take this bias into account. The qualitative validation consists on the inspection of the assignment of images to landmarks. Plotting this assignment, the best result is a pure staircase shape, meaning that no confusion between views have been detected, once a view is recognized the robot continues steadily to recognize it until the new landmark is recognized.

In section 2.4 the PCA transformation of the images was performed, allowing dimension reduction and introducing some additional robustness, because the PCA dimensional reduction implies some noise cleaning. For the used algorithm, PCA dimension reduction had the advantage of removing irrelevant morphological independence due to small translations in the image. It is therefore desirable to have some dimension reduction algorithm applied on the data. However, in this case the PCA can not be computed, because we can neither estimate the mean and covariance matrix from *a priori* information, nor use an adaptive estimation. Instead, the Discrete Cosine Transform (DCT) of the images was computed as feature extraction, keeping the 10x10 lower frequency coefficients, located on the upper left corner of the transform. This feature extraction does not depend on image stream statistics and the

Table 2.7: Landmark recognition success rate based on the DCT low frequencies and the assignment of images to landmarks based on the odometry.

	W1	W2	W3	W4	W5	W 6
Path 1	0.83	0.75	0.76	0.60	0.69	0.64
Path 2	0.84	0.68	0.74	0.76	0.59	0.67
Path 3	0.80	0.66	0.48	0.76	0.71	0.65
Path 4	0.80	0.49	0.39	0.76	0.41	0.67
Path 5	0.81	0.72	0.69	0.77	0.63	0.57

resulting feature vector components can also be accepted to move around zero, allowing the application of endmember extraction algorithm without an estimation of the population mean.

To obtain the landmark positions shown as red circles in figure 2.14, the modified endmember extraction algorithm was applied to the feature vectors of the images in the first wandering from each path. The feature vectors consist of the 10x10 lower frequency coefficients of the Discrete Cosine Transform of each image. Figures 2.15 to 2.19 show the images corresponding to landmarks detected in the five paths tested. To obtain a qualitative validation of the approach we plot the closest landmark to each image in the recorded sequence, computed over the euclidean distance of the feature vectors. These plots appear in figure 2.20 for each of the selected paths. The staircase shape of the plots demonstrates qualitatively that the selected landmarks adequately characterize the path allowing for a steady recognition of each landmark image along the first wandering on the path.

For SLAM purposes it is necessary that the landmarks selected will be recognized in the future, so that the robot could be able to detect its position relative to the qualitative non metric map constructed from the first wandering on the path. The recognition rate was computed based on the assignment of images to landmarks according to the odometric information. As commented before, this introduces some bias, because the landmarks were not selected to perform this task, however we think that this recognition ratio across wanderings is informative of the future self localization of the robot based on the selected landmarks.

The first appreciation about the results in table 2.7 is a degradation of the recognition as the time between the capture of the sequence used for the landmark selection and the sequence used for the actual recognition increases.

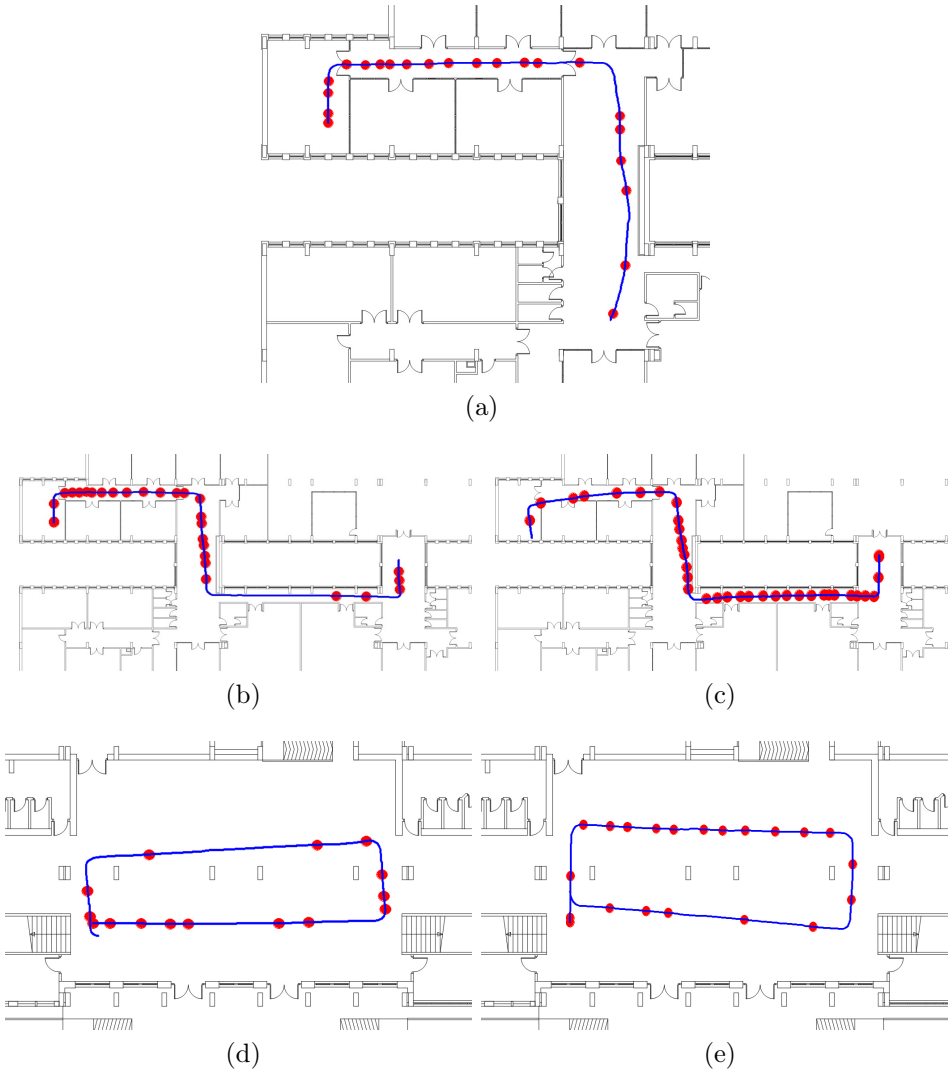


Figure 2.14: The first wandering on each path. Red circles correspond with the position of each view detected as a landmark.

Figure 2.15: Images corresponding to the landmarks detected in first path shown in figure 2.14a.

Table 2.8: Landmark recognition success rate based on the DCT low frequencies and the assignment of images to landmarks based on the odometry. Filtering of the decision as described in the text.

	W1	W2	W3	W4	W5	W 6
Path 1	0.86	0.80	0.81	0.70	0.77	0.78
Path 2	0.87	0.75	0.82	0.83	0.70	0.75
Path 3	0.83	0.71	0.51	0.79	0.78	0.74
Path 4	0.82	0.51	0.39	0.80	0.41	0.77
Path 5	0.85	0.77	0.75	0.81	0.73	0.60

Figure 2.16: Images corresponding to the landmarks detected in first path shown in figure 2.14b.

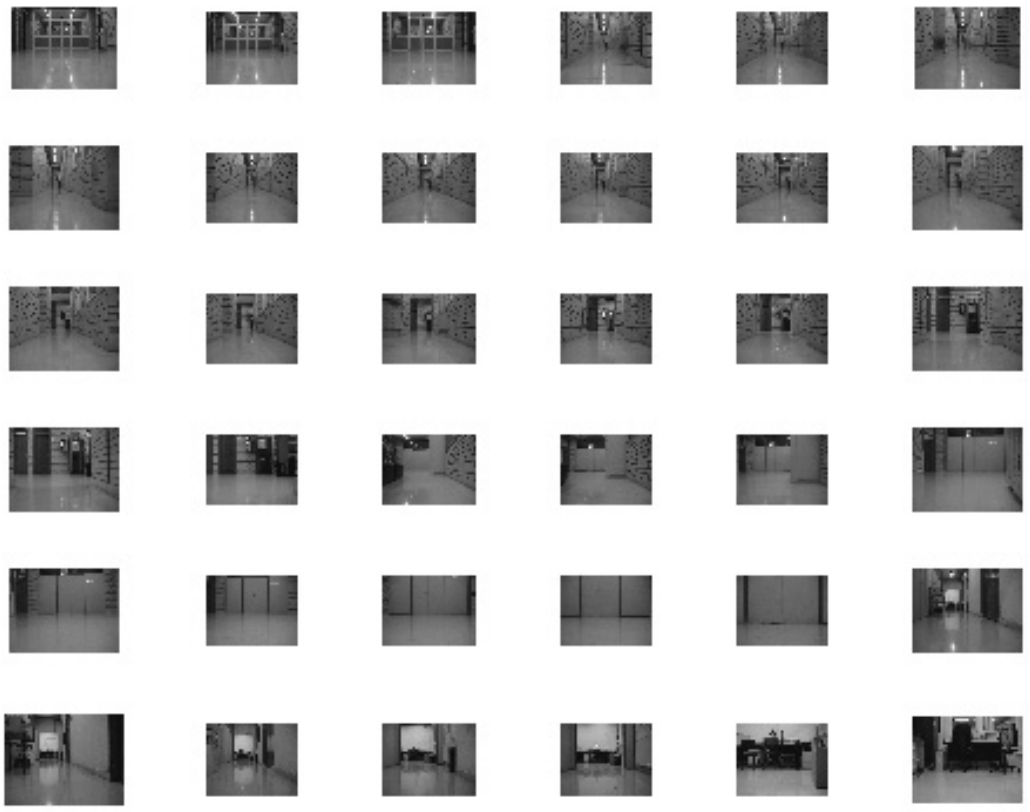


Figure 2.17: Images corresponding to the landmarks detected in first path shown in figure 2.14c.

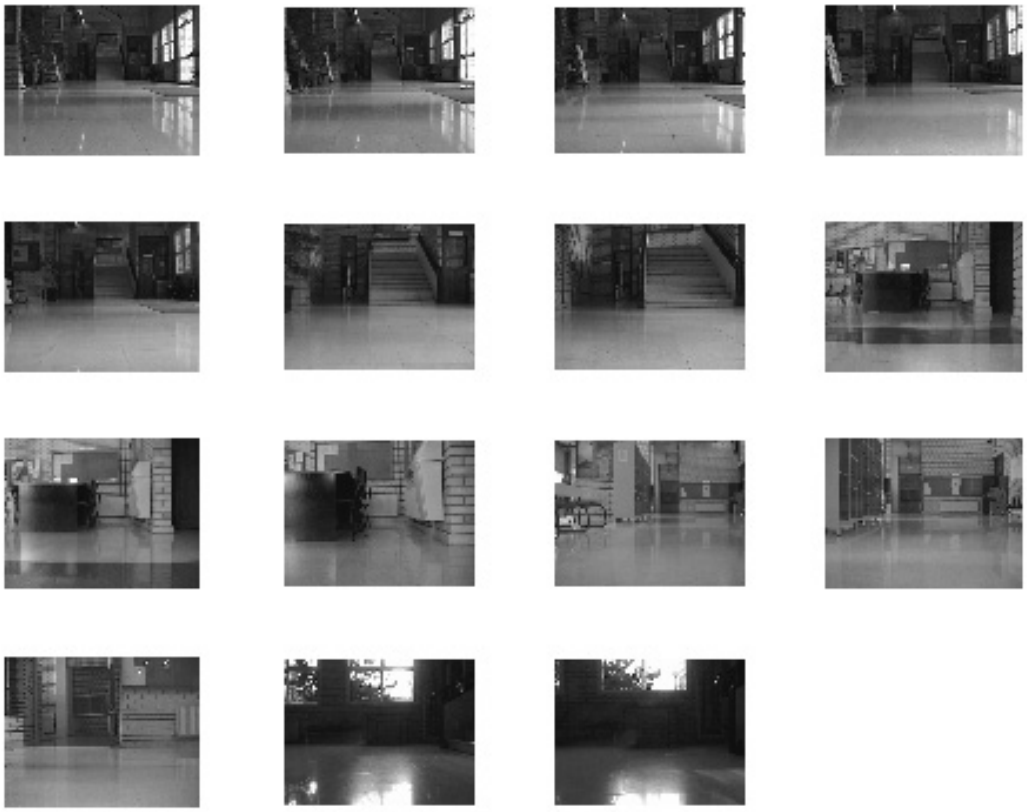


Figure 2.18: Images corresponding to the landmarks detected in first path shown in figure 2.14d.

Figure 2.19: Images corresponding to the landmarks detected in first path shown in figure 2.14e.

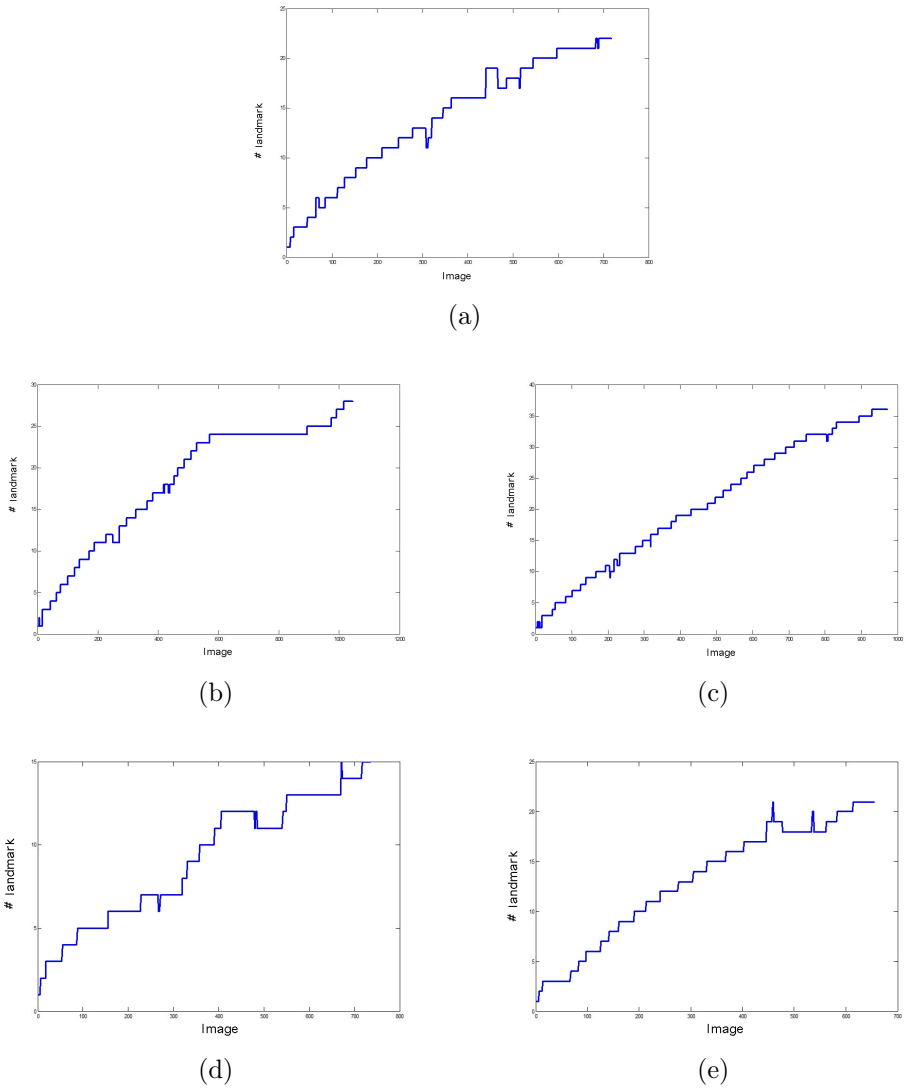


Figure 2.20: Plot of the landmark recognition for the first wandering of each path.

The results decrease for the successive wanderings. This is consistent with the classification results of previous sections. Overall the results in table 2.7 may appear to be low from a pure classification point of view. However, several factors must be taken into account in order to moderate these negative conclussions. The first is that the ground truth partition of the images has been done on the basis of the odometry dividing the path into segments where the landmark lies in the middle. The landmark selection was not meant to be optimal for this setting. In fact, the results for the first wanderings are rather low, despite the fact that the landmarks perform a good partition of the image sequence for each path, as shown in figure 2.14. The second factor is the different illumination conditions that prevailed in some wanderings over some paths. Paths 1 to 3 were more stable regarding illumination conditions. In fact path 1 was recorded during the night with artificial light. Accordingly it is the most stable of all paths. Path 4 and 5 were recorded during a windy and cloudy day, so that the illumination conditions vary widely between wanderings. This explains the big fluctuations on the results for path 4. The use of illumination invariant features probably would help to alleviate the problem.

The results of table 2.7 may be improved as shown in table 2.8 by the introduction of a simple mechanism similar to a discrete Kalman filter or a naive Hidden Markov Model. The mechanism performs the voting between the three last recognitions (not to be confused with the 3-NN classification of the current image). The winning landmark is the one recognized. If we visualize the landmark based map as a graph, whose nodes are the landmark views and the arcs the transitions between recognition, then we restrict the transitions to those between neighboring nodes whose index distance is two or less. This simple process improves the recognition results in almost all cases. The most important effect is that the recognition degradation with time is lower than in the original classification. That is, the decreases in recognition for successive wanderings on a path are smaller than in table 2.7.

2.6 Summary and conclusions

The works reported in this chapter follow a historical sequence, going from first brute force attempts into more sophisticated applications of the LAMs. First we report in section 2.3 on the real-time experimental results on the use of LHAM for landmark image storage and recognition confirming theoretical

and simulation results of previous works [85, 87]. The system responded well on most situations, failing only on quite hard conditions. It has a very low computational cost that allows its implementation on the on-board computer of the Pioneer robot (AMD K6-II, 400MHz). It seems that it can be useful on less structured environments, and embedded in more complex navigation systems of autonomous mobile robots.

In order to build up more exhaustive maps to obtain spatially dense recognition, it was needed to devise an unsupervised method. While the robot is map-making, instead of taking positions at fixed distances, new ones could be stored when the new view does not match with any other stored one, avoiding also very similar conflicting memories. This eliminated much of the discontinuity of the map (when it goes out of a recognized position to an unrecognized one, it will immediately store it) and reduced the problem of recognizing multiple stored positions from one view, since stored locations' difference will be assured and the intersection between the physical recognition areas will be minimized.

Results in section 2.4 show that the convex coordinates of the data points based on the endmembers induced by the EIHA algorithm can be used as features for pattern classification. Mobile robot self-localization is stated there as the classification of images taken from the robot's camera by a classifier trained on the feature vectors of hand selected landmark positions. PCA and convex coordinates feature vectors are used for such classification. Results show that this approach improves the PCA one in some runs, while on average performs similarly like most PCA with different eigenvector selections tried. However a big decrease of the number of features needed to obtain the results can be appreciated using convex coordinates as feature vectors. Also, as a result from the observation of the selected endmember sets, the use of the EIHA for the detection of endmembers as an automatic landmark selection tool was explored, leading to the works reported in section 2.5.

In that last section, the algorithm for endmember induction has been modified to serve for the automatic detection of image landmarks from the image stream provided by the on-board camera while the mobile robot is moving. The recognition of these landmarks allow for the future self localization of the robot. The approach is not metric, because we do not obtain spatial measurements from the images, instead it can be considered as a topological map.

This approach has been tested over a set of five paths, which the robot has covered six times. The qualitative results given by the plot of the assign-

ment of images taken on the path to the selected landmarks are good. The quantitative results, given by the recognition rate against a ground truth classification based on odometry, are encouraging. As the main problems seem to arise from varying illumination conditions, further work must be addressed to incorporate illumination robust features into the system.

Chapter 3

Localization from 3D imaging

In this chapter, we report the works made in this PhD thesis towards an innovative neuro-evolutionary system for egomotion estimation with a 3D Time of Flight (ToF) camera. This system is composed of two main modules following a preprocessing step. The first module is a Competitive Neural Network which computes a Vector Quantization of the preprocessed camera 3D point cloud. The second module is an Evolution Strategy which estimates the robot motion parameters by performing a registration process, searching on the space of linear transformations, restricted to the translation and rotation, between the codebooks obtained for successive camera readings. The fitness function is the matching error between the predicted and the observed codebook corresponding to the next camera readings.

The structure of this chapter is as follows: In section 3.1 some background and motivations will be given. In the following section 3.2, the proposed system is described. The system is tested in section 3.3, where several experiments are presented with comparisons between our and other well known registration algorithms, and their results discussed. Finally, some conclusions are provided in section 3.4.

3.1 Background and motivation

In spite of impressive development and improvements in robotics, both in hardware and algorithm and techniques, few changes have happen in the spectra of available of sensors used in mobile robotics, which still are more or less the same as in the origins of the field. Video cameras, laser range

finders, sonar or infra-red sensors keep being the main ways to obtain information about the robots environment, with improvements based mainly in augmented range, reductions on its size, weight or consumption or new ways of processing acquired information thanks to the higher computational power of newer hardware.

The recent market introduction of lightweight Time of Flight (ToF) 3D cameras [79] which can be mounted on mobile robots opens a broad new spectrum of possibilities. Those cameras blend some characteristics of traditional range sensors with those of video cameras, providing depth information covering a wide field of view not restricted to a narrow line or cone. The data generated by them can be processed both with artificial vision techniques and other techniques used with laser range finders or LiDAR. The use of these cameras in mobile robotics is still a mostly unexplored area, with very few pioneering efforts like [124], where the Swissranger 3000 is used to extract edges to build symbolic representations of the environment, or [50], where Local Linear Embedding is used to model the geometry of data points as a way of generating view invariant references.

We are focusing our efforts on the use of ToF 3D cameras to perform Simultaneous Localization and Mapping (SLAM) [23, 110]. The first step toward this objective is the robot egomotion estimation from the 3D camera readings. The registration of 3D data has a long tradition in computer graphics and geoscience for surface reconstruction [98]. Point clouds are matched in order to obtain an estimation of the camera displacement, so several partial views of the surface could be accurately integrated in order to achieve a full model reconstruction. Since the basic problem is similar to the egomotion estimation problem in mobile robotics, we have studied the development of a neuro-evolutionary system which is able to perform real time registration of the 3D camera data in order to estimate the trajectory of a robot.

Our neuro-evolutionary system is composed of two main modules plus a preprocessing step. The first module computes the approximation of the preprocessed camera 3D data by means of a Competitive Neural Network (CNN). This approximation is a Vector Quantization of the 3D data encoded by a codebook composed of a set of 3D codevectors. We have evaluated the Self-Organizing Map (SOM) and the Neural Gas (NG) neural architectures for this task and we have found that the NG is more appropriate. The second module is an Evolution Strategy (ES) [9, 88] which performs the task of estimating the motion parameters by searching on the space of linear transformations restricted to the translation and rotation, akin to the registration

problem, applied on the codebooks obtained by the CNN on the point clouds given by successive camera readings. The fitness function is the matching error between the predicted and the observed codebook corresponding to the next camera readings.

3.2 Description of the system

Our approach is composed of several modules, which will be described next. The first module is a preprocessing step, described in section 3.2.1, along with a brief overview of the ToF camera, the data it provides and the justification of this preprocessing. Next a module that fits a codebook over the data by means of a Competitive Neural Network will be presented in section 3.2.2. That codebook will be used as input for the egomotion estimation performed by an Evolution Strategy, as described in section 3.2.3. The algorithm 3.1 specifies one time step in the path estimation of the robot by the proposed neuro-evolutionary system.

3.2.1 Sensor data and preprocessing

In our work, the 3D data required for the egomotion estimation is obtained through a Swissranger SR-3000 [79] 3D camera mounted on a Pioneer 3 robot. A more complete description of this camera and its working principle can be found in section C.2.4 of appendix C. Some of its characteristics determine how its data must be processed. The SR-3000 is based on phase-measuring Time of Flight principle [62]. It uses an array of near-infrared LEDs in order to illuminate the scene. Knowing the amplitude of the wavelength of the light emitted, it can determine the distance to objects measuring the phase of the reflected light, within a distance range determined by the wavelength and the frequency at which IR pulses are emitted. Objects lying farther than this distance may be detected as close to the camera, due to the periodic nature of the illumination source. This kind of ambiguous readings must be removed. Figure 3.1 shows the conventional visible wavelength image obtained from a specific robot position and orientation. Data from this position will be used in the following illustrations. Figure 3.2 shows the raw data provided by the 3D camera, which consists of two matrices storing at each pixel site i the measured distance D_i and the intensity I_i of the reflected infrared light. Since the optical characteristics of the camera are known, each pixel

Figure 3.1: Reference visible wavelength image from the optical camera view.

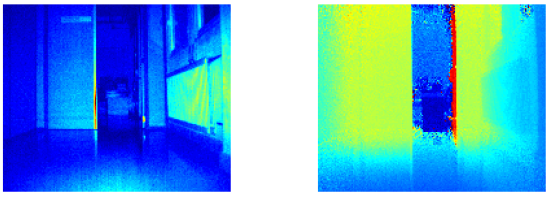


Figure 3.2: Near-infrared Intensity (left) and distance (right) images from the 3D camera. Note that the distance measured beyond the door is very low (dark blue) due an ambiguous reading.

has corresponding fixed azimuth and zenith angles in spherical coordinates. Thus, we can transform the distance image to more appropriate Cartesian coordinates (shown in figure 3.3) and obtain a cloud of 25344 3D points, representing the camera measurements in its field of view of 47.5 x 39.6 degrees.

This 3D point cloud is extremely noisy and it is necessary a specific filtering process in order to retain really informational points. As can be seen in figures 3.2 and 3.3, most of the noise comes from measurements beyond the non-ambiguity distance range: far away surfaces (like those from the space that is through the door) appear to be very close to the camera, forming a dense cone near the vertex of the point cloud. Also, pixels without any object in their line of sight present measurements spread along all distances, caused by specular reflections. As both types of points are supposed to be far away, the reflection intensity should be very low. Taking this into account, we define a reliability coefficient R_i for each pixel i, computed as:

$$R_i = I_i \times D_i. \tag{3.1}$$

Filtering of the point cloud is based on the following observation: nearby surfaces reflect light with more strength than a far away one, so it can be expected some kind of balance which will keep the values of R_i of valid measurements clustered while ambiguous measurements will have outlying low values of R_i. In figure 3.4 we show the result of applying this kind of

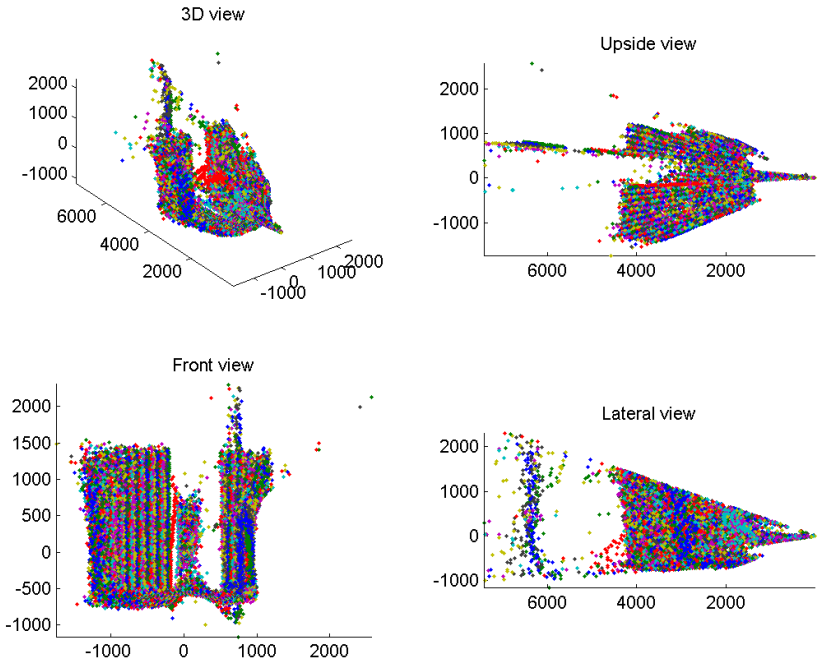


Figure 3.3: Plotting of the cloud of points in 3D Cartesian coordinates extracted from the distance image provided by the ToF 3D camera.

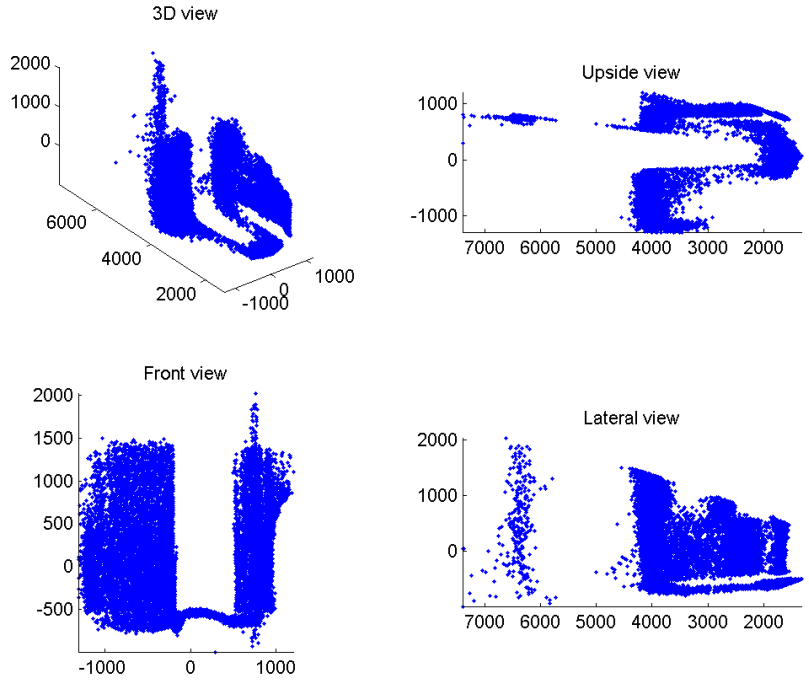


Figure 3.4: Plotting of the cloud of points in 3D Cartesian coordinates extracted from the distance image provided by the TOF 3D camera after filtering out ambiguous readings. Relevant surfaces can be appreciated.

filtering to the data in figure 3.3. It can be appreciated that most ambiguous measurements have been successfully filtered.

A final minor optimization was done in the data preprocessing step. Since the floor does not provide any useful matching information (the floor is a featureless uniform surface in which new viewed parts are indistinguishable from the ones just left behind), we remove it. It has the additional benefit of increasing the density of codevectors in the areas of interests computed by the CNN module. The floor filtering is performed just by eliminating every point below a *safe* height. In this case, an height of 10 mm. was chosen, since any object of this height can be easily overran by the robot.

3.2.2 Competitive Neural Network module

The point cloud size resulting from the above filtering process is usually still greater than 15.000 points. Processing a point cloud this size is too costly

for real-time operation, therefore some data reduction technique is required. Also, the surfaces in the point cloud have some uncertainty, which increases with distance, which should be desired to avoid. That is, we need a procedure that both reduces that problem size and smooths the data, removing additive noise.

The way to face both problems simultaneously came by the use of competitive neural networks that perform some Vector Quantization process to fit the point cloud. The objective of this system's module was to obtain a codebook that (1) kept the spatial shape of the point cloud and, hopefully, of the objects in the environment, and (2) at the same time reduces the size of the data set to a fixed, manageable, number of points. The codebook obtained will be the input data used for the spatial transformation estimation module.

For this purpose we have tried first the Self-Organizing Map (SOM) [61]. Figure 3.5 shows a SOM generated by fitting a filtered point cloud acquired by the 3D camera at a given position. This approach, however presented several strong shortcomings. In the maps calculated by the SOM, nodes lie in a fixed grid. After training, some of those nodes can fall in spaces in which there are no actual surface points in the real environment, because of the topology preservation property of the SOM. On the positive side, we may work under the assumption that the same node in the two SOMs extracted from two consecutive data captures will roughly correspond to the same spatial region of the 3D surface, which makes a lot easier the computation of the matching distance between grids as a node-to-node distance. This assumption will only hold if the movement is smooth enough. In fact, it needs to be so smooth that the approach becomes useless in the practice. Robot motion is therefore as additional source of problems. Since we are acquiring the 3D measurements from a mobile robot, traveling at a reasonable speed, some of the measurements in each frame corresponds to environment surfaces patches that were out of range in the previous frame, while some of the previously measured surface patches would fall behind in the new frame. That is, the physical surfaces imaged in two consecutive frames may overlap only partially, thus the ToF 3D Camera readings will inevitably have only optimal partial matches.

We have found that all the issues raised by the use of SOM are better tackled with the use of Neural Gas (NG) networks [69, 70]. The NG networks were developed in an attempt to achieve an optimal utilization of all neural units and as a flexible network able to quantize those topologically hetero-

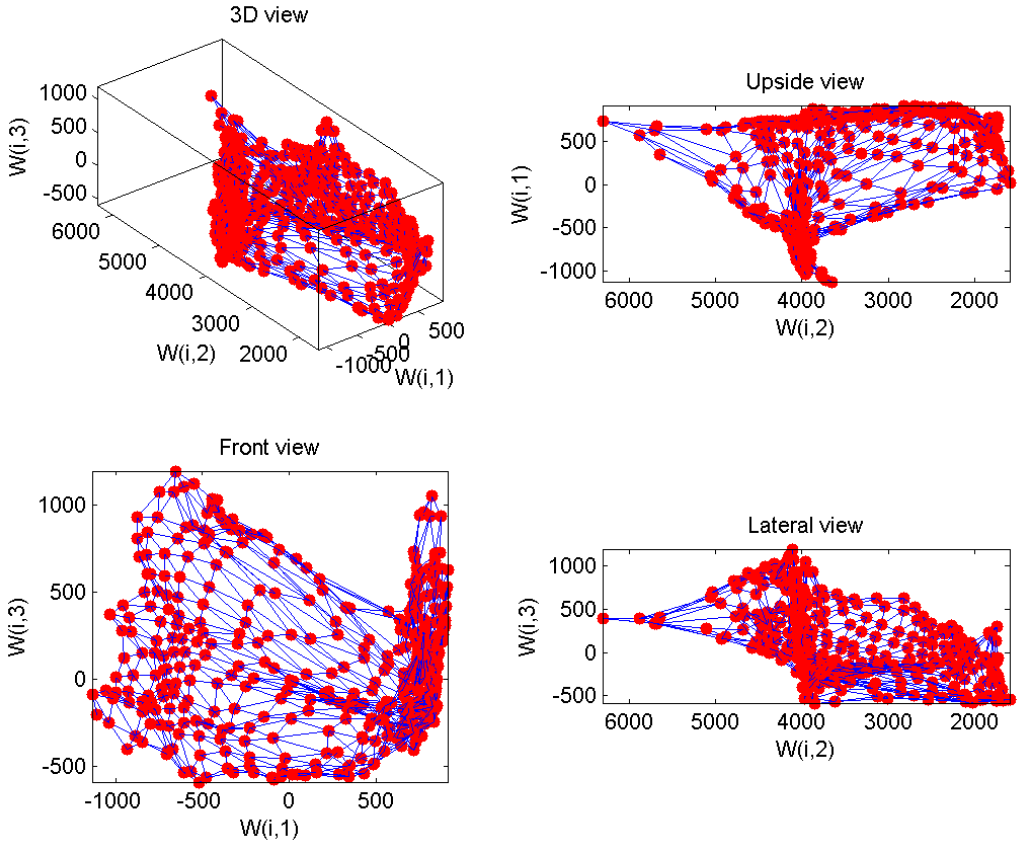


Figure 3.5: A grid of 20x20 nodes trained by SOM to quantize the reference position filtered point cloud shown in 3.4.

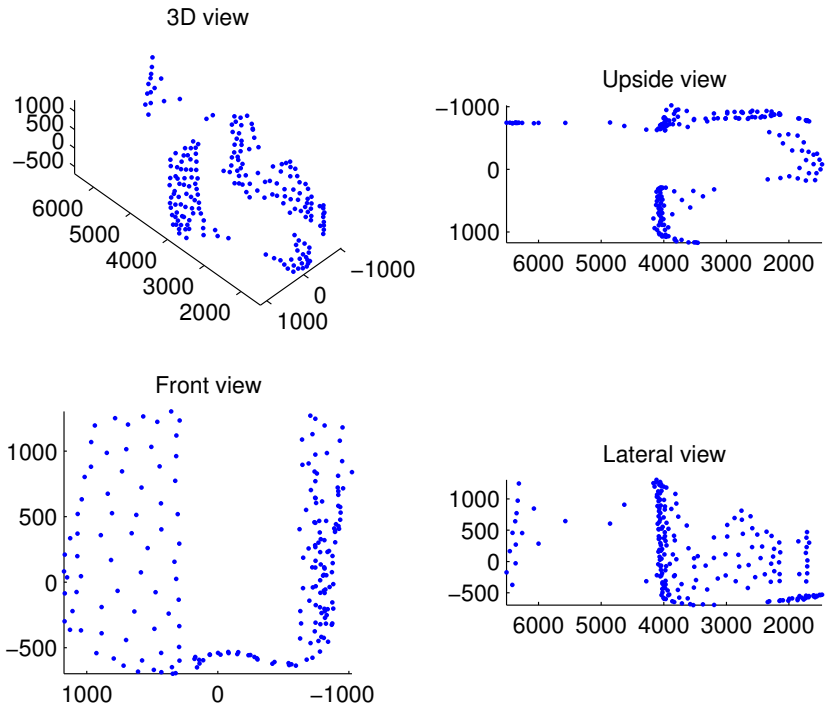


Figure 3.6: A codebook of 200 nodes trained by NG to quantize the position filtered point cloud shown in figure 3.4.

geneous structured data while at the same time learning the relationships between the input signals without specifying a network topology.

In figure 3.6 can be seen the point cloud generated by fitting a 200 node NG to the reference position measures. It can be appreciated that physical surfaces can be easily identified. In the following only results based on the NG approximation of the 3D camera point cloud will be reported.

The main drawback of this data fitting and reduction process is its computational cost. While it allows the Evolution Strategy of the following module to run real-time, the cost of fitting each point cloud somehow hinders the real-time operation of the complete system. Since the computational experiments reported below where made using a free SOM Toolbox for Matlab [112] with default parameters, we expect that a compiled version of the code will reduce this computation time to fit into real-time constraints.

3.2.3 Evolution Strategy module

The input data for the egomotion parameter estimation consists of a sequence of codebooks $S_t = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ $t = 0, 1, ...$ computed by the NG module over the sequence of ToF 3D camera captures. These codebooks give an approximation of the 3D shape of the environment in front of the robot at each time instant. The robot is described by its position and orientation at each time instant t given by a real position $P_t = (x_t, y_t, \theta_t)$, an estimated position $\hat{P}_t = (\hat{x}_t, \hat{y}_t, \hat{\theta}_t)$ and the codebook S_t. At the next time instant $t + 1$, we obtain S_{t+1} from the application of the NG to the camera data. Our objective is to compute an accurate estimation of the position $\hat{P}_{t+1}$ from the knowledge of the codebook S_{t+1} and the previous estimation of the position $\hat{P}_t$ at time t. We assume that, from two consecutive positions, the view of the environment is approximately the same, but from a slightly different point of view (i.e., the robot is viewing roughly the same things, but from other position). The way to calculate this new position estimation is to calculate the transformation T_t that gives the best prediction $\hat{S}_{t+1}$ of S_{t+1} given S_t. In other words, the estimation of the motion of the robot is stated as the following optimization problem: search for the parameters of T_t which minimize the matching distance between S_{t+1} and the prediction $\hat{S}_{t+1} = T_t \cdot S_t$:

$$\hat{T}_t = \arg\min_T \left\{ d\left(S_{t+1}, \hat{S}_{t+1}\right) \right\}.$$

This search is made by means of an $(\mu/\rho + \lambda)$-ES Evolution Strategy [9, 88]. Since we are looking for the transformation matrix T_t between the ToF camera data at times t and $t + 1$, the traits of the ES individuals will encode the parameters of T_t. Although the data consists of clouds of 3D points, the robot is moving only along the plane of the floor, so we only need the parameters necessary for the transformation within that plane. Each individual would be an hypothesis $h_i = (\triangle x_i, \triangle y_i, \triangle \theta_i)$, where $\triangle x_i$, $\triangle y_i$ and $\triangle \theta_i$ are the hypothesis about the translation and rotation parameters of the transformation matrix hypothesis $\hat{T}_i$, which specifies the motion between the relative positions P_t and P_{t+1}.

$$\hat{T}_i = \begin{bmatrix} \cos(\triangle\theta_i) & -\sin(\triangle\theta_i) & \triangle x_i \\ \sin(\triangle\theta_i) & \cos(\triangle\theta_i) & \triangle y_i \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

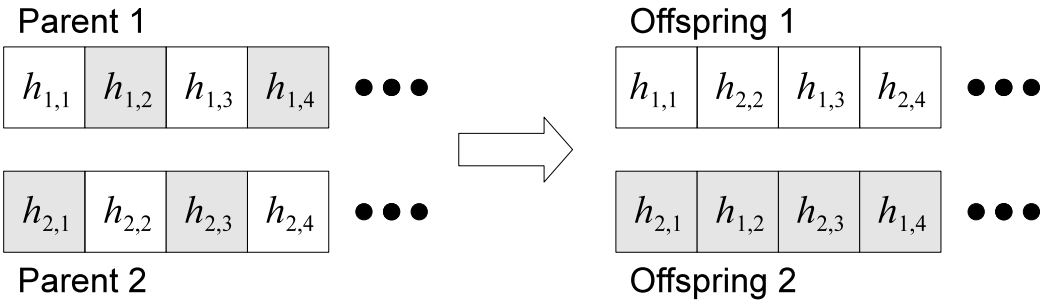


Figure 3.7: Offspring generation from a couple of parents. White traits from each parent go to the first offspring, gray ones to the second.

For each hypothesis h_i encoded by an ES individual we have a prediction

$$(\hat{S}_{t+1})_i = \hat{T}_i \times S_t, \tag{3.3}$$

which is used to calculate the fitness function $F(h_i)$ as a matching distance between codebooks.

An initial population H_0 of 30 individuals is randomly built by random mutation from one initial hypothesis assumed to be the null hypothesis $h_0 = (0, 0, 0)$ (i.e., no transformation: the robot has not moved from previous position). Each generation k a new parent population is built selecting the one third individuals from the previous population H_{k-1} ($\mu = 10$, in this case) with the best fitness function values. Random couples ($\rho = 2$) are selected from those best fitted individuals and their traits are exchanged by a crossover operator generating $\lambda = \mu \times 2$ offspring ($\lambda = 20$ in this implementation). The crossover operator works in a deterministic way, alternating the parent traits between the offsprings as shown in figure 3.7. Two offsprings (o_1, o_2) are generated from each parent couple (p_1, p_2) as follows:

$$o_1 = (\triangle x_{p_1}, \triangle y_{p_2}, \triangle \theta_{p_1})$$
$$o_2 = (\triangle x_{p_2}, \triangle y_{p_1}, \triangle \theta_{p_2})$$

The traits of the offspring are mutated by adding Gaussian perturbations. No adaptive mutation was used.

With the obtained transformations, we can estimate the egomotion of the robot. Starting from initial position P_0, and given an estimated transformations sequence $T_1, ..., T_t$, the robot's position at time step t will be calculated applying consecutively the transformations to the starting position:

$$\hat{P}_t = T_t \cdot ... \cdot T_1 \cdot P_0 \tag{3.4}$$

The full egomotion process is outlined in algorithm 3.1.

Fitness function

Matching two point clouds is done finding the matching point in the target point cloud S_{j*} of each point in the matching point cloud $\hat{S}_k$. The matching point is the one at minimum distance in the Euclidean distance sense:

$$j* = \arg\min_j \{\|S_j - S_k\|\}.$$

The matching distance is the Euclidean distance between the matched points $d_k = \|S_{j*} - S_k\|$. The goal of the minimization performed by the ES is to look for a distribution of the matching distances maximizing the number of points whose matching distance is close to zero (i.e. maximizes the number of points which have a correspondence in both point clouds). We have tested the mean of the matching distances as the fitness function $F(h_i)$:

$$F(h) = \frac{1}{N}\sum_k d_k.$$

This approach has some inconveniences shared by other classical registration algorithms like ICP [8], caused by the non-overlapping points in the point clouds, which can be considered outliers. The use of the mean as the value to minimize will only find an optimal solution if the distribution is centered around zero. When some data is matched against some model from which is a subset, as is required by the ICP algorithm, it can still provide that optimal solution. However, the presence of outliers in the distribution will introduce a bias in the mean of any possible distribution, thus preventing the algorithm from reaching an optimal solution. We have found better results (as will be shown below) using the distance distribution median as the fitness function:

$$F(h) = d_{k*} s.t. P_d(d_{k*}) = 0.5,$$

where $P_d(.)$ denotes the empirical distribution of matching distances.

Minimizing the median will increase the points with matching distance close to zero, being quite robust against the presence of outliers. This approach is similar to the genetic algorithm used by Chow et al. [15].

Algorithm 3.1 One step of the egomotion estimation algorithm.

Given the previous position estimation $\hat{P}_t$.
The robot moves to a new physical position P_{t+1}

1. Take measurements from the camera.
2. Filter the cloud of 3D points.
3. Obtain S_{t+1} fitting the Neural Gas network to the cloud of filtered 3D points.
4. Generate an initial population H_0.
5. Iterate until stopping condition, k is the generation index:
   (a) Select a parent population H_k^μ from population H_{k-1}.
   (b) Stop if convergence conditions in equations (3.5) and (3.6) are matched. Continue otherwise.
   (c) Generate the λ offsprings H_k^λ by recombination ($\rho = 2$) and mutation.
   (d) For each $h_i \in H_k^\lambda$.
      i. Build $\hat{T}_i$ from h_i and compute the prediction $(\hat{S}_{t+1})_i$ from S_t.
      ii. Calculate fitness $F(h_i)$ as the matching distance between $(\hat{S}_{t+1})_i$ and S_{t+1}.
   (e) Build population $H_k = H_k^\mu \bigcup H_k^\lambda$.
6. Build the estimation of the transformation matrix $\hat{T}_t$ from the best h_i in the last population.
7. Compute position estimation $\hat{P}_{t+1}$ at time $t+1$ using equation (3.4).

To reduce the computational cost of searching for the matching point we use the nearest neighbor search realized with kd-trees [6], which are a special sub-case of BSP-trees that partition a k-dimensional space. Each non-leaf node generates an hyperplane orthogonal to one of the coordinate axis partitioning the space into in two subspaces. The partitioning follows some convention in the order in which the coordinate axis are considered while descending in the tree. Typically, the points used as non-leaf nodes are the median points in the partitioned axis. This technique provides nearest neighbor search (minimum distance search for points) with a cost in the order of $O(log(n))$. The use of this technique provides a couple of orders of magnitude improvement in speed in our application.

Stopping condition

Usually an ES converges fast to the space close to the optimal solution. However, when close to the global optimum, full convergence of the population can take a great number of generations. As execution time is a crucial issue, a compromise between result optimality and the time employed to obtain it is necessary. So, instead of waiting for full convergence of the population, the ES will stop when the population has stabilized around some value. This is achieved by establishing a distance threshold δ, as a maximum Euclidean distance between the individuals of the last parent population. In experimental implementations this threshold was set in the unit. This threshold does not take into account the third trait of the individuals, the angle increment $\triangle\theta$. However, this third trait fully converges very quickly, so waiting for it does not involve any additional delay. Stopping condition will be achieved, then, when the two conditions are matched:

$$\max \|h_i - h_j\| < \delta, \quad i, j = 1, ..., \mu \tag{3.5}$$

$$\triangle\theta_1 = \triangle\theta_2 = ... = \triangle\theta_\mu \tag{3.6}$$

3.3 Experimental validation

Experiments on this approach have been done off-line in order to guarantee that the results could be reproducible. Several data sets containing sequences of point clouds given by 3D camera measurements where recorded in series

of walks across the corridors and rooms of our building, as described in section C.4.2 of appendix C. These datasets are described in appendix E. The objective of the experiment was to check how well the egomotion algorithm could reconstruct the paths followed by the robot in those walks using the recorded data. We report here the results obtained over one of those recorded walks. This walk consists in a wall-following tour around one big sized (88 m^2) empty room. This sample was selected since it presented some characteristics that we expected could be troublesome for the effectiveness of the algorithm, which will be discussed below. The nominal trajectory of the robot was estimated approximately by measuring the path followed by the robot in the room's floor. This nominal path will be used to compute the path estimation error of the different algorithms tested.

In figure 3.8 a comparison of the robot trajectories estimated by the egomotion algorithm using median and mean as fitness functions is shown, along with the path reconstruction performed by the odometry and the nominal path. The trajectories estimated by our system do not follow any kind of correction algorithm. It can be appreciated that the deviation from the path in the case of the median fitness function is lower than for the mean fitness function, and that, in spite of the accumulated error, in both cases the estimated path keeps the shape of the real one, obtaining an estimation comparable to the one provided by the odometry. In general, odometry obtains a better estimation on the pure lineal displacement sections of the path, as it has no disturbing effects. On the other hand, where there are turning angles the true path is better estimated by our system. Notice the loop artifacts shown in the right-turning corners of the ES estimated trajectories. They are caused by the installation of the camera on top of the robot, which was mounted in a position to the right of the physical rotating axis of the robot. Because of that, when the robot performs a closed turn to the right, the position of the camera actually moves back and left from its former position. Since our egomotion system estimates the position of the camera, those small loops shown in the corners correspond to the true motion of the camera. Also it must be noticed that when there are left-turn corners in the path this artifact does not appear.

The use of kd-trees for searching matching points allows for bigger codebooks to be trained. In figure 3.9, egomotion estimations using 100 and 400 codebooks are shown. The improvements obtained by using a 400 codebook does not justify the increase in computation time shown in table 3.2. Therefore, further experiments were performed with a codebooks of size 100.

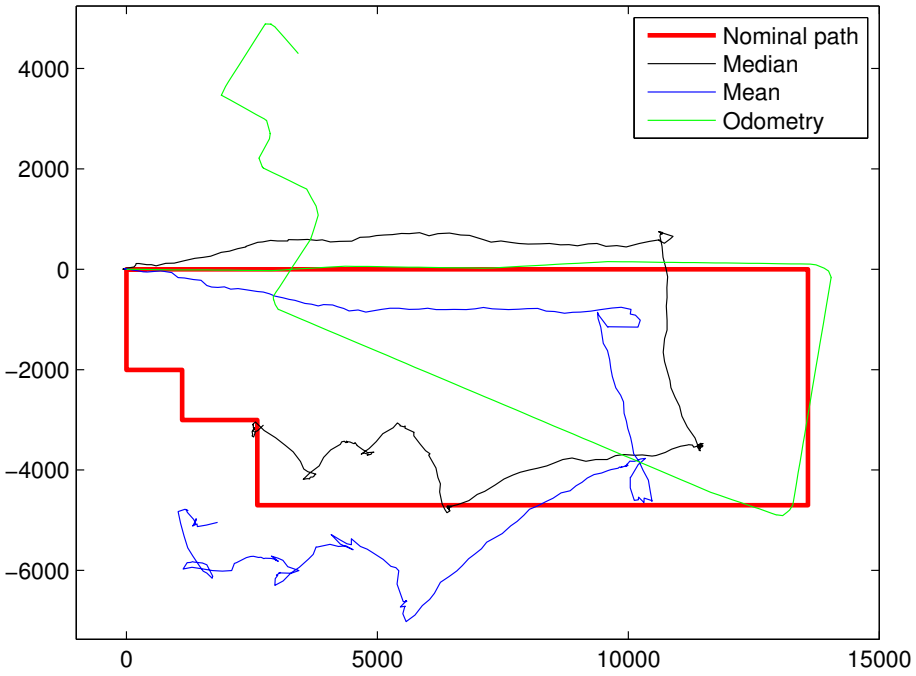


Figure 3.8: Comparison between estimated egomotion with mean and median fitness function of the ES module. Odometry and approximate real paths are shown as reference.

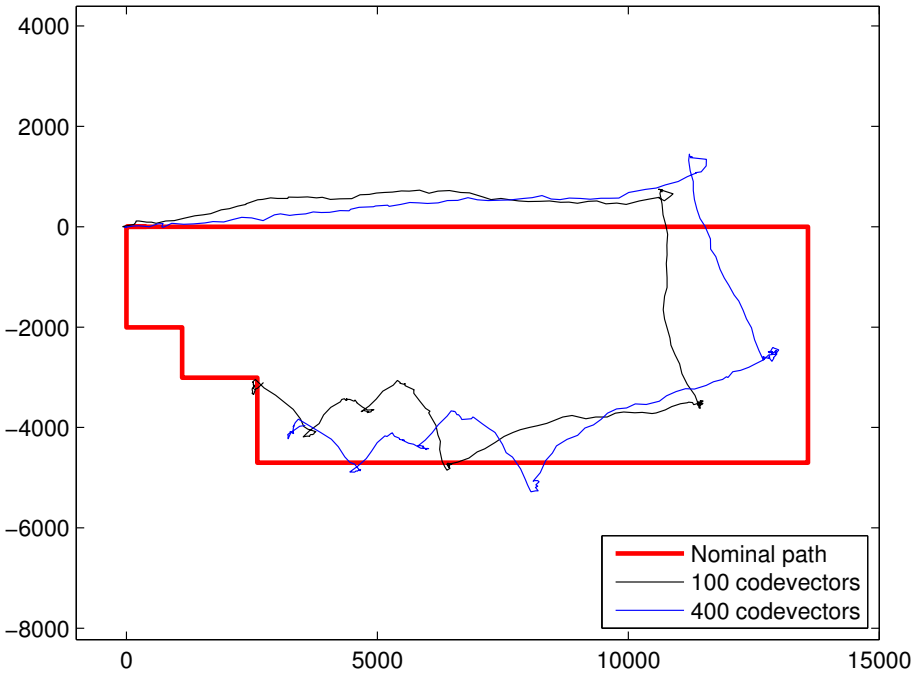


Figure 3.9: Comparison between estimated egomotion with 100 and 400 codebooks.

Some registration algorithms found in the literature where tested on our data for comparison with our ES approach, using Matlab code provided by [98]. Results were quite surprising to us. In general, the classical algorithms tested gave quite bad results. The best qualitative result was given by a variation of ICP developed by Zinsser [126], shown in figure 3.10 against our ES and the other algorithms tested. We were expecting much better results, since all of the algorithms are recognized in the literature, with proven registration efficiency. Our interpretation of the results is that, as those algorithms perform full 3D registration, they are very sensitive to small variations of the camera position. Small errors in the mounting of the camera (e.g., being it lightly tilted or rotated in respect of the longitudinal or transverse axis of the robot) or produced by the movement of the robot (e.g., small tilts of the camera if the frame is captured while accelerating or braking), could induce rotations in X or Y axis which could be disastrous for the egomotion estimation, when only Z axis rotations are expected. It seems that our approach, while maybe unable to provide an optimal registration in a more general situation, is more robust and able to cope with those issues more satisfactorily.

A comparative table of the error for the registration algorithms tested, odometry included, is shown in table 3.1. The error is calculated against the nominal trajectory of the robot. Mean error along the path, accumulated error and final position error are shown in millimeters. It can be appreciated that our system obtains results comparable to the odometry, with a better final position estimation. The other tested registration algorithms diverge greatly from the real path, as is seen in the big mean error. The low final error of the Besl and Chow algorithms is easily explained by the closed loop shape of the path: as they are unable to reconstruct the path, they do not move far from the starting point, remaining always close to the final point.

Figure 3.11 shows the evolution of the error along the path for all the egomotion estimation algorithms tested. It can be observed that the error in both odometry and our ES grows smooth and slowly. On the other hand, the other approaches diverge very fast initially but later decrease at the end of the path, as the path gets closer back to the origin.

Our neuro-evolutionary system still suffers from a drawback coming from the matching of the codebooks. If the overlapping areas of the consecutive frames cover less than 50% of the points in the set, the algorithm will be unable to achieve an optimal solution. This is a problem common to any registration algorithm, but in this setting the problem can worsen if the

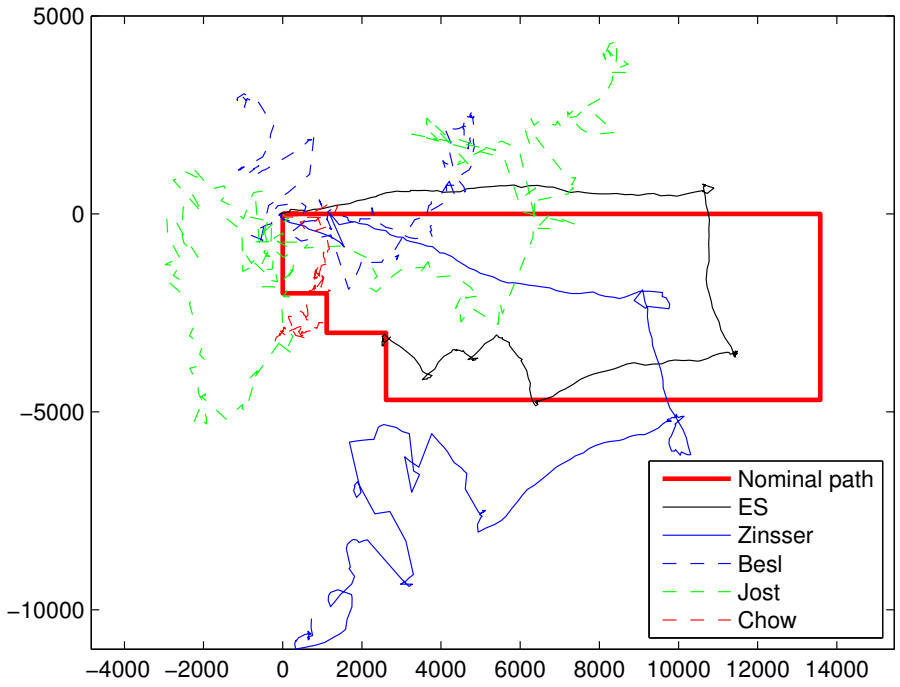


Figure 3.10: Comparison between estimated egomotion with ES and Zinsser.

Algorithm	**Mean error**	**Acc. error**	**Final error**
Odometry	2585	695602	5255
ES	2952	794266	3881
Zinsser	12711	3419386	10291
Besl	9300	2501695	3017
Chow	6893	1854391	2999
Jost	8738	2350702	8478

Table 3.1: Mean, accumulated and final position error in mm. for the estimated path by different registration algorithms, plus the odometry.

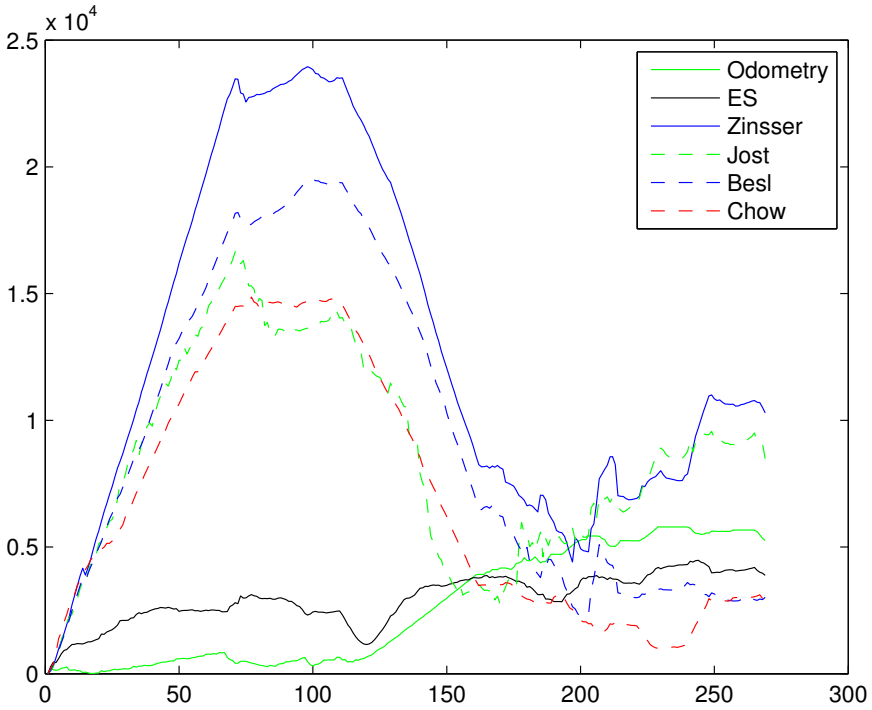


Figure 3.11: Error evolution along the path estimated by different registration algorithms.

Algorithm	**100 Codevectors**	**400 Codevectors**
Besl	84	394
Chow	5224	14936
ES	9564	N/A
ES kd-trees	277	964
Jost	63	257
Zinsser	50	389

Table 3.2: Execution times (in seconds) of different registration algorithms for the sample path of 269 frames.

relation between motion speed and capture rate is not controlled properly (i.e., if the robot moves 'too much' from one frame to the next), considering that the field of view of the used camera is relatively narrow. Depending on the kind of robot operation or control model this can become a serious issue.

Another annoying detected issue is pervasive of the typical scenarios that a mobile robot has to face in indoor environments. When following a wall, distance and angle from the wall are properly calculated, but correct estimation of the forward linear motion is difficult to obtain. As happens with the floor, a typical wall is a featureless uniform structure in which, when moving along, new viewed parts are indistinguishable from parts left behind, thus making extremely difficult to estimate how much wall have been traveled. In absence of other objects in the field of view this estimation has to come from small features in the walls or from far away front-faced walls, which do not provide optimal matching features. This problem is equivalent to the *aperture problem* present in many other computer vision techniques, when only a featureless patch of the surface of an object is visible.

In the execution times for the sample path shown in table 3.2 for several registration algorithms, it can be seen that our first approach is the slowest one. The use of kd-trees for closest point search improves performance more than 30 times using 100 codevector codebooks, getting closer to the other algorithms and outperforming significantly the other evolutionary approach present, the GA from Chow. Even though other registration techniques are faster, they do not give a good egomotion estimation. As the ES with kd-trees, while slow, is fast enough to real-time operation, it seems to be the overall better suited approach to this problem.

3.4 Summary and conclusions

In this chapter, we have reported our advances towards a new mobile robot egomotion estimation neuro-evolutionary algorithm, based on information obtained from a 3D camera and a system composed of a NG and an ES. The 3D camera provides information about the distances to the objects in the robot environment. This information is translated into a point cloud, which is filtered to remove wrong measurements and approximated with a NG, in order to obtain a codebook that maintains the shape of the objects in the near environment of the robot. Those codebooks are used to estimate the spatial transformation leading to the next position by an ES, evolving a population of hypothesised positions using a matching distance as fitness function. Some experimental results have also been presented and its shortcomings discussed.

Results obtained show that the egomotion estimation system by itself provides a path reconstruction comparable to or even improving the one provided by odometry. We have also shown that the use of the median of the minimum distances between points of the codebooks improves the results of the registration process. Also, the use of kd-trees to search for the closest point reduces computation times manifold. The use of bigger sized codebooks does not seem to improve the results, while increasing notably computation time. Qualitative and quantitative comparisons with other registration algorithms have been also reported. While those algorithms have been shown to be faster that the NG-ES approach, our system has resulted in the best egomotion estimation.

Several drawbacks of our egomotion neuro-evolutionary system have been identified, namely the problem with the slightly overlapping consecutive frames and the aperture problem with long uniform featureless structures. Both problems derive from the very nature of the data used. Our most immediate future work will be to try to overcome those issues by the integration of the neuro-evolutionary system into a Kalman or particle filter architecture. Also, fusion of the 3D data with the optical information provided by the robot's video camera could be used to face the problems inherent to the registration approach.

Chapter 4

Multi-robot Visual Control

This chapter contains the description of an experimental proof-of-concept of a new paradigm in the general field of Multi-component Robotic Systems (MCRS) [26]. Our efforts are focused on the visual control of a specific kind of MCRS, called Linked MCRS in [26], characterized by the existence of a passive link between active agents. Specifically, we are trying to develop the tools and techniques necessary to build a MCRS able to displace a hose along an unstructured environment. In section 4.1 we give a short overview of multi-robot systems in order to motivate the experimental work described below, and a definition of the hose manipulation problem that will be addressed in this chapter. Section 4.2 will detail the guidelines of the proof of concept which implementation will be described in the following section 4.3. Finally some conclusions will be discussed in section 4.4.

4.1 Multi-robot systems

Multi-Component robotic systems, or multi-robot systems, have been proposed in several application domains as a way to fulfill more efficiently a task by cooperation between several robots. Also, there are complex tasks that can not be accomplished by a single robot and must be performed necessarily by a multi agent system [25]. We are not interested, however, in systems which are a mere collection of robots performing a fixed task in a static environment, like the industrial robots of a production line. The kind of multi-robot systems we are interested in has to be composed by a variety of interacting, cooperating autonomous agents with physical embodiment

that impose restrictions on what they can do, but also give them power to do some specific things, and specific ways to contribute to the collective behavior for the realization of tasks [26]. For instance, the swarm-bots [24] are able to grasp one another to build specific morphological configurations of the whole system that allow them to overcome obstacles like trenches or bumps which can not be overcome by a single robot. Multi-robot systems have been proposed for the operation in big, unstructured and very dynamic environments. In this type of environments, a multi-robot system possesses several advantages from a single robot. For instance, while a single robot can only perceive a very limited portion of the environment, a group or robots communicating between them cover a much wider space, allowing them to respond better to changes in that dynamic environment, as those changes are perceived earlier and transmitted to the other members of the group.

In the last two decades, a lot of effort has been put in transferring the multi-agent paradigm to mobile robotics. However, it is still difficult to properly categorize this research field, as the same nature of the paradigm, with almost limitless possibilities in configurations and objective tasks, does not impose any clear distinction between different approaches. There are several reviews giving different categorizations [25, 10, 54, 30, 26], focusing in different aspects of the multi-robot systems. For instance, in [25] a taxonomy of multi-robot systems was created focusing on the size of the robot group, their communications range, topology and bandwidth, reconfigurability of the system, processing abilities of the individuals and the homogeneity of the group. In [10] several axes were defined for this classification. Those axes represented the group architecture, the conflict resolving mechanism, origins of cooperation, learning and adaptability of the system and the geometric problems the system is tied to due its physical capabilities. Both [54] and [30] proposed a taxonomy based in four layers: cooperation, knowledge, coordination and organization. Finally, in [26] a categorization of MCRS is done focusing in the coupling of the individuals, their morphology, the tasks they have to perform and the environment in which have to be carried out, the control of the system and the perception used to obtain feedback from the actions taken an their effects. Focusing in the morphological properties of the robot, the way they are physically connected, three main types of MCRS were identified: Distributed, Linked and Modular:

- The Distributed MCRS correspond to groups of robots without physical connection, performing tasks such following the leader, herding objects,

etc.

- The Modular MCRS correspond to groups of modular robotic elements attached with rigid, strong links to assume a given morphology which can be task-dependent.

- The Linked MCRS corresponds to groups of autonomous robots linked through a passive non-rigid linking element. This passive element introduces dynamic problems and restrictions that may greatly influence the control of the whole robot ensemble.

The Distributed and Modular MCRS are familiar concepts, however the Linked MCRS is a new category, not previously identified in the literature. For instance, if we consider the task of self-perception, the ability to measure the configuration of the system, it differs greatly in the case of the Linked MCRS from the other kinds. The Modular MCRS may sense its configuration through the state of its rigid links between modules. The Distributed MCRS may estimate it from the information gathered by the individual robots by themselves. The Linked MCRS, however, need to make (visual) observations on the configuration of the passive linking element. If we consider the problem of defining (computing) control rules, the Modular MCRS may rely in the rigidity of its joints between modules in order to solve the inverse kinematics problem. The Distributed MCRS need only to command the individual robots, without taking into account interaction other than avoiding avoiding collision and traffic jams. The Linked MCRS needs to model the behavior of the passive linking element, which may depend on some physical parameters such as elasticity or weight, and to take it into account to compute the individual robots control commands. The control on the linking element is indirect and its response can be highly non-linear. These kinds of problems are new in the robotics literature, and we are starting to deal with them from several points of view. In a companion PhD Thesis, Zelmar Echegoyen deals with the problem of modelling and derive the formal inverse kinematics and dynamics of this kind of robots. In this PhD dissertation we dwell more on the physical realization of a proof-of-concept of a Linked MCRS with a concrete set of robots and a piece of electric cable as the passive linking element.

4.1.1 Control of a Linked MCRS for hose transportation

The transportation, deployment and manipulation of a long[1] almost unidimensional object is a nice example of a task that can not be performed by a single robot. It needs the cooperative works of a team of robots.

In some wildly unstructured environments like shipyards or large civil engineering constructions a typical required task is the transportation of fluid materials through cables, pipes or hoses. The manipulation of these objects is a paradigmatic example of a Linked MCRS, where the carrier robot team will have to adapt to changes in the dynamic environment, avoiding mobile obstacles and adapting its shape to the changing path until it reaches its destination.

The general structure and composition of this hose transportation MCRS system would be that of a group of robots attached to the hose at fixed or varying points. The robots would search for space positions in order to force the hose to adopt a certain shape that adapts to the environment, while trying to lead the head of the hose to a goal destination where the corresponding fluid will be used for some operation. In figure 4.1 we give a rough illustration of this problem. The pipe at the lower left corner represents the fluid source, the hose is represented by the black thick line, and the small robots attached to it try to move it so that its head (attached to a robot) reaches the goal represented by the red circle. The other objects in the scene represent changing environment conditions that may force changes in the hose spatial disposition. This general form can take multiple implementations depending on several elements of its design:

- Robot-hose attachment: Robots could be fixed to a point of the hose, they can move along it, or they can pull it through special gripping mechanisms.

- There can be a centralised control which determines the positions of each of the robots or it can be decentralised, taking each robot its own control decision.

- Robots can be homogeneous or heterogeneous, having different configurations and tasks (e.g., "pulling" robots, which tow the hose, and

[1]The adjective "Long" used here is relative to the size of the individual robots. The object's length must be some order of magnitude greater than the robot's size.

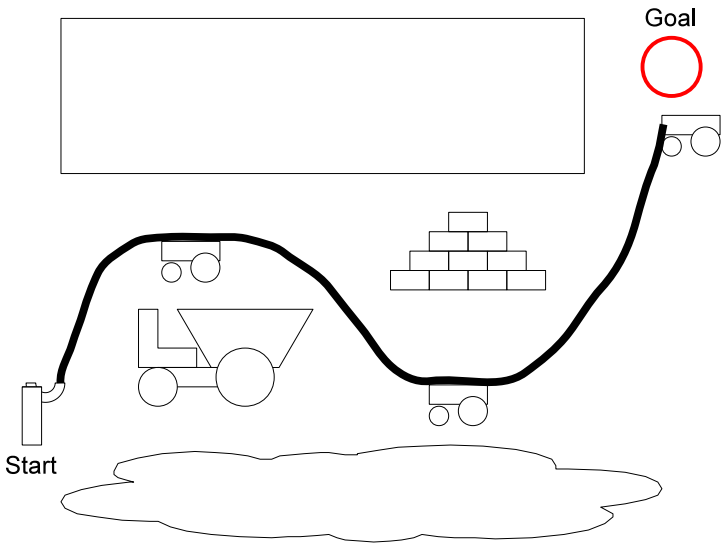


Figure 4.1: Hose transportation and deployment by three robots between starting and goal positions avoiding obstacles.

"cornering" robots, which take fixed positions and give shape to the hose).

- Perception can be global, with some agent acquiring a global view of the system, or local, with every robot acquiring information of its close surrounding, which can share with the remaining robots or not.

Our research group is actively involved in studying the hose manipulation problem from diverse points of view producing modelling and simulation of the hose-robots dynamics, and some problem characterizations [28, 65, 27]. As a starting point to achieve this system on a real robotic platform, we have made a physical realization using real robots of a vision controlled MCRS which faces the basic issues in this hose transportation problem.

4.2 Proof of concept experiment

In order to perform a proof-of-concept physical realization we have defined as the basic simplest problem that of controlling a MCRS linked system whose objective is to perform is the transportation of the hose in a straight line in an environment without obstacles from an arbitrary configuration of hose and

robots. In the case of having freely attached robots, being each one powerful enough to pull the hose, this task would be trivial. Only one robot would be needed to pull the hose head towards the objective position, being the only issue to cope with the drag caused by the hose. However, in the case that the robots are fixed to the hose or that an individual robot is not powerful enough to pull it or the initial configuration of the hose is arbitrary, this task has to be necessarily performed by several robots. Then, the control gets more complex, as there are several robots that need to guarantee a certain hose configuration and each individual robot motion has to be controlled in order to keep a desired formation. Thus, although the task is the simplest one that can be defined, it poses several problems that are the basic stones of the solution of more sophisticated tasks. Besides, we will need to deal with the embodiment problem: in a physical implementation the real robots and other systems that are being used impose restrictions and conditions that must be coped with in order to obtain a working system realizing the proposed task. In the following we define with precision the task imposed to the MCRS systems.

4.2.1 Task statement

A hose of 2 meters long has to be transported in straight line by three robots. The robots are attached to the hose with a rotating coupling which allows them to freely rotate below the hose. Each robot will take a fixed position in the hose. The leader and last robot will be in both ends of the hose, while the second robot will be fixed in the middle of it.

Starting conditions Robots and hose could be in any configuration which keeps the following restrictions:

- Leader robot has to be more advanced than the second robot, and this has to be more advanced than the last robot.
- The hose can be folded, but not so much as to block its nearest robot when it starts moving.
- Robots should be able to start moving without obstacles (e.g. with enough separation from the wall).

Experiment stopping condition The leader reaches a stated goal or goes out of the camera's field of view.

Hose transportation conditions

- Robots must follow the direction of the leader.
- Robots must be separated enough to avoid that the hose forms loops.
- Robots must be close enough to avoid the hose tightening and causing dragging between robots.

Perception Centralised perception, provided by an USB camera in a high position (approximately 2.5 meters high). It should be pointed to the floor in an angle that covers at least 2 meters of it, in order to be able of view the tree robots at the same time when the hose is fully extended.

Control

- The leader robot's orientation is controlled manually, its speed is controlled autonomously.
- Second and third robot are controlled autonomously.
- The control commands are computed by a centralising agent which also processes the perception.

Communication The communication of the central control agent and the hose carrying robots is a two way wireless connection, over a shared channel. Robots are identified by the central control in a round-robin schema.

4.3 Embodiment, tools and solutions

The task defined above has rather diverse solutions depending on the embodiment of the task, that is, the actual robots employed and the actual physical features of the passive element. In this section we give an account of the hardware used, the image analysis procedures applied to obtain the visual feedback and the control heuristic applied to define the control strategies.

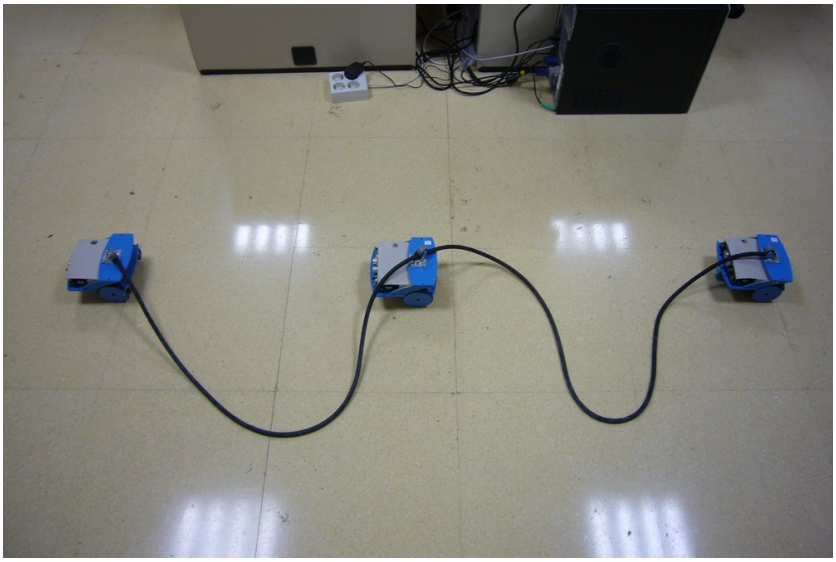


Figure 4.2: Snapshot of the Robot-hose transportation system.

4.3.1 Hardware

The experimental solution to this problem was implemented using three small SR1 robots (see section C.1.2 in appendix C). Each robot was attached to an electrical cable of 1 cm. of diameter, which takes the place of the hose (figure 4.2), by means of a bearing which allow the robot to rotate freely. One camera was placed in a 2.5 meters high mobile stand, facing down in an angle of around 60^{o} and capturing about 2-3 meters of the floor in front of it. The camera was attached to one laptop PC which performed the centralised perception and control. Control commands were sent to the robots using RF wireless communications, as detailed in section C.4.3 of appendix C. The robot's compass information is used in the follow the leader strategy described below. This is a minimal sensorisation for the task at hand, however this sensor is quite sensitive to the magnetic fields that abound in laboratory environments.

4.3.2 Perception

The centralised perception is provided by a single camera that captures the scene encompassing the three robots and the hose. The images acquired are segmented in search for the three robots and the hose. This segmentation

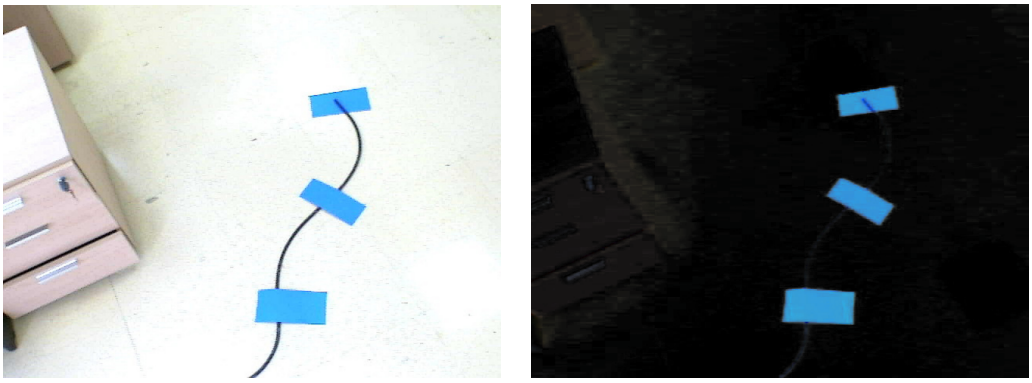
process assumes several conditions on the environment's configuration:

1. Uniform floor of bright color, close to the white color.
2. Blue colored robots.
3. Non-blue, dark colored hose.
4. White, uniform illumination, which may produce strong reflections as can be appreciate in figure 4.2.

Image segmentation will be composed of two separated processes for the detection of the robots and of the hose, each fit for each kind of object.

Robot's segmentation For the segmentation of the robots we are mainly interested in avoiding the effect of strong reflections on the floor and enhance the image color contrast in order to bring out the blue robots from the bright floor. This is achieved by means of a preprocessing step in which a Specular Free (SF) image [107] is created. Assuming the Dichromatic Reflection Model (DRM) [100], images are the sum of two components: the diffuse component (which models the chromacity of the observed surfaces) and the specular component (which models the chromacity of the light source which illuminates the scene). This model implies that the pixels with the reflections we want to avoid have a specular component. A Specular Free image is, then, an image geometrically identical to the original one but with its specular component removed. Several algorithms have been proposed in the literature for computing SF images [34, 108, 125]. One important step in those algorithms is the estimation of the chromacity of the source of illumination. As we already know this chromacity, because we assume a white light source, we have used a custom algorithm to obtain a SF image adapted to our needs.

Using a white light source, in this algorithm we profit from the characteristics of the RGB cube, as pixels corresponding to reflections (and bright surfaces close to white color) will be very close to the axis that goes from point (0,0,0) to point (1,1,1) of the RGB space while pixels corresponding to diffuse components in the image will move away from it and will be closer to the pure color axes. This property is used to reduce the intensity of the specular pixels and improve the intensity of the diffuse ones proportionally to their distance with the axis (0,0,0)(1,1,1). Given an input RGB image



(a) (b)

Figure 4.3: Specular Free image computation. Original (4.3a) and resulting SF (4.3b) images.

$X = \{x(i,j)\}$, where $x(i,j) = \{r_{ij}, g_{ij}, b_{ij}\}$, an intensity image $I = \{d(i,j)\}$ is computed as

$$d(i,j) = \max\{r_{ij}, g_{ij}, b_{ij}\} - \min\{r_{ij}, g_{ij}, b_{ij}\}, \tag{4.1}$$

in this equation, white/gray pixels will have value $d(i,j)$ close to zero, while colored regions, corresponding to diffuse components, will be greater than zero.

The RGB image is then transformed to HSV space and its intensity channel is replaced with the computed intensity image I, so that white/gray pixels become black ones. This HSV image is then transformed back to RGB space. The result of this process can be appreciated in figure 4.3. Since we are looking for blue robots, they can be easily found in the SF image looking for the regions with highest intensities in the B channel. The result of this step is a collection of boxes $\mathbf{R} = \{R_1, ..., R_n\}$ giving the location of the robots. When processing a sequence of images, the robot detection process is done in the neighborhood of the previous image detected boxes.

Hose's segmentation The segmentation of the hose profits from the strong contrast of a dark object over a bright floor. Given the original RGB frame and the regions obtained from the robot's segmentation, hose's segmentation is performed by the following processing steps:

1. The image is binarized, white pixels code the hose detection.
2. The binary image is skeletonized.
3. Regions of the skeletonized binary image are identified and labeles.
4. Discard regions with very few pixels.
5. Discard regions that do not connect two of the boxes found before containing a robot

Each region obtained after this process is considered a segment of the hose, we denote them $\mathbf{S} = \{S_1, ..., S_{n-1}\}$ where segment S_i connects robot boxes R_i and R_{i+1}.

In summary, the outputs of the visual perception system are the rectangular regions $\mathbf{R} = \{R_1, ..., R_n\}$ of the image where the locations of the n robots are detected (robot position p_i will be the centroid of that region) and several connected components $\mathbf{S} = \{S_1, ..., S_{n-1}\}$ corresponding to detected hose segments. Those hose segments are checked to make sure that they connect two robots. Segments that do not connect two robots are discarded. At each new image, the previous detections are used as the basis for the search of the new detections. The full perception process is outlined in algorithm 4.1.

4.3.3 Control heuristic

Although the control is centralised because the actions are determined by a central agent installed on a specific separate computer and then communicated to the robots, each of the actions of the robots are computed independently, without taking into account the state of the other robots, that is, each robot is an autonomous agent acting as slave to the central controller. Each robot control will be determined only by the segment of the hose that is immediately ahead of him, and the information about the orientation of the leader. In this way the system is very scalable and can be extended to any number of robots.

The trajectory of the robots will be controlled by a "follow the leader" strategy. The leader will be remotely controlled and the remaining robots will follow its orientation: at each time step, the tema robots will check if

Algorithm 4.1 Perception step of the robot-hose transportation system.

For each step k, given an input RGB image X_k and the regions $\mathbf{R}_{k-1} = \left\{R_1^{k-1}, ..., R_n^{k-1}\right\}$ containing the n robots in the previous step:

1. Define a region of interest (ROI_k) in X_k centered in $\mathbf{R}_{k-1}$ where the new positions of the k-th robot will be searched.

2. Detection of the robot locations:

   (a) Compute the intensity image I_k for each ROI_k (equation 4.1).

   (b) Transform each ROI_k to HSV, replace intensity channel with I_k, transform back to RGB, obtaining an image SF_k.

   (c) Obtain new regions $\mathbf{R}_k = \left\{R_1^k, ..., R_n^k\right\}$ as the regions with highest intensity in the B channel of SF_k and close to $\mathbf{R}_{k-1}$.

3. Detection of hose segments:

   (a) Binarize the original RGB image looking for black pixels.

   (b) Skeletonize and discard small and isolated regions.

   (c) Label connected components,

   (d) Discard connected components not connecting two regions in $\mathbf{R}_k$.

   (e) Return resulting $\mathbf{S}_k = \left\{S_1^k, ..., S_{n-1}^k\right\}$ regions as detected hose segments.

they have the same orientation as the leader, using their compasses. They will reorient themselves trying to align with the leader in case they are not.

Each robot's speed will be given by a control heuristic that takes into account the state of the hose segment ahead of it. This state is a function of the curvature of the hose segment. This heuristic undelying reasonoing is that if two robots are too close, the hose segment between them will shrink and increase its curvature, with the risk of forming loops. As only the robot at the hose segment's rear will adapt its behavior to the segment state, to increase the distance between robots the only possible policy is to reduce the speed of this robot. On the other hand, if the two robots attached the hose segment are separated enough the curvature of the segment will be very close to the straight line. This can state also a problem, since a too tightly stretched segment will produce dragging between the robots, which should be avoided. In this case, the rear robot should accelerate to ease the tension of the hose.

Having a perfectly zenithal camera watching the scene the state of the hose could be easily estimated by measuring the distance between the robots. However, as we have a tilted camera in order that our field of view covers enough space for the observation of a significant behavior, the perspective projection may cause some problems when measuring distances between robots placed at different distances from the camera's stand. Given a hose segment $S = \{s_1, ..., s_m\}$, where s_j is a pixel site coordinates belonging to the connected component S, we define the curvature c of the segment S as the proportion between the maximum distance d_h from the hose segment points s_j to the line L_{p_1,p_2} that crosses both robot's positions (p_1, p_2) and the distance between the robots d_r:

$$c = \frac{d_h}{d_r}, \tag{4.2}$$

where

$$d_h = \max_i \|s_i - L_{p_1,p_2}\|,$$

$$d_r = \|p_1 - p_2\|.$$

This is equivalent to obtain the ratio of the sides of the rectangle that encloses the hose segment and has the length of the distance between the

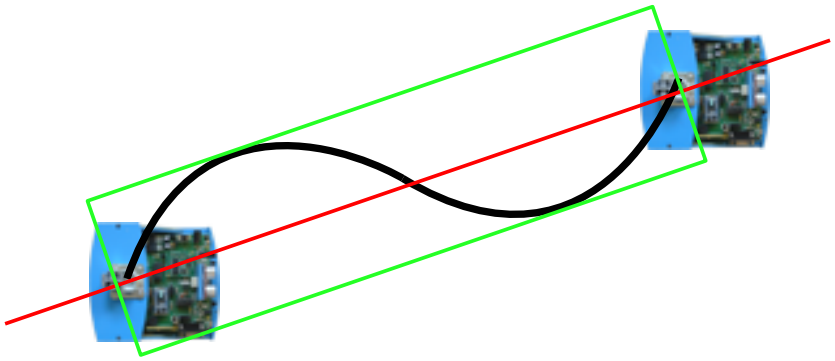


Figure 4.4: Hose state calculation. The proportion between the width and length of the green box will be the measure of the state of the hose segment.

robots (figure 4.4) . Being it a ratio value, it is not so influenced by the perspective. Three rules determining the consequent robot speed where defined over the values of this proportion c:

- $c \leq 0.15$: The hose segment has stretched too much. The rear Robot takes *fast* speed trying to shrink the hose to avoid dragging the front robot.
- $c \geq 0.30$: The hose segment has shrink too much. The rear Robot *stops* while the front robot continues its motion and waits for the hose to stretch to avoid the formation of loops.
- $c > 0.15$ & $c < 0.30$: The hose has the correct length. The rear Robot takes *cruise* speed and continues advancing.

The Control heuristic executed at the central agent is outlined in algorithm 4.2.

4.3.4 Experiment realization

In figure 4.2 an example of the realization of the hose transportation task defined above is shown. In each frame, the robots are marked with their state (advancing, stretching, shrinking) and the curvature of their respective hose segment. Detected hose segments are marked in red. The six frames are extracted from the video generated in the test and shows how the system reacts to the different states that the hose takes:

Algorithm 4.2 One step of the robot control heuristic.

Each time step k:

1. Capture a frame from the global camera.
2. Extract robot positions $\mathbf{R}_k$ and hose segments $\mathbf{S}_k = \left\{S_1^k, \ldots, S_{n-1}^k\right\}$ from the frame as described in algorithm 4.1.
3. For each i-th robot, $i = 2, ..., n$
   (a) Get the curvature c_{i-1} of the hose segment ahead S_{i-1}^k (equation 4.2).
      i. If $c_{i-1} \leq 0.15$ then shrink the hose.
         A. Set i-th robot's speed at *fast*.
      ii. If $c_{i-1} \geq 0.30$ then stretch the hose.
         A. Set i-th robot's speed at *stop*.
      iii. If $c_{i-1} > 0.15$ and $c_{i-1} < 0.30$ then keep advancing
         A. Set i-th robot's speed at *cruise*.
   (b) Read the the leader's compass to know its orientation.
   (c) Set the i-th robot's orientation equal to the leader's orientation.

- Figure 4.5a: Starting position. The leader starts towing the hose, while the 2nd and 3th robots wait for it to stretch enough.
- Figure 4.5b: The first segment's curvature falls below 0.30 ($c_1 = 0.27$). The 2nd robot starts advancing at cruise speed. The 3th robot keeps waiting ($c_2 = 0.67$).
- Figure 4.5c: The first segment is too stretched ($c_1 = 0.11$). The 2nd robot accelerates to fast speed to shrink it. The 3th robot keeps waiting ($c_2 = 0.6$).
- Figure 4.5d: The first segment's curvature is within limits ($c_1 = 0.24$), the 2nd robot brakes itself to attain cruise speed. Second segment's curvature also enters within limits ($c_2 = 0.28$), therefore the 3th robot starts advancing at cruise speed.
- Figure 4.5e: Second segment raises again above 0.30 ($c_2 = 0.32$), the 3th robot stops. The 2nd robot keeps advancing at cruise speed ($c_1 = 0.2$).
- Figure 4.5f: Second segment falls below limits ($c_2 = 0.15$), the 3th robot accelerates to fast speed. The 2nd robot keeps advancing at cruise speed ($c_1 = 0.27$).

Sample videos can be found on our web site in:

http://www.ehu.es/ccwintco/index.php/DPI2006-15346-C03-03-Resultados-videos-control-centralizado

4.4 Conclusions and discussion

We have realized the physical proof-of-concept of the vision based control of a Linked Multi-component Robotic System (MCRS) performing the transportation of a hose-like object, in fact, a length of electrical cable. Many of the problems found were due to the need to start the whole experiment from scratch:

- Identification of magnetic hot-spots to avoid during experimentation.
- Robot assembly, design of the robot-hose attachment allowing rotations, robot painting of an appropriate color for enhanced robustness of image segmentation.

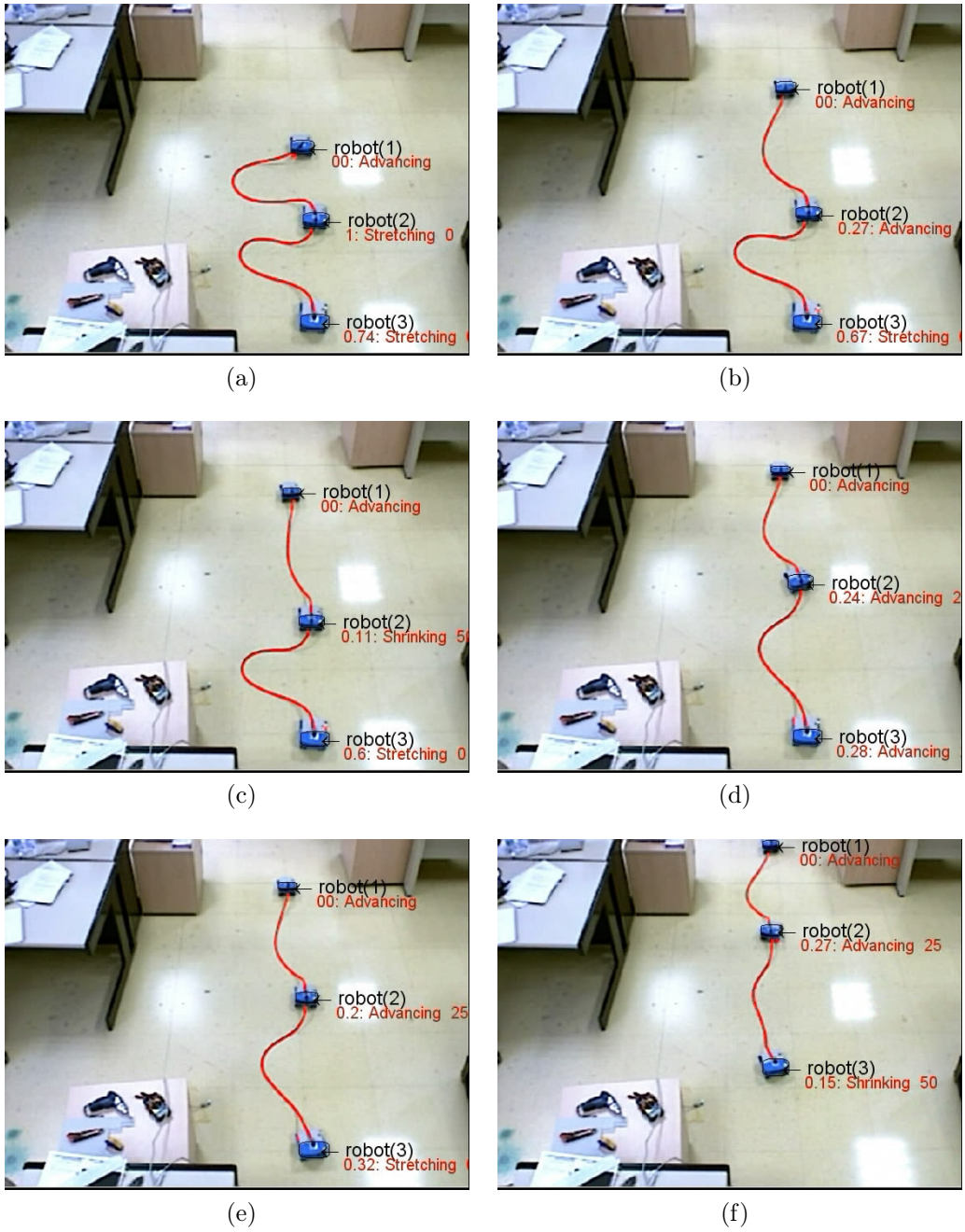


Figure 4.5: Frames extracted from the video of an example realization of the hose transportation task.

- Design and implementation of a communication protocol, due to the difficulties produced by channel shift in the wireless communication. Robust communication is an issue when working with robots like SR1.
- Shortcomings of the image segmentation process due to uncontrollable environmental conditions, like parts of the robot that were not taken into account and interfere the segmentation, or changes in the perceived color.

However, the experiment of physical realization of the task has serve also to demonstrate some specific traits of the hose-robot Linked MCRS. The first question that such a system may rise is: is there any substantial difference with a distributed MCRS (independent, not-linked robot units)? The experiment gives a definitive Yes. For such a simple behavior as the "follow-the-leader", the realization on a Distributed MCRS is relatively immediate. The robots may start motion at any moment and nothing hinders them in their motion. For a linked system it is fairly obvious that the hose-like passive element introduces quite a lot of dynamic interaction problems.

- The hose may be an obstacle for the advance of the robot.
- The hose may drag the advance of the robot, if the robots behind do not match speed with the leader.
- The weight and rigidity of the hose may impose motion constraints that may deviate the robot from the desired path.

The hose-like element introduces new perception and measurement needs: we need to observe (segment) the hose, and to compute some measure of its state that allows to build a system of rules that determines the robot behavior as a function of this state. The hose imposes also an ordering on the motion of the robots, both spatial and temporal, which is a new feature relative to the Distributed MCRS. In our experiment, the motion starts first for the first robot, after some while starts the second robot, an so on. The perception of the hose and the fine tuning of the effects of its state's measure are problems not shared by the Distributed MCRS paradigm. The experiment realization has been a success in the sense of demonstrating the inherent features of Linked MCRS and their need.

Companion work on this topic has been performed by Zelmar Echegoyen, and some of the future works suggested in this paradigm could be shared

with him. For instance: work on the simulated evaluation of different control heuristics, and the simulation of more complex environments and tasks. The final goal would be the realization with a distributed control and distributed perception of the deployment and transportation of a hose, however we need to define a collection of tasks that gradually approaches this goal. We are currently working on this and on the physical realization of the robot-hose system with other robotic modules.

Appendix A

LAM theoretical foundations

In this appendix we introduce some theoretical background about Lattice Associative Memories (LAM) and the Linear Mixing Model (LMM) used in the experimental work whose description appears in chapter 2. Section A.1 gives the basic definitions and fundamental results about Lattice Associative Memories. Section A.2 reviews the well-known Linear Mixing Model. Section A.3 gathers definitions and results linking underlying the approach of endmember induction based on Lattice Autoassociative Memories (LAAM). Section A.4 recalls the definition of the Endmember Induction Heuristic Algorithm (EIHA). Section A.5 recalls the definition of the incremental on-line variation of EIHA used for SLAM applications. Finally, section A.6 gives an alternative way to obtain the set of endmembers using the LAAM columns directly.

A.1 Definition of Lattice Associative Memories

The work on Lattice Associative Memories (LAM) stems from the consideration of the algebraic lattice structure $(\mathbb{R}, \vee, \wedge, +)$ as the alternative to the algebraic framework given by the mathematical field $(\mathbb{R}, +, \cdot)$ for the definition of Neural Networks computation. The LAM were first introduced in [95, 91] as Morphological Associative Memories, but we follow the new convention introduced in [92, 94] because it sets the works in the more general framework of Lattice Computing. The operators $\vee$ and $\wedge$ denote, respectively, the discrete max and min operators (resp. sup and inf in a continuous setting). Given a set of input/output pairs of pattern $(X, Y) = \left\{ \left(\mathbf{x}^{\xi}, \mathbf{y}^{\xi} \right) ; \xi = 1, .., k \right\}$,

a linear heteroassociative neural network based on the pattern's cross correlation is built up as $W = \sum_{\xi} \mathbf{y}^{\xi} \cdot \left(\mathbf{x}^{\xi}\right)'$. Mimicking this constructive procedure [95, 91] propose the following constructions of LAM as follows:

$$W_{XY} = \bigwedge_{\xi=1}^{k} \left[\mathbf{y}^{\xi} \times \left(-\mathbf{x}^{\xi}\right)'\right] \text{ and } M_{XY} = \bigvee_{\xi=1}^{k} \left[\mathbf{y}^{\xi} \times \left(-\mathbf{x}^{\xi}\right)'\right], \tag{A.1}$$

where $\times$ is any of the $\boxtimes$ or $\boxtimes$ operators. Here $\boxtimes$ and $\boxtimes$ denote the max and min matrix product [95, 91]. respectively defined as follows:

$$C = A \boxtimes B = [c_{ij}] \Leftrightarrow c_{ij} = \bigvee_{k=1,\ldots,n} \{a_{ik} + b_{kj}\}, \tag{A.2}$$

$$C = A \boxtimes B = [c_{ij}] \Leftrightarrow c_{ij} = \bigwedge_{k=1,\ldots,n} \{a_{ik} + b_{kj}\}. \tag{A.3}$$

When the input is different from the output $X \neq Y$ the LAM is a heteroassociative memory which we will call Lattice Heteroassociative Memory (LHAM). If $X = Y$ then the LAM are Lattice Autoassociative Memories (LAAM). Conditions of perfect recall by the LHAM and LAAM of the stored patterns proved in [95, 91] encouraged the research on them, because in the continuous case, the LAAM is able to store and recall any set of patterns: $W_{XX} \boxtimes X = X = M_{XX} \boxtimes X$, for any X. However, this result holds when we deal with noise-free patterns. Research on robust recall [86, 91, 93] based on the so-called kernel patterns lead to the notion of morphological independence, in the erosive and dilative sense, and finally to the definition of Lattice Independence (LI) and Strong Lattice Independence (SLI), that lead to the results reviewed in section A.3 below.

A.2 The linear mixing model

The linear mixing model (LMM) [59, 58] assumes that the data follows a linear model, which can be expressed as follows:

$$\mathbf{x} = \sum_{i=1}^{M} a_i \mathbf{s}_i + \mathbf{w} = \mathbf{S}\mathbf{a} + \mathbf{w}, \tag{A.4}$$

where $\mathbf{x}$ is the d-dimension pattern vector, $\mathbf{S}$ is the $d \times M$ matrix whose columns are the d-dimension vertices of the convex region covering the data corresponding to the so called endmembers $\mathbf{s}_i, i = 1, .., M$, $\mathbf{a}$ is the M-dimension abundance vector, and $\mathbf{w}$ is the d-dimension additive observation noise vector.

The LMM is applied when some item is assumed to be the combination of several pure items, called endmembers. In [59, 58] the items are light spectra in the context of hyperspectral image processing, here the items are the singular images which could be used as landmarks. Abundance coefficients correspond to the fraction of the contribution of each endmember to the observed item. From this physical interpretation, follows that the linear mixing model is subjected to two constraints on the abundance coefficients. First, to be physically meaningful, all abundance coefficients must be non-negative $a_i \geq 0, i = 1, .., M$, because the negative contribution is not possible in the physical sense. Second, to account for the entire composition, they must be fully additive $\sum_{i=1}^{M} a_i = 1$. As a side effect, there is a saturation condition $a_i \leq 1, i = 1, .., M$, because no isolate item can account for more than the existent material. From a geometrical point of view, these restrictions mean that we expect the endmembers in $\mathbf{S}$ to be affinely independent and that the convex region defined by them covers *all* the data points. This is a very important observation, because it has deep implications in the following reasoning about the inversion processes. The mixing inversion process (often called unmixing) consists in the estimation of the abundance coefficients, given the endmembers $\mathbf{S}$ and the observation data $\mathbf{x}$. The simplest approach is the unconstrained least squared error (ULSE) estimation given by:

$$\widehat{\mathbf{a}} = \left(\mathbf{S}^T\mathbf{S}\right)^{-1}\mathbf{S}^T\mathbf{x}. \tag{A.5}$$

The coefficients that result from equation (A.5) do not necessarily fulfill the non-negativity and full additivity conditions. The full additivity restriction can be incorporated in the abundance coefficients estimation using Lagrange multipliers [59, 58] introducing a correction term that moves the ULSE estimation to the hyperplane that satisfies the full additivity constraint. From the physical interpretation point of view, the non-negativity restriction is more fundamental. The Non-Negative Least Square estimation (NNLS) [63] can be used to enforce this condition. The estimation problem is treated as a quadratic programming problem with linear inequalities as constraints, solved iteratively. In each iteration the endmembers whose

abundances are positive are used to refine the estimation. It was shown in [59, 58] that non-negative estimations do not fulfill the full additivity condition. Besides, the NNLS computation time may be high, specially as the data dimension grows.

Some approaches to endmember determination (i.e., [94]) ensure that the computed endmembers define a convex polytope that covers *all* the data points, so that a proper convex inversion can be attempted. They find that NNLS provide meaningful abundances. However, they also found that the full additivity was not fulfilled. The heuristic algorithm EIHA described in section A.4 [45] always produces convex regions that lie *inside* the data cloud, so that enforcing the non-negative and full additivity restrictions would be impossible for some data points. Enforcing them for some points may introduce undesired distortions of their abundance values. Nevertheless we have made some attempts to apply NNLS to our data which have resulted in prohibitive computational times. For the this reasons, we use systematically the unconstrained estimation of equation (A.5) to compute the abundance coefficients.

Our reasoning for the application of the LMM to vision based mobile robot localization is as follows: When $M \ll d$ the computation of the convex coordinates can be interpreted as a dimension reduction process, or a feature extraction process as used in 2.4. Also, if the convex coordinates are a good representation of the data, then the convex polytope vertices are distinctive data items that can be used as landmarks for the representation of the data space, or as landmarks of the physical space in the SLAM process, as used in 2.5.

A.3 Lattice Independence and Lattice Autoassociative Memories

We gather some results from [94] that set the theoretical background for the approach to endmember induction applied.

Definition Given a set of vectors $\left\{\mathbf{x}^1, ..., \mathbf{x}^k\right\} \subset \mathbb{R}^n$ a *linear minimax combination* of vectors from this set is any vector $\mathbf{x} \in \mathbb{R}^n_{\pm\infty}$ which is a *linear*

minimax sum of these vectors:

$$x = \mathcal{L}\left(\mathbf{x}^1, ..., \mathbf{x}^k\right) = \bigvee_{j \in J} \bigwedge_{\xi=1}^{k} \left(a_{\xi j} + \mathbf{x}^\xi\right),$$

where J is a finite set of indices and $a_{\xi j} \in \mathbb{R}_{\pm\infty}$ $\forall j \in J$ and $\forall \xi = 1, ..., k$.

Definition The *linear minimax span* of vectors $\left\{\mathbf{x}^1, ..., \mathbf{x}^k\right\} = X \subset \mathbb{R}^n$ is the set of all linear minimax sums of subsets of X, denoted $LMS\left(\mathbf{x}^1, ..., \mathbf{x}^k\right)$.

Definition Given a set of vectors $X = \left\{\mathbf{x}^1, ..., \mathbf{x}^k\right\} \subset \mathbb{R}^n$, a vector $\mathbf{x} \in \mathbb{R}^n_{\pm\infty}$ is *lattice dependent* if and only if $\mathbf{x} \in LMS\left(\mathbf{x}^1, ..., \mathbf{x}^k\right)$. The vector $\mathbf{x}$ is *lattice independent* if and only if it is not lattice dependent on X. The set X is said to be *lattice independent* if and only if $\forall \lambda \in \{1, ..., k\}$, $\mathbf{x}^\lambda$ is lattice independent of $X \backslash \left\{\mathbf{x}^\lambda\right\} = \left\{\mathbf{x}^\xi \in X : \xi \neq \lambda\right\}$.

The definition of lattice independence supersedes and improves the early definitions [93] of erosive and dilative morphological independence, which, however, have more intuitive appealing. Nevertheless, this definition has the additional advantage of establishing a formal parallelism with the definition of linear independence.

Definition A set of vectors $X = \left\{\mathbf{x}^1, ..., \mathbf{x}^k\right\} \subset \mathbb{R}^n$ is said to be *max dominant* if and only if for every $\lambda \in \{1, ..., k\}$ there exists and index $j_\lambda \in \{1, ..., n\}$ such that

$$x^\lambda_{j_\lambda} - x^\lambda_i = \bigvee_{\xi=1}^{k} \left(x^\xi_{j_\lambda} - x^\xi_i\right) \forall i \in \{1, ..., n\}.$$

Similarly, X is said to be *min dominant* if and only if for every $\lambda \in \{1, ..., k\}$ there exists and index $j_\lambda \in \{1, ..., n\}$ such that

$$x^\lambda_{j_\lambda} - x^\lambda_i = \bigwedge_{\xi=1}^{k} \left(x^\xi_{j_\lambda} - x^\xi_i\right) \forall i \in \{1, ..., n\}.$$

The expressions that compound this definition appeared in the early theorems about perfect recall of Morphological Associative Memories [95, 91]. Their value as an identifiable property of the data has been discovered in the context of the formalization of the relationship between strong lattice independence, defined below, and the classical affine independence.

Definition A set of lattice independent vectors $\left\{\mathbf{x}^1, ..., \mathbf{x}^k\right\} \subset \mathbb{R}^n$ is said to be *strongly lattice independent* (SLI) if and only if X is max dominant or min dominant or both.

As said before, min and max dominance are the conditions for perfect recall. Per construction, the column vectors of Lattice Autoassociative Memories are min or max dominant, depending of their erosive or dilative nature, therefore they will be strongly lattice independent, *if* they are lattice independent.

Conjecture A.3.1 *[94] If $X = \left\{\mathbf{x}^1, ..., \mathbf{x}^k\right\} \subset \mathbb{R}^n$ is strongly lattice independent then X is affinely independent.*

This conjecture (stated as theorem in [92]) is the key result whose proof would relate the linear convex analysis and the nonlinear lattice analysis. If true, it means that the construction of the LAAM provides the starting point for obtaining sets of affine independent vectors that could be used as endmembers for the unmixing algorithms described in section A.2.

Theorem A.3.2 *[94] Let $X = \left\{\mathbf{x}^1, ..., \mathbf{x}^k\right\} \subset \mathbb{R}^n$ and let W (M) be the set of vectors consisting of the columns of the matrix W_{XX} (M_{XX}.). Let $F(X)$ denote the set of fixed points of the LAAM constructed from set X. There exist $V \subset W$ and $N \subset M$ such that V and N are strongly lattice independent and $F(X) = F(V) = F(N)$ or, equivalently, $W_{XX} = W_{VV}$ and $M_{XX} = M_{NN}$.*

The key idea of this theorem is to test the lattice independence of the already known as min or max dominant sets of vectors. Removing lattice dependent vectors will not affect this min/max dominance property. The smart way to test lattice dependence lies in the fact that removing a lattice dependent vectors does not alter the set of fixed points of the remaining ones. This theorem is proved following a constructive reasoning, giving way to an algorithm for the construction of the set of affine independent sets of vectors from LAAM, the EIHA, discussed in [45, 94].

A.4 Endmember Induction Heuristic Algorithm (EIHA)

The Endmember Induction Heuristic Algorithm (EIHA) [45] is a tool for the extraction of an affine independent set of vectors from a set of input data

based on the results of section A.3. Let us denote $\left\{\mathbf{f}(i) \in \mathbb{R}^d : i = 1, .., n\right\}$ a set of high dimensional data vectors, which could correspond, in our works, to the images acquired by a mobile robot's camera. Vectors $\overrightarrow{\boldsymbol{\mu}}$ and $\overrightarrow{\boldsymbol{\sigma}}$ are, respectively, the mean vector and the vector of standard deviations computed componentwise over the units of data sample, α the noise correction factor, and E the set of already discovered endmembers. The noise amplitude of the additive noise in equation (A.4) is $\overrightarrow{\boldsymbol{\sigma}}$, the patterns are corrected by the addition and subtraction of $\alpha\overrightarrow{\boldsymbol{\sigma}}$, before being presented to the LAAM's. The gain parameter α controls the amount of flexibility in the discovering of new endmembers. For $\mathbf{e} \in \mathbb{R}^d$, let $\mathbf{b}(\mathbf{e})$ denote the binary version of $\mathbf{e}$ defined by $b(\mathbf{e})_i = 1$ if $e_i > 0$ and $b(\mathbf{e})_i = 0$ if $x_i \leq 0$, where $b(\mathbf{e})_i$ denotes the ith coordinate of $\mathbf{b}(\mathbf{e})$. Finally, we will denote by X the set of binary signatures used to build the lattice memories, and I the set of sample indices corresponding to the endmembers selected by the algorithm.

The detailed description of the steps in the heuristic algorithm is presented as algorithm A.1. The starting endmember set consists of a randomly image. However, this selection is not definitive, because the algorithm may later change this endmember for another, more extreme, one. The noise correction parameter α has a great impact on the number of endmembers found. Low values imply large number of endmembers. It determines if a vector is interpreted as a random perturbation of an already selected endmember.

This algorithm does not need *a priori* information about the nature of the distinctive images that we want to detect.

A.5 An on-line EIHA for SLAM

In the experimental works reported in section 2.5 the EIHA is used for unsupervised landmark selection in SLAM applications. The use of the EIHA for SLAM purposes had to cope with some issues typical of that paradigm. The main issue comes from the fact that the full set of the data will not be available from the start, since SLAM applications are supposed to work in unknown environments and information about it will be acquired progressively, as more space is explored. So, the EIHA has to be adapted to work on-line, processing the incoming image feature vectors and making decisions based on the available information, and under the assumption that these decisions can not be reverted.

In the version of the algorithm described in section A.4, vectors $\overrightarrow{\boldsymbol{\mu}}$ and

Algorithm A.1 Endmember Induction Heuristic Algorithm (EIHA)

1. Shift the data sample to zero mean $\left\{\mathbf{f}^c(i) = \mathbf{f}(i) - \overrightarrow{\boldsymbol{\mu}}; i = 1, .., n\right\}$.

2. Initialize the set of endmembers $E = \{\mathbf{e}^1 = \mathbf{f}^c(i^*)\}$ where i^* is a randomly picked sample index. Initialize the set of lattice independent binary signatures $X = \{\mathbf{x}^1\}$ where $\mathbf{x}^1 = \mathbf{b}(\mathbf{e}^1)$. The initial set of endmember sample indices is $I = \{i^*\}$.

3. Construct the LAAM's based on the lattice independent binary signatures: M_{XX} and W_{XX}.

4. For each incoming image feature vector $\mathbf{f}^c(i)$

   (a) Compute the noise corrections sign vectors$\mathbf{f}^+(i) = \mathbf{b}\left(\mathbf{f}^c(i) + \alpha\overrightarrow{\boldsymbol{\sigma}}\right)$ and $\mathbf{f}^-(i) = \mathbf{b}\left(\mathbf{f}^c(i) - \alpha\overrightarrow{\boldsymbol{\sigma}}\right)$

   (b) Compute $y^+ = M_{XX} \boxtriangle \mathbf{f}^+(i)$

   (c) Compute $y^- = W_{XX} \boxtriangledown \mathbf{f}^-(i)$

   (d) If $y^+ \notin X$ or $y^- \notin X$ then $\mathbf{f}^c(i)$ is a new endmember to be added to E, execute once 3 with the new E and resume the exploration of the data sample. Add i to the set of indices I.

   (e) If $y^+ \in X$, let k be the index in E of the corresponding endmember. If $\mathbf{f}^c(i) > \mathbf{e}^k$ then execute step 4g.

   (f) If $y^- \in X$, let k be the index in E of the corresponding endmember. If $\mathbf{f}^c(i) < \mathbf{e}^k$ then execute step 4g.

   (g) The new data sample is more extreme than the stored endmember, then substitute $\mathbf{e}^k$ in E with $\mathbf{f}^c(i)$. Index i substitutes the corresponding index in I.

5. The output set of endmembers is the set of original data vectors $\{\mathbf{f}(i) : i \in I\}$ corresponding to the sign vectors selected as members of E.

$\overrightarrow{\boldsymbol{\sigma}}$ were used, being respectively the mean vector and the vector of standard deviations computed over the data sample. The first effect of the on-line nature of SLAM is that it is not possible to use them, because we can not have *a priori* estimations. An adaptive estimation strategy can not be used, because this information would be used to perform decisions (selection of an endmember) which can not be undone when the mean and covariance matrix estimations are updated. In other words, each image must be treated independently, and we can not assume it as a sample from a stationary distribution because the images will change unpredictably as the robot moves on. As in previous version, α denotes the noise tolerance and E the set of already discovered endmembers, corresponding to the already identified view landmarks for SLAM.

Another of the effects of the on-line processing is that it can no longer be assumed that the amplitude of the additive noise in equation (A.4) is σ, therefore it can not be computed an interval based on the addition and subtraction of $\alpha\sigma$, to the feature vector $\mathbf{f}(t)$ before being presented to the LAAM's (note that t is used as index instead of i to indicate a temporal relation of the data samples). Nevertheless, this interval can be computed applying a scaling factor of the input feature vector magnitude. The gain parameter α continues to control the amount of flexibility in the discovering of new endmembers. Other effect of the on-line processing is that the data can not be shifted to zero mean, but as the algorithm is going to be applied to the low frequency of the Discrete Cosine Transform (DCT), it can be expected that their coefficients will be placed around zero. The expression $\mathbf{x} > \mathbf{0}$ denotes the construction of the binary vector $(\{b_i = 1 \text{ if } x_i > 0; b_i = 0 \text{ if } x_i \leq 0\}; i = 1, .., n)$.

The steps in the procedure of on-line endmember induction are described in the algorithm A.2.

A.6 Endmember Induction from the LAAM matrix

Theorem A.3.2 gives also a key for single LAAM endmember induction. If a LAAM is computed from the full dataset, theorem A.3.2 entails that a subset of the columns of the obtained LAAM is strongly lattice independent. Searching for this subset can be an approach to obtain a suitable set of endmembers.

Algorithm A.2 An on-line EIHA for SLAM applications.

1. Initialize the set of endmembers with the first data sample $E = \{\mathbf{e^1} = \mathbf{f}(0)\}$. Initialize the set of morphologically independent binary signatures $X = \{\mathbf{x^1}\} = \{\mathbf{e^1} > \mathbf{0}\}$. The initial set of endmember sample indices is $I = \{0\}$.

2. Construct the LAAM's based on the lattice independent binary signatures: M_{XX} and W_{XX}.

3. For each incoming image feature vector $\mathbf{f}(t)$

   (a) compute the noise tolerance sign vectors $\mathbf{f}^+(t) = (\mathbf{f}(t) + \alpha\mathbf{f}(t) > 0)$ and $\mathbf{f}^-(t) = (\mathbf{f}(t) - \alpha\mathbf{f}(t) > 0)$

   (b) compute $y^+ = M_{XX} \boxtriangle \mathbf{f}^+(t)$

   (c) compute $y^- = W_{XX} \boxtriangledown \mathbf{f}^-(t)$

   (d) if $y^+ \notin X$ or $y^- \notin X$ then $\mathbf{f}(t)$ is a new endmember to be added to E. Compute its signature $\mathbf{f}(t) > \mathbf{0}$, adding it to the set of morphologically independent binary signatures X. Add $\mathbf{f}(t)$ to E. Execute (2) once with the new E and resume the exploration of the data sample. Add t to the set of indices I.

   (e) if $y^+ \in X$, let k be the index in E of the corresponding endmember. If $\mathbf{f}(t) > \mathbf{e}^k$ then execute step (3g).

   (f) if $y^- \in X$, let k be the index in E of the corresponding endmember. If $\mathbf{f}(t) < \mathbf{e}^k$ then execute step (3g).

   (g) The new data sample is more extreme than the stored endmember, then substitute $\mathbf{e}^k$ in E with $\mathbf{f}(t)$. Index t substitutes the corresponding index in I.

4. The output set of endmembers is the set of original data vectors $\{\mathbf{f}(t) : t \in I\}$ corresponding to the sign vectors selected as members of E.

Algorithm A.3 Endmember Induction from a single LAAM

1. Compute the min auto-associative memory W_{XX} from the data.
2. Compute the conjugate of each element in the matrix, obtaining the conjugate matrix $-W_{XX}$.
3. The rows of the matrix $-W_{XX}$ is considered as the strong lattice independent set. The minimum of each dimension in the data set is added to the corresponding dimensions of the strong lattice independent vectors.
4. Select the vectors with minimum mutual information to obtain a set with smaller cardinality than the data dimensionality.

Following this approach, an algorithm for a single LAAM endmember induction is proposed, composed of a four step process described in the algorithm A.3.

This approach has some advantages for the desired application. First, as it does not have a random start like the EIHA, the resulting endmember set is always the same for a given dataset. And secondly, the number of the endmembers extracted can be determined, instead of having a variable number of endmembers depending on the tuning of the α parameter. This can be helpful if we want a fixed size feature vector.

However, this approach also has several unsolved problems which need to be considered:

- The number of strongly lattice (and affinely) independent vectors is quite high, close to the space dimensionality. As the desired goal is to use the endmembers to obtain the convex coordinates and use them as feature vectors, some dimensionality reduction is desired. The approach followed selects a smaller set of endmembers, but the optimality of this selection can not be guaranteed.
- The assumption that a subset of a set strongly lattice independent is also strongly lattice independent has not been proved yet.
- The convex polytope defined by set V of strongly lattice independent vectors is not related to the data points. For the desired application,

the data point set needs to be covered as much as possible, approaching the data convex hull, in order to have meaningful convex coordinates.

- We do not know of appropriate ways to perform on-line endmember induction following this approach, which can be of use for SLAM related applications.

Appendix B

Computational Intelligence tools

In this appendix we will give a brief review of the Computational Intelligence tools used in the works presented in chapter 3. First, two Competitive Neural Network (CNN) approaches, namely the Self-Organizing Maps (section B.1) and the Neural Gas Networks (section B.2), will be presented. We will follow the original notations found in the literature, so there are minor notation variations between the presentations of both algorithms. Both CNN were used in chapter 3 to fit the 3D point cloud. While some CNN approaches, for instance SOM, are used to perform a non-linear dimensionality reduction, the emphasis here is made into obtaining a reduction on the number of points treated while preserving most of the spatial information. In other words, the CNN are applied to obtain a set of codevectors optimal in the sense of minimal distortion relative to the 3D camera measurements. After that, in section B.3 we review the definition of the Evolution Strategies that where used to find the optimal transformation between consecutive views from the 3D ToF camera as the optimal registration between the codevectors computed for these consecutive views. Finally, section B.4 contains a description of the kd-trees and its use for Nearest Neighbor search.

B.1 Self-Organizing Maps

A Self-Organizing Map (SOM), known also as Kohonen network [61, 60], is a computational tool for vector quantization, visualization and analysis of high-dimensional data. This type of artificial neural network uses unsupervised learning to produce a low-dimensionality, discretized representation

of the input space called *map*. In this map, the relationships on the original high-dimensional data are converted into simple geometric relationships, preserving their topological and metric characteristics. This preservation, achieved by using a neighborhood function, makes SOM different from other Artificial Neural Networks.

B.1.1 Learning Algorithm

The SOM defines an ordered mapping, or projection, of the space of input data vectors $\mathbf{x} \in \mathbb{R}^n$ onto an array of nodes, usually in the form of a two-dimensional grid with a rectangular or hexagonal neighborhood pattern of the node locations. Each node (or neuron) of the grid has an associated *reference vector* $\mathbf{m}_i \in \mathbb{R}^n; i = 1, ..., M$ (figure B.1). The collection of all the reference vectors forms a non-parametric model of the input data distribution. The input data can be of any form, as long as a distance function $d(\mathbf{x}, \mathbf{m}_i)$ can be defined over it. Here we use as distance function the Euclidean distance.

We denote $\mathbf{x}(t)$ a sample from the input process which is fed to the SOM. The reference vectors at the map's nodes are updated following a competitive adaptation rule:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)\, h_{c,i}(t) \left[\mathbf{x}(t) - \mathbf{m}_i(t)\right], \tag{B.1}$$

where $\alpha(t)$ is a monotonically decreasing scalar learning factor that defines the magnitude of the correction. The neighborhood function $h_{ci}(t)$ is a smoothing kernel defined on the space of neuron indices centered on the best-matching node identified by c. This node $\mathbf{m}_c(t)$ is the one which satisfies

$$d(\mathbf{x}(t), \mathbf{m}_c(t)) = \min_i \{d(\mathbf{x}(t), \mathbf{m}_i(t))\}, \tag{B.2}$$

or, in our case with Euclidean distance,

$$\|\mathbf{x}(t) - \mathbf{m}_c(t)\| = \min_i \{\|\mathbf{x}(t) - \mathbf{m}_i(t)\|\}. \tag{B.3}$$

The neighboring function $h_{ci}(t)$ is usually defined as a function of the map locations of the nodes,

$$h_{ci}(t) = h(\|r_c - r_i\|, t),$$

where $r_c \in \mathbb{R}^2$ and $r_i \in \mathbb{R}^2$ will be the coordinates of the nodes c and i in a map with the typical bi-dimensional lattice. One of the simpler definitions commonly used is by establishing a neighborhood radius σ around $\mathbf{m}_c$ which defines a neighborhood set

$$N_c = [\mathbf{m}_i | \, \|r_c - r_i\| < \sigma] \, ,$$

(figure B.2). In this case, the neighborhood function will be

$$h_{ci}(t) = \alpha(t) \, ,$$

when $\mathbf{m}_i \in N_c$ and $h_{ci}(t) = 0$ otherwise, with $0 < \alpha(t) < 1$. Another widely used definition is as a radial basis function,

$$h_{ci}(t) = \exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right) , \tag{B.4}$$

where $\sigma(t)$ is the width of the neighborhood function, which decreases monotonically along time, as does $\alpha(t)$.

Nodes had to be initialised with some value at the start of the algorithm. Although $\mathbf{m}_i(0)$ can be initialised with random values and still the process may converge to a good map of the data, this approach can be quite slow. However, the validity of a random start proves that any arbitrary values can be used for initialization. Thus, an ordered initial state is desirable. One successful approach to this initialization is to select initial values from the subspace spanned by the eigenvectors corresponding to the two largest principal components of input data. Those values are arranged in a rectangular array, which is used to give values to their approximately correspondent nodes in the SOM.

B.1.2 SOM for 3D data fitting

One of the main characteristics of the SOM is its ability to preserve the topological relationships between the input data points. That is, the SOM acts like a nonlinear projection of the input data distribution $p(x)$. When applied to $\mathbb{R}^3$ data (like a cloud of 3D points), the elastic network of the SOM will be spread along the distribution of the input data points, creating a model or projection of the data which preserves its three-dimensional shape. This

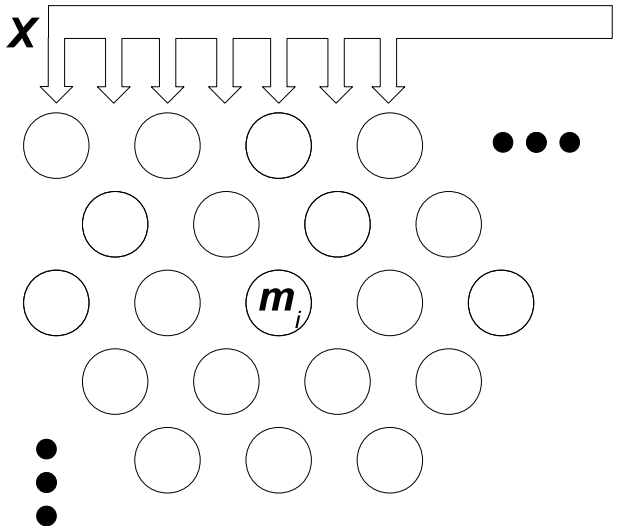


Figure B.1: Array of nodes in a two dimensional SOM grid defined over an hexagonal lattice.

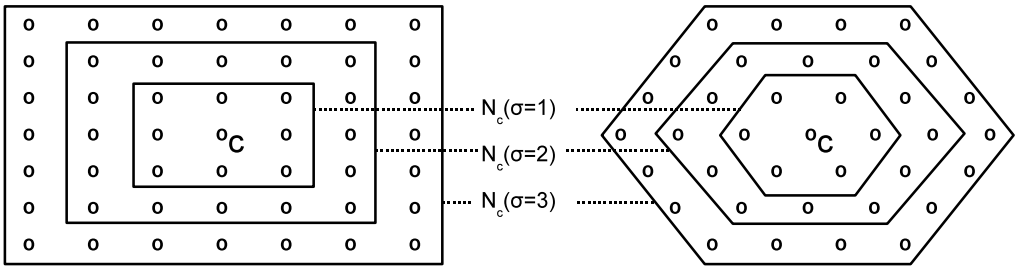


Figure B.2: Examples of topological neighborhood in rectangular and hexagonal maps.

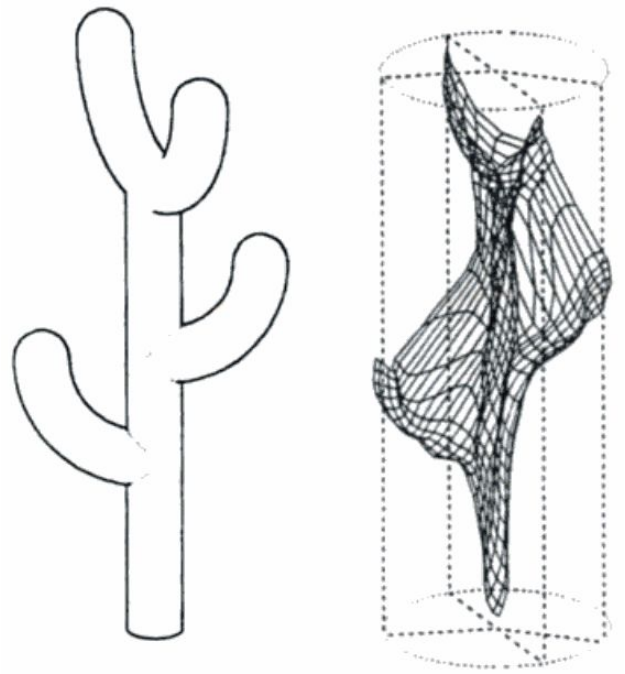


Figure B.3: Projection of the distribution $p(x)$ shown in the left by a SOM, shown to the right. [61]

model has, however, a drawback caused by the fixed neighborhood relationships inside the map, as nodes can fall in spaces without input data when trying to link separated high data density areas. An example of this data fitting as shown in [61] can be seen in figure B.3.

B.2 Neural Gas Networks

Neural Gas (NG) networks [69] are another type of Artificial Neural Network for Vector Quantization. Neural Gas networks were developed in an attempt to achieve an optimal utilization of all neural units. We have already seen that for the SOM preservation of the topology was prioritised, which resulted in "misuse" of some of the nodes in the map as the network tried to span the lattice of nodes to cover the topological structure that is the domain of the distribution of the input data when it is not convex (e.g, like the example seen in figure B.3, where data was distributed along several "branches"). NG

networks has been proposed as a flexible network able to quantize those topologically heterogeneous structured data and at the same time learning the relationships between the input signals without specifying a network topology.

B.2.1 Learning algorithm

The main feature of the neural gas network is that the weights $\mathbf{w}_i$ of the nodes adapt to the input data without taking in consideration any kind of network topology, being affected only by the relative distances between the neural units within the input space. That means also that the network nodes constantly change their neighborhood relationships with other nodes as they adapt to the data. The NG network is then defined by a set of vectors $\mathbf{w}_i \in \mathbb{R}^n, i = 1, ..., N$.

Given an input vector $\mathbf{v}(t)$, presented to the network at time t, the relationship in the input space between $\mathbf{v}(t)$ and the reference vectors stored at the nodes in the network will be given by the set of distortions $D_v = \{\|\mathbf{v}(t) - \mathbf{w}_i(t)\|, i = 1, ..., N\}$. Ordering this set will determine the proximity of the neurons to the data and the adjustment of the weights $\mathbf{w}_i$ and its influence in the actualization of its value. This actualization is computed in step t as

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \epsilon \cdot f_i(D_v) \cdot (\mathbf{v}(t) - \mathbf{w}_i(t)), \tag{B.5}$$

where the step size $\epsilon \in [0, 1]$ describes the overall extend of the modifications and $f_i(D_v) \in [0, 1]$ describes the influence of the arrangement of the $\mathbf{w}_i(t)$ within the input space. This influence depends on the number k_i of nodes that are closer to the input data than the node i, being maximum in the node i_0 which satisfies

$$\|\mathbf{v}(t) - \mathbf{w}_{i_o}(t)\| = \min_j \|\mathbf{v}(t) - \mathbf{w}_j(t)\|. \tag{B.6}$$

Besides, $i_k, k = 0, ..., N-1$ is the node for which there are exactly k nodes which satisfy $\|\mathbf{v}(t) - \mathbf{w}_j(t)\| < \|\mathbf{v}(t) - \mathbf{w}_{i_k}(t)\|$, and $i_0, i_1, ..., i_{n-1}$ will be then the order of the set D_v. The influence function $f_i(D_v)$ will be given by a function $f(k_i)$ which will equal to 1 for $k_i = 0$ and decrease as k_i increases. The typical form of this function is

$$f_i(D_v) = f(k_i) = e^{-k_i/\lambda}, \tag{B.7}$$

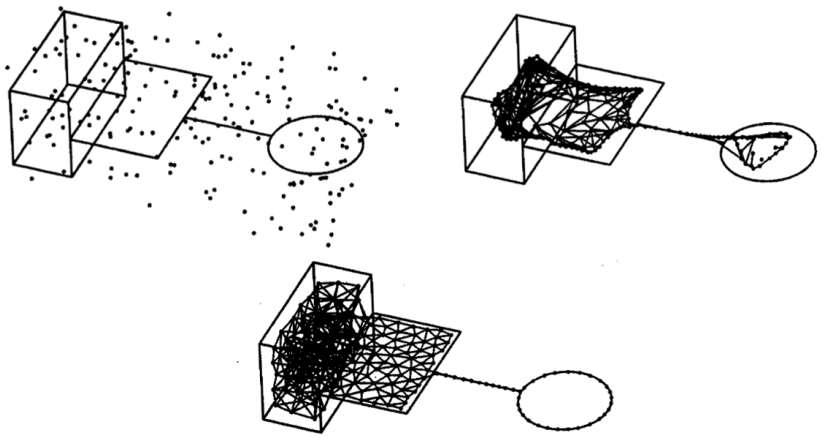


Figure B.4: Three steps of a Neural Gas fitting topologically heterogeneous data from the starting configuration to the final projection of the data. [69]

where λ determines the number of nodes significantly changing their synaptic weights, which is a parameter similar to the shrinking neighborhood of the SOM.

If there is any interest in data visualization, topological relationships in the space of node indices between nodes of the network can be established by means of a $C_{ij} \in \{0, 1\}$ connections matrix. When an input data $\mathbf{v}$ is fed to the network, a connection between node i_0 and i_1 is established by setting the element $C_{i_0 i_1}$ from 0 to 1. This connection has a maximum lifetime T, after which it is considered broken and put again to 0, unless some other input data renews it, resetting its age t_{ij} to zero.

B.2.2 Neural Gas for 3D data fitting

The main objectives of the Neural Gas networks was the quantization of the data under an heterogeneous distribution avoiding nodes lying outside the data space. The flexible neighborhood relationships of the Neural Gas allows this. In figure B.4 we can see an example from [69], where a Neural Gas fits tightly a very heterogeneously spatially distributed data. It can be appreciated how it fits the data without "losing" nodes in spaces without input data.

B.3 Evolution Strategies

Evolution Strategies (ES) as described in [9] belongs to the family of the nature-inspired Evolutionary Algorithms. ES are methods of stochastic iterative optimization based on the adaptation of a population of candidate solutions by means of an evolutionary approach that uses mutation, recombination and selection on those individuals as tools to evolve them, obtaining solutions closer to the optimum each iteration of the algorithm. The main difference between Evolution Strategies and Genetic Algorithms is the representation of the traits of the individuals of the populations. While in GAs the genetic representation of the solution encoded by each individual is usually an array of bits, in ESs the traits are usually a real-valued n-dimensional vector. Also, the main search operator of the ES is the mutation, although modern approaches also make use of recombination as a way of avoid more easily local optima.

B.3.1 The ES Algorithm

As stated above, the objective of the ES is to optimize a given objective function F with respect to a parameter set $y = (y_1, y_{2,}...)$, referred as *objective parameters*. This set can be any finite data structure, typically a real valued vector. The ES operate on populations P of individuals $\mathfrak{a}$. An individual $\mathfrak{a}_k$ has the form

$$\mathfrak{a}_k = (y_k, s_k, F(y_k)), \tag{B.8}$$

where y_k are its related object parameters and $F(y_k)$ its objective function value $F_k = F(y_k)$ or *fitness*, s_k is an optional set of *endogenous strategy parameters* used to control certain parameters of the strategy, like adaptive mutation, in self-adaptive ESs.

Each generation, or step, of the ES a set of μ parents is used to generate a number λ of offspring individuals $\tilde{\mathfrak{a}}$, mixing ρ parents to generate each offspring. The parameters μ, ρ and λ are the *exogenous strategy parameters*, and they are constant for all the execution of the ES. In following steps, the parent population can be selected from the offspring (“,” in the notation) or from the union of the offspring and parent population (“+” notation). The notation $(\mu/\rho^{+}_{,}\lambda)$ will, thus, express the configuration of each ES. For example, a $(\mu + \lambda)$-ES does not use recombination for generating offspring

Algorithm B.1 The ES general algorithm.

1. Initialize parent population $P_\mu = \{\mathfrak{a}_1, ..., \mathfrak{a}_\mu\}$.
2. Generate a new offspring population $P_\lambda = \{\tilde{\mathfrak{a}}_1, ..., \tilde{\mathfrak{a}}_\lambda\}$. Each offspring $\tilde{\mathfrak{a}}$ is generated:
   (a) Select ρ parents from P_μ
   (b) Recombine the selected parents.
   (c) (optional) Mutate the strategy parameters set s.
   (d) Mutate the object parameters set y.
3. Select a new parent population from
   - $(\mu\ , \lambda)$ case: the offspring population P_λ.
   - $(\mu + \lambda)$ case: the combination of parent P_μ and offspring P_λ populations.
4. Repeat from 2 until termination criterion is matched.

and a $(\mu/\rho + 1)$-ES is a *steady state* strategy in which each generation only one offspring will be generated. The outline of an ES algorithm, following [9] notation, can be seen in algorithm B.1.

Selection is the process that guides the algorithm towards the optimal solution. This is achieved by means of selecting the best fitted individuals (i.e. the ones with better F_k) from the population to build the parent population P_μ. That guarantees that only the best solutions will be used for the next step, ensuring that the best solution in following generation will be at least as good as in the previous one. In $(\mu\ , \lambda)$ strategies a necessary condition is that $\mu < \lambda$, otherwise no convergence can be guaranteed as no selection is performed, since all offsprings are selected as parents and best individuals of previous generation are lost. $(\mu + \lambda)$ do not have this restriction, since both parents and offspring are used to obtain the next generation, which guarantees the survival of the best individual found so far. The selection techniques which preserve the best individuals are called *elitist*. The $(\mu\ , \lambda)$ strategies are not elitist, therefore they can not be proved to converge to the

global optima in any case.

The variability of the individuals in ES is achieved mainly with the mutation operator. There is no fixed mutation operator design and it is usually problem dependant. In any case, the mutation operator has to assure *reachability*: each point of the search space has to be reachable within a finite number of iterations. Also, as variation is the way to explore the search space (in contrast with the selection, which guides the search to promising areas), the introduction of bias in the search through the mutation operator should be avoided, satisfying the condition of *unbiasedness*. Finally, if a self-adapting ES is desired, the *scalability* of the operator should be also guaranteed, as it should adapt to the properties of the fitness landscape.

The second operator which introduces variation in the population is the recombination. The search performed by the mutation only takes into account the information from one parent, which reduces the search to a limited region around it. Recombination, on the contrary, uses the information of ρ parents to search in a much wider space, increasing the variability of the offspring and, at the same time, keeping the properties of the parents. There are two common classes of recombination: the discrete recombination, in which each trait of the offspring is inherited from one of the parents selected randomly, and intermediate recombination, in which each trait is the arithmetic mean of that trait in the parents.

B.3.2 Evolution Strategies for registration

The registration problem can be defined as the optimization problem in which, given two clouds of points, we look for the optimal spatial transformation that provides the best matching between them. Matching goodness is measured as a function of the distance between the transformed clouds of points. Usually only the points in one of the clouds is transformed to match the points in other one. Point correspondence need to be established in order to compute the global distance among clouds of points.

An Evolution Strategy can be applied to search for the solution of this problem in a direct way, where the objective parameters set y_k will be the parameters of the sought transformation matrix and the fitness function $F(y_k)$ will be a defined distance function between the transformed and goal clouds of points, e.g. the mean or the median of the euclidean distances between the corresponding points in the clouds.

B.4 *k*d-trees

A kd-tree [6, 7] (short for k-dimensional tree) is a data structure in the form of a binary tree. It is a special sub-case of the binary space partitioning (BSP) trees, which partitions a k-dimensional data space, storing a finite number of points in that space. The kd-trees are used for very fast searching and nearest-neighbor queries in high-dimensional spaces and/or with big data point collections.

B.4.1 Construction

The kd-tree will be a binary tree in which each node is a k-dimensional point. Each non-leaf node generates an hyperplane perpendicular to one of the dimensions of the space which partitions that space in two subspaces. Each of those subspaces will be stored in the subtrees. There is no fixed splitting node selection method, as different approximations could be optimal for different problems. Usually the tree is built in a balanced way, trying that every leaf node is at approximately the same distance from the root. This requires that each partition contains the same number of points. This is achieved using as non-leaf nodes the hyperplanes that cut the partitioned axis at the median points of the distribution of the data point projections into this axis (i.e. the coordinate values). Also, consecutive non-leaf nodes generate an hyperplane cutting the space following a different axis (e.g. the root node's hyperplane cuts the space in the X axis, the subspaces generated are partitioned following the Y axis, etc.). This process is repeated in a recursive way until all the points are in the tree, being the leaf nodes the points which do not have more space to partition. In algorithm B.2 this recursive step is outlined as is described in [20] and figure B.5 shows an example of 2-dimensional kd-tree generated following it. A kd-tree constructed in this way will use $O(n)$ storage and can be constructed in $O(n \log n)$ time.

B.4.2 Nearest Neighbor search

One of the main uses of kd-trees is nearest neighbor search of one input data point into the data used used to build the tree. The properties of the tree allow for large portions of the space to be quickly discarded from the search, making the search highly efficient and fast.

Algorithm B.2 kd-tree construction recursive algorithm.

Given a set of points $P = \{p_1, p_2, ..., p_n\}$ and a current depth d.

1. If $P = \{\}$, return empty tree.
2. Else
   (a) Select splitting axis a in function of d.
   (b) Find the median point $m \in P$ with respect to a.
   (c) Split P in two subsets $m^+ = \{p_i(a) < m(a)\}$ and $m^- = \{p_i(a) > m(a)\}$.
   (d) Build the kd-tree T_l with $P = m^+$ and $d = d + 1$.
   (e) Build the kd-tree T_r with $P = m^-$ and $d = d + 1$.
   (f) Make the tree T with root m and branches T_l and T_r.
   (g) return T.

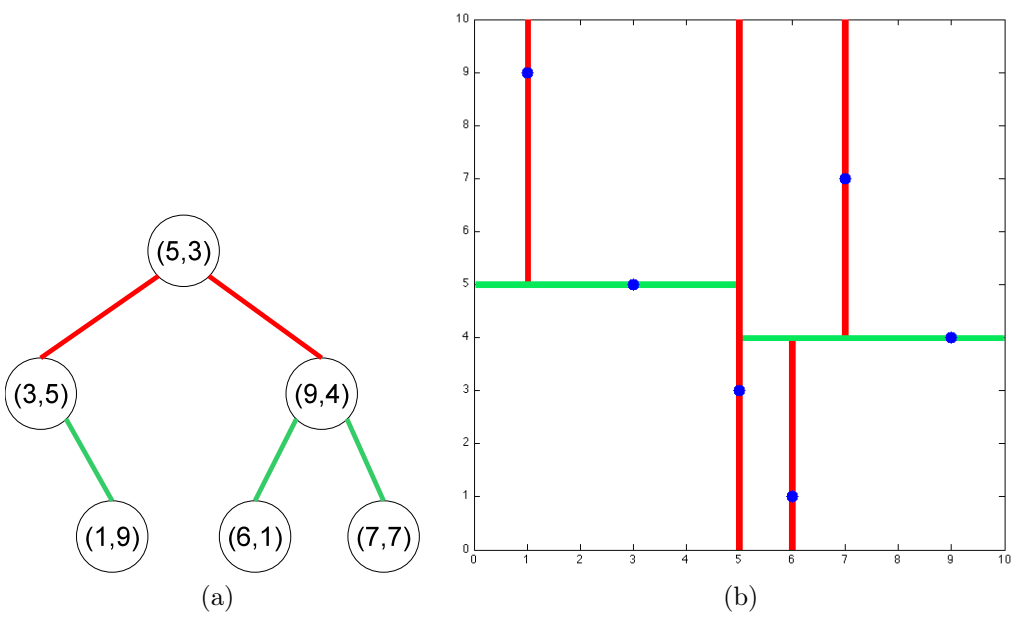


Figure B.5: Example of the kd-tree (B.5a) generated with a set of 2-dimensional points and the space partitioning it represents (B.5b).

Algorithm B.3 Nearest neighbor search algorithm with kd-trees.

Given a kd-tree T and a point p

1. Follow the tree until a leaf node is found, marking the visited branches.
2. Mark the found leaf as current best n_b and $R = \|n_b - p\|$.
3. Unwind the path. For each node n_c.
   (a) If $\|n_c - p\| < R$ then $n_b = n_c$ and $R = \|n_c - p\|$
   (b) If the hyperplane intersects the hypersphere with origin n_b and radius R:
      i. Search the unvisited branch and get its best n'_b and R'.
      ii. If $R' < R$ then $n_b = n'_b$ and $R = R'$.
   (c) Go up in the tree.

The search is performed traversing the tree from root to the leaf that identifies the region of the space. In each node, the search continues following a branch depending on whether the point is greater or less than the node reference value for the partitioning axis. When the first leaf node is reached, it is marked as the closest point found up to that moment and its distance stored as a search range. Then the algorithm goes up the tree, unwinding the path followed. In each node it checks if the splitting hyperplane intersects an hypersphere of the radius of the search range centered in the current best. This means that there could exists another closer point in the section of the space separated by that hyperplane and that the other branch of the tree must be explored. If it does not cut the hypersphere, the branch is discarded and the algorithms keeps unwinding the path up the tree. This recursive process is outlined in algorithm B.3.

The nearest neighbor search with kd-trees can be performed in $O(\log n)$ time.

Appendix C

Experimental settings

As other robotics related research, the works presented in this PhD Thesis required the setting and maintenance along all the Thesis of a suitable experimental environment composed of several hardware and software platforms. Dealing with the technical problems implied by this maintenance, while they may not look relevant to the contributions to the state of the art, takes a great amount of time and effort that we think should be somehow represented in this report.

This Appendix will be devoted to describe those experimental settings. In the following section C.1, the different robotic platforms used will be described. Section C.2 contains a description of the image acquisition hardware that was mounted on the robots to acquire visual information. Section C.3 refers to the software tools and environments used. Finally, in section C.4 a description of the configurations of the previously described software and the programs developed for the experimental validation of the approaches presented in chapters 2 to 4 of this report is provided. The tables showing the specifications of the different hardware used can be found at the end of this appendix.

C.1 Robotic platforms

As the works of this PhD Thesis are devoted to the application of Computational Intelligence tools to mobile robotics, several robotic platforms have been used to test them and to carry out other auxiliary tasks like data recording. These robotic platforms are described below.

C.1.1 Pioneer robotic platform

The main robotic platform used in the works of this PhD Thesis is the Mobile Robots' Pioneer DX, in two of its versions. The Pioneer robots are a commercial platform widely known and used in the robotics research field. In general terms, the Pioneer is a wheeled tricycle robot, with two motorised wheels, which provide drive and steering, and a rear caster. It is a medium sized robot designed following the regulations of the mid-size RoboCup league. This makes its dimensions (figure C.1) and characteristics very suitable for indoor environments like laboratories or offices.

Early experiments were performed with the older model, a Pioneer 2 DXE, while late experiments made use of the Pioneer 3 DX. Both robots have similar physical characteristics and capabilities, being the main differences related with the on-board electronics, like the microcontroller or the embedded PC.

As described by the manufacturer, the Pioneer is an agile, versatile intelligent mobile robotic platform. Late versions have been updated from previous versions to carry loads more robustly and to traverse small floor obstacles, such as door sills, more safely. The on-board power supply has been also upgraded with high-performance current management to provide power when it's needed. Its a robust platform, one step ahead of smaller and weaker hobby robots, designed to last for years of experimentation and built on a rugged aluminium body. This body has a back door which provides access to the batteries, allowing easy hot-swapping. The three batteries store up to 252 watt-hours of charge, providing up to 18-24 hours of operation of the bare platform with a full charge. Pioneer's nose is also removable, allowing access to the embedded computer, which can be expanded with up to 3 PC104+ cards. This embedded computer allows on-board Ethernet based communications and other autonomous functions like vision or laser processing with the adequate accessories.

On-board standard sensing is composed of 500 tick encoders in the two motors and a frontal ring of ultrasonic sensors providing 180^{o} forward coverage, with ranges from 15 cm. to approximately 5 m. Additional sensing capabilities can be improved with several options, like bumpers, compass, optical cameras or stereo and laser range finders.

Robot's drive is provided by the two 19 cm. pneumatic rubber wheels, powered by two separate motors. Those motors allows for top speeds of 1.6 meters per second and payloads of up to 23 Kg., including batteries

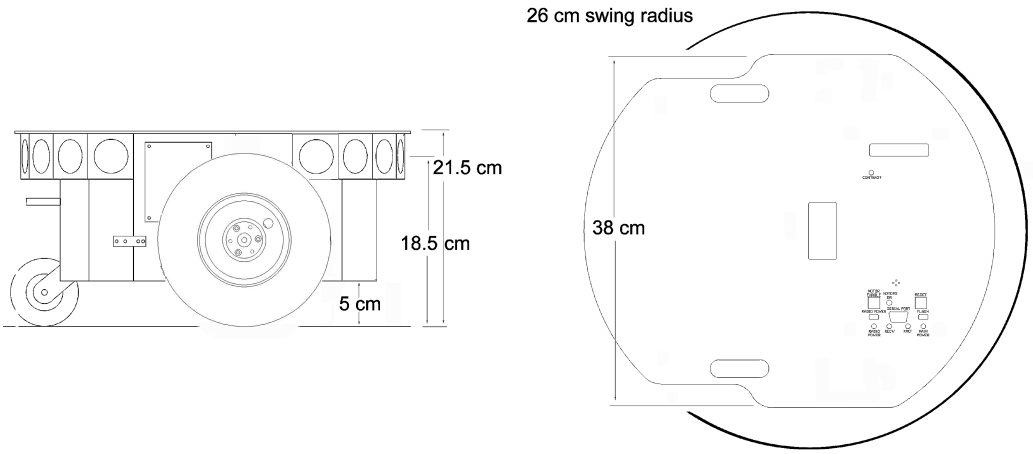


Figure C.1: Pioneer robotic platform dimension.

and accessories, and it can also climb 25% grades and sills of 2.5cm. This differential drive platform is highly holonomic and can rotate in place moving both wheels, or it can swing around a stationary wheel in a circle of 32cm radius. The rear caster balances the robot.

As every Mobile Robots Inc. platforms, Pioneer robots use a client-server architecture for control in which a software running in an embedded microcontroller controls directly the actuators of the robot, providing also an interface through RS-232 serial connection to other client applications for higher-level control.

The two robots used in the works of this PhD Thesis, being from two different generation Pioneer robots, had some differences in their configuration and accessories, which will be specified next. Full comparative specifications are shown in tables C.2 and C.3.

Pioneer 2 DXE

The first Pioneer robot used was a second generation Pioneer 2 DXE, with the following configuration:

- On-board computer based on a AMD K6-II 400 MHz processor with 128 MB of RAM, running Microsoft Windows Me operative system. This on-board computer is expanded with two PC104+ cards:

  - 802.11b compliant Orinoco Wi-Fi card for wireless communication.
  - Imagenation PXC-200A frame grabber for image acquisition from the pan-tilt-zoom (PTZ) camera.

- Sensing capabilities:
  - Standard frontal ultrasonic sensor ring.
  - Rear ultrasonic ring.
  - Frontal and rear bumpers.
  - Sony EVI-D30 PTZ camera connected to the on-board computer.

- The embedded control is provided by an 20 MHz Siemens 80C166 16 bit microcontroller, running the P2OS version of the server software.

- External ports and indicators are distributed between the sides of the body and the top roof.
  - On-board computer ports and switches are located in the right side panel.
  - Microcontroller ports and switches are located in the top frontal part of the robots roof, to the left of the PTZ unit.
  - Status LCD display is located to the left of the PTZ unit.
  - Power switch is located on the left side panel.

In figure C.2 the used Pioneer 2 DXE robot is shown.

Pioneer 3 DX

The second Pioneer robot used was a third generation Pioneer 3 DX, with the following configuration:

- On-board computer based on a Intel Pentium III 800 MHz processor with 256 MB of RAM, running Microsoft Windows 2000 operative system. This on-board computer is expanded with two PC104+ cards:
  - 802.11b compliant Orinoco Wi-Fi card for wireless communication.

Figure C.2: Pioneer 2 DXE robot with the Sony EVI-D30 mounted.

  - Imagenation PXC-200A frame grabber for image acquisition from the PTZ camera.

- Sensing capabilities:

  - Standard frontal ultrasonic sensor ring.
  - Canon VC-C4 PTZ camera connected to the on-board computer.

- The embedded control is provided by an 33 MHz RISC-based Hitachi H8S 32 bit microcontroller, running the ARCOS version of the server software.

- External ports and indicators are distributed between both the sides of the body:

  - On the right side: on-board computer ports and switches.
  - On the left side: microcontroller ports and switches, plus LED status indicators and power switch.

In figure C.3 the used Pioneer 3 DX robot is shown.

Figure C.3: Pioneer 3 DX robot

Control architecture

Pioneer robots use a client-server architecture for control. In this architecture the robot acts as the server, handling the low-level control of mobile robotics, like maintaining the platform's drive speed and heading over uneven terrain, acquiring sensor readings, such as the sonar, and managing attached accessories. The direct control of the robot's actuators and sensors is managed by an embedded microcontroller running the server software.

The client-server architecture is completed with client software applications running higher-level control and connected to the embedded microcontroller through RS-232 serial connection. Figure C.4 shows the diagram of the architecture. This client software can be running in an on-board or external computer which can be connected to the server directly through the serial communications or other bridge accessories, allowing several control configurations, as shown in figure C.5.

This client-server environment eases the development of high-level control programs, providing an abstraction layer between the physical robot and the control software, allowing the same client applications to be ported between

Pioneer platforms without modifications.

Besides this client-server control, the robot also allows for microcontroller-level robot programming, loading the developed low-level control software in the microcontroller's flash memory.

C.1.2 SR1 Robot

The SR1 Robot (figure C.6) is a small sized wheeled robot, designed mainly for educational purposes. This makes the SR1 robot a very versatile platform, as it must offer a configuration complete enough to address the typical problems that any student interested in mobile robotics has to know. Therefore, despite its small size, the robot has a very complete range of sensors (contact switches, IR sensors, photoelectric cells, compass) which can be expanded thanks to the I2C interface that it implements.

On-board processing is provided by a Netmedia BasicX-24P microcontroller and the possibility of off-board computing is provided through wireless communications by means of a ER-400TRS RF transceptor. Also a DB-9 serial port for RS-232 communications is provided in the main board, allowing also for connection of other communication devices for Bluetooth or WiFi RS-232 tunneling, with appropriate devices.

The PVC chassis also allows it to be expanded with other complements like tracks for off-terrain drive or a pan-tilt unit for a camera and ultrasonic sensor.

The drive is provided by two radio-control HS-422 servos which propel each rear wheel independently. Frontal part of the robot is balanced by an unpowered wheel. The robot is powered by 6 AA batteries, which provide around 20 hours of operation without the RF module.

Overall, the SR1 robot turns out to be a suitable low-priced platform for prototype building. Some of its specifications are shown in table C.4.

C.2 Image acquisition hardware

In this section we review the specifications of the image acquisition hardware used in the experimental works reported in this dissertation. The two main sources of visual information have been the PTZ cameras mounted on the Pioneer robots and the 3D ToF camera, which we have also mounted temporarily on the Pioneer robot.

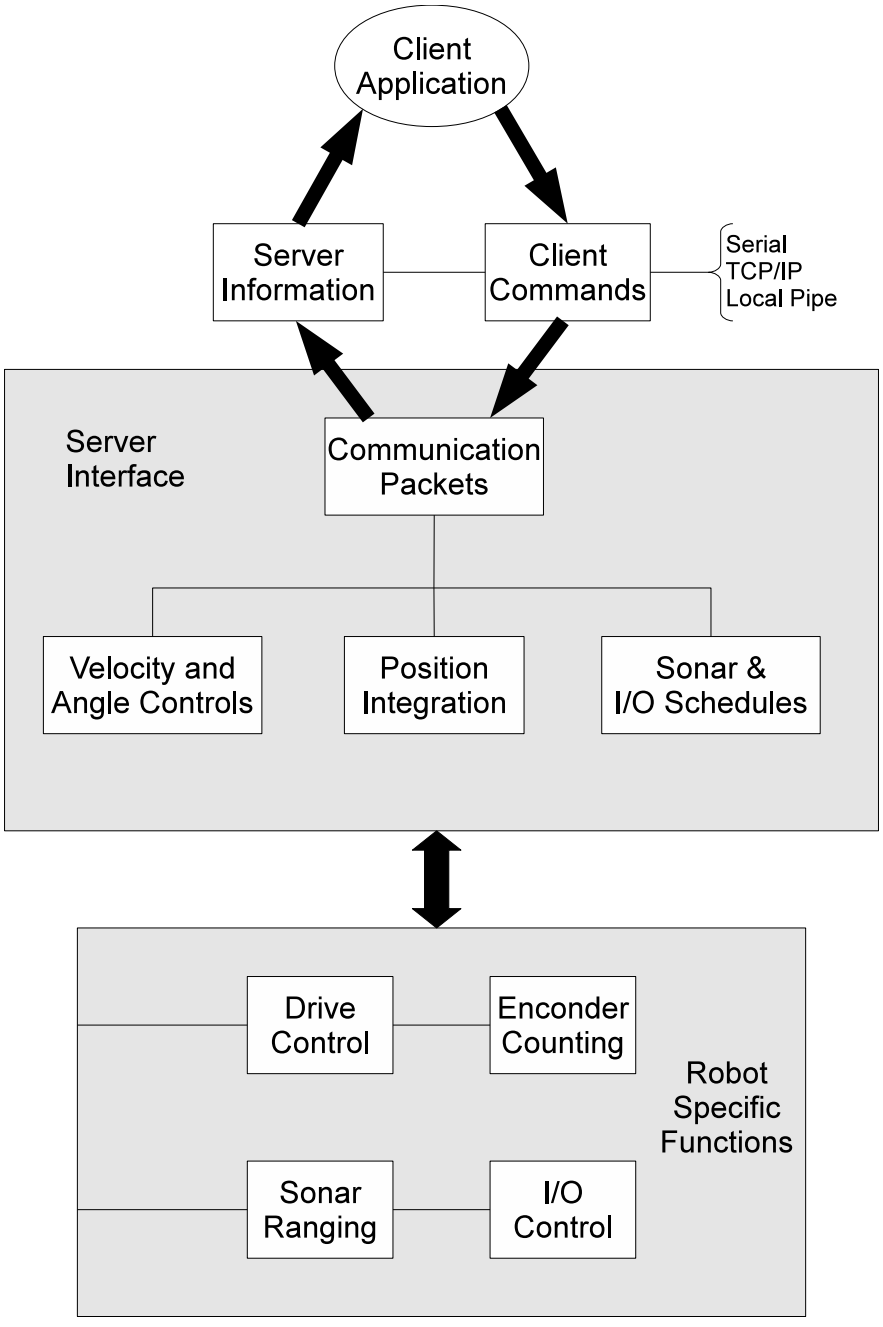


Figure C.4: Pioneer client-server control architecture.

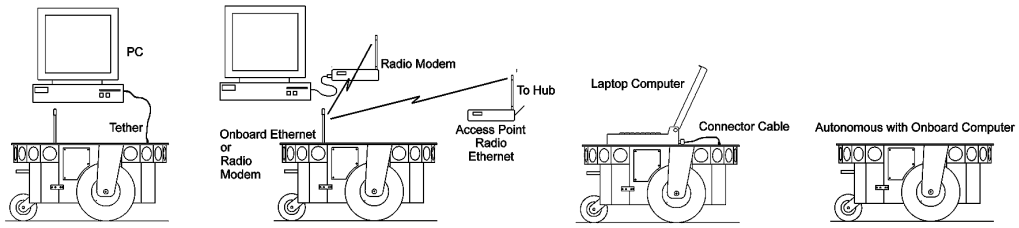


Figure C.5: Client server connection options.

Figure C.6: SR1 robot.

C.2.1 Sony EVI-D30 PTZ camera

The Pioneer 2 DXE had a Sony EVI-D30 pan-tilt-zoom camera mounted over the flat roof. The camera's optical unit is mounted on a pan-tilt platform that allows it to be oriented through command controls from the robot. Those command controls are sent to the camera through a proprietary VISCA RS-232 connector. Although being a digital camera with a CCD sensor, it only provides analog output through a composite RCA video connection. Thus, a frame grabber is required to capture the images provided by the camera with a computer. Specs of the EVI-D30 camera are shown in table C.5.

C.2.2 Canon VC-C4 PTZ camera

The Pioneer 3 DX came with a lighter PTZ camera, a Canon VC-C4. Mounting of the camera on the robot was in the same position on the flat roof. Connection to the robot's on-board computer was also a composite RCA video cable connected to the frame grabber, with control commands sent from the robot by a VISCA RS-232 connection. Full specifications of the VC-C4 camera are shown in table C.6.

C.2.3 Imagenation PXC-200A frame grabber

Both Pioneer robots had their PTZ cameras connected to the on-board computer by means of a Imagenation PXC-200A PCI frame grabber. This frame grabber is a commercial model oriented to industrial applications, using the industry standard PCI PC104+ socket to connect to the computer's motherboard. The cameras were connected with RCA connectors, providing images with the NTSC video format. PXC-200A specifications are shown in table C.7.

C.2.4 Mesa Imaging Swissranger 3000 ToF 3D camera

Swissranger 3000 camera (http://www.mesa-imaging.ch/) (figure C.7) is a 3D Time-of-Flight digital camera based in the principle of phase measuring to estimate distances. The camera uses an array of LEDs which emits light in the close infra-red spectrum to illuminate the scene. Knowing the wavelength of the light emitted, when a pixel of the sensor detects some light rebounded from an object in the scene, the distance traveled by this ray of light since

it was emitted by the LED array can be estimated by measuring its phase (figure C.8). This distance is calculated following the algorithm known as the four buckets, or four steps method, computed according to the equations:

$$D = L \cdot \frac{\varphi}{2 \cdot \pi}, \tag{C.1}$$

$$L = \frac{c}{2 \cdot f_m}, \tag{C.2}$$

where φ is the measured phase, L represents the non-ambiguity distance range, c the speed of light and f_m the RF modulation frequency [79]. Due to the periodicity of the phase of the infra-red light wave, distances can only be measured with precision inside the non-ambiguity range defined by L, since the same phase can be measured in different periods of the wave. The specifications of the camera state that within the non-ambiguity range, distances are measured with a typical precision of 1% of the measured distance (table C.1).

Compared to other 3D imaging technologies, such as laser triangulation or stereoscopic vision systems, ToF based measurements have several important advantages :

- Measurements are not dependant on external reference points or presence of contrast in the surface of measured objects.
- Real time, video frame rate measurements with no dependency on scanning cycles or limitations imposed by computer processing power.
- Scene illumination is provided by the camera, and it is not dependent on illumination from external, visible light.
- Camera directly delivers depth values for each pixel, without the need of complex calculation algorithms, resulting in faster, less computation intensive systems.
- Larger objects can be measured easily without fitting any additional system parameter, such as the camera base length in stereo vision.

As the optical characteristics of the camera are known, each pixel has a correspondence to fixed azimuth and zenith angles in spherical coordinates. Using the measured distance for each pixel, the spatial location with respect to the camera of the object which reflected the detected light can be obtained.

Figure C.7: Swissranger 3000 3D ToF digital camera.

Operating Range [meters]	**0.3**	**1**	**2**	**3**
X-Y Resolution (one pixel) [mm]	1.5	5.0	10.0	15.0
Distance Resolution [mm]	2.5	6	13	22

Table C.1: Measurement accuracy of the Swissranger 3000 camera (central pixel).

Raw data provided by the Swissranger 3000 camera consists of two images: one distance image and one intensity image. In the distance image the pixel value corresponds with the measured distance in that orientation in spatial coordinates. The pixel value in the intensity image corresponds with the amount of reflected light that the sensor's pixel detected.

Full specs of the Swissranger 3000 3D camera are shown in table C.8.

C.3 Software

In this section we review the characteristics of the main software tools used for the development of the PhD Thesis works. We review the ARIA libraries provided for interfacing the robot control and the client applications, the Basic Express environment for SR1 software development, the image capture libraries of the frame grabber provided with the Pioneer robot versions, the

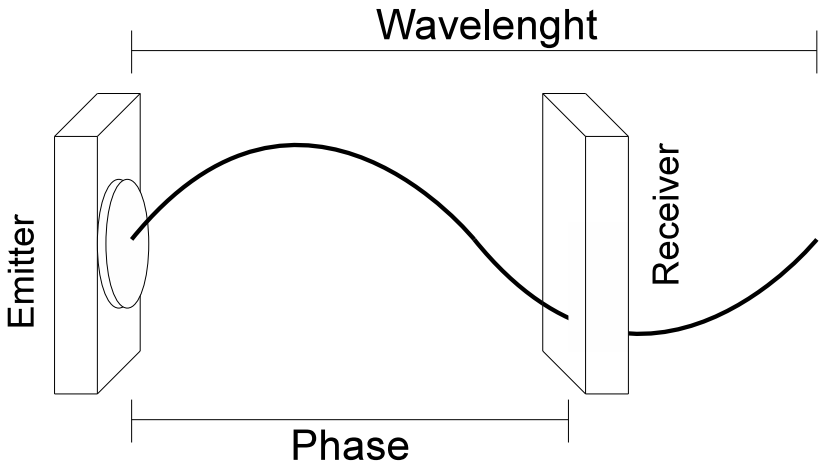


Figure C.8: Phase measuring distance estimation principle.

OpenCV libraries used for some image processing applications and the two main programing environments used: Microsoft Visual C++ and MATLAB.

C.3.1 ARIA libraries

Advanced Robot Interface for Applications (ARIA) is an object oriented C++ SDK library provided by Mobile Robots for programming all of its robotic platform series. ARIA offers high-level functions for building a client-side control software, providing dynamic control to all of the robot's parameters, like its velocity or heading, and allowing also reading from robot sensors like its odometry, sonar, or any other operating data provided by the robot platform. Besides the high-level Actions infrastructure, ARIA also allows low-level command communication with the robot.

ARIA provides an abstraction layer to the robot's main functions through the high-level class ArRobot (figure C.9). This class offers client side access to robots operative system's functions, as well as other components like sensors of the PTZ cameras.

Connections with the robot is managed through the ArDeviceConnection. Recent versions of ARIA also includes a library called ArNetworking which implements an extensible infrastructure for easy remote network operations, user interfaces, and other networked services. Through a server executing on the robot's PC, ArNetworking-enabled clients connect from another com-

puter on the network to get data and issue commands.

A variety of other useful tools for building robot applications are included in ARIA or available as separate libraries, including speech synthesis and recognition; sound effect playback; mathematical functions, cross-platform (Windows/Linux) thread and socket implementations; and more.

C++ development with ARIA is supported on GNU/Linux with GCC (3.4 or greater; ARIA is precompiled with GCC 3.4 but may be recompiled with any later version) and on Windows with MS Visual C++ .NET 2003 (7.1) and Visual C++ 2008 (9.0). As the robots used in this PhD Thesis works were Windows based, the programming environment used with ARIA was Microsoft Visual C++.

ARIA comes with full source code under the GNU General Public License. The license allows re-distribution of code as long as all is distributed freely. Proprietary distributions (without releasing your own source code, for example) requires a different, commercial license. ARIA includes a full API reference manual and example code.

Along the time this PhD Thesis works were done, ARIA libraries have gone through several revisions and updates, from 1.1 version to the last 2.5.1 used one (at the time of writing this report, last available version is 2.7.1). The two main versions were used according to their availability and capabilities:

- Version 1.x: This version was used for the tasks carried out with the Pioneer 2 DXE.
- Version 2.x: Version used with the Pioneer 3 DX. It introduced the new library ArNetworking, which allowed the building of the necessary client-server control program that was required to acquire data with the Swissranger SR3000 camera, as is explained in section C.4.2.

Using ARIA, a custom library was built with the standard actions we wanted to perform with the robot, like robot connection, standard movements or camera control. This custom library was used along all the PhD Thesis.

C.3.2 Basic Express BX-24

The Basic Express is a programming environment provided by Netmedia for the development of software for their BX series of microcontrollers. The BasicX language is based on Visual Basic, with a very similar syntax. The

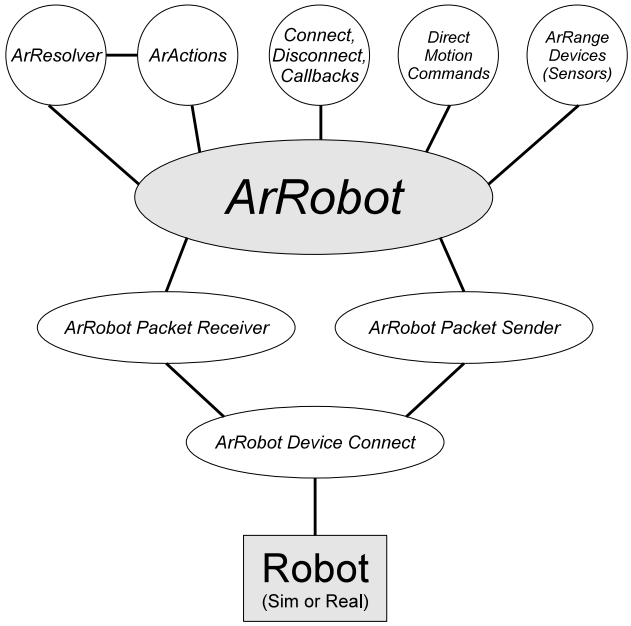


Figure C.9: ARIA library's structure diagram.

programs developed has to be compiled and downloaded to the SR1 robot through RS-232 serial connection.

Several examples of programming for the SR1 robot where provided by the manufacturer, which where used as the base for our own control functions and communications interface through RF communications.

C.3.3 Imagenation PXC-200 libraries

The PXC-200 frame grabber version used in these PhD Thesis works came without VfW (Video for Windows) compliant Windows drivers, so our programming environment (Microsoft Visual C++) could not made use of this multimedia framework to access the frame grabber device through DirectShow. Instead, the PXC-200A frame grabber came with an interface library to develop applications which communicate with it, available to DOS and Windows platforms through C libraries (DOS based programs) and DLLs (Windows based programs). This interface is implemented as a set of three libraries, of which only two were used, as the third was a DOS-only library, not needed in the working environment. Those libraries and their purpose are:

- **PXC200A Frame Grabber Library:** Includes the functions used to control the frame grabber, including capturing images, setting image resolution, switching video inputs, and setting image contrast, brightness, hue, and saturation.

- **Frame Library:** Includes the functions used to access captured image data and to read and write image files. In later developments, this library was substituted in the applications developed by the more general, computer vision oriented OpenCV libraries.

As was done with the ARIA libraries, PXC-200 libraries were used to build up a custom library with the most used functions in our applications.

C.3.4 OpenCV libraries

OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision. It is free for commercial and research use under the open source BSD license. The library is cross-platform, and runs on Windows, Mac OS X, Linux, PSP, VCRT (Real-Time OS on Smart camera) and other embedded devices. OpenCV was originally developed by Intel. It focuses mainly on real-time image processing, and as such, if it finds Intel's Integrated Performance Primitives (IPP) on the system, it will use these commercial optimized routines to accelerate itself. Earlier versions of this library had some problems running on non-Intel platforms as they made intensive use of the MMX and SSE primitives introduced in the Pentium processor, preventing us from using them with the AMD based on-board computer of the Pioneer 2 DXE as that generation's AMD processors did not include those extensions.

OpenCV is a well-known, widespread computer vision library used in many researches in this field. It has been used in some notable applications, like the vision system of Stanley, Stanford's 2005 DARPA Grand Challenge race winner, or Google's Street View.

The OpenCV library is composed of five modules:

- **cxcore:** Basic data structures and linear algebra support.

- **cv:** Main image processing and vision OpenCV functions.

- **cvaux:** Auxiliary, experimental and defunct OpenCV functions.

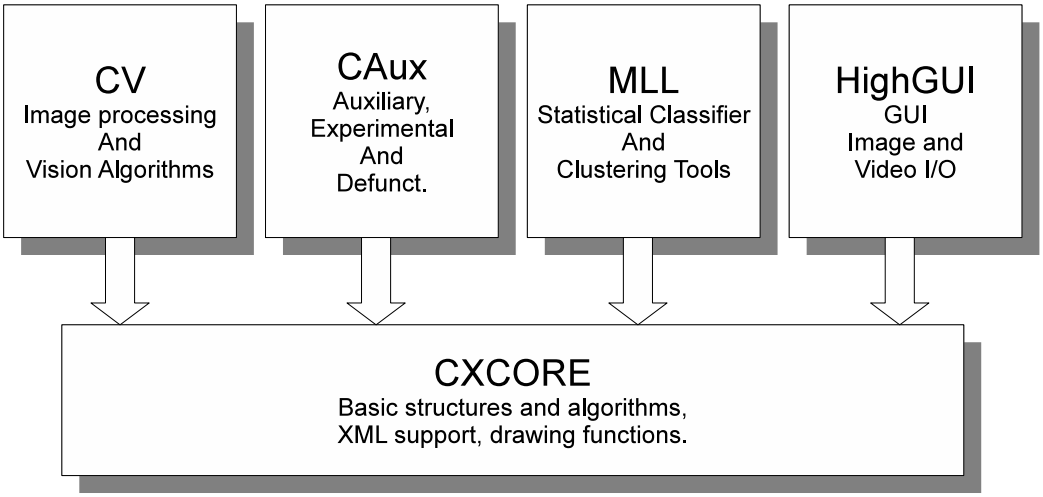


Figure C.10: OpenCV basic structure.

- **mll**: Statistical classifier and clustering machine learning OpenCV tools.
- **highgui:** GUI and I/O functions.

C.3.5 Microsoft Visual C++

Microsoft Visual C++ (MSVC) is a commercial integrated development environment (IDE) product engineered by Microsoft for the C, C++, and C++/CLI programming languages. It has tools for developing and debugging C++ code, especially code written for the Microsoft Windows API, the DirectX API, and the Microsoft .NET Framework. This development platform was chosen since it was the recommended and supported platform by the ARIA libraries for code development in the Windows platform. Initially Visual C++ 6.0 was used, being later changed to Visual C++ .NET 2003, as the ARIA 2.x libraries required this new version.

C.3.6 The MathWorks MATLAB

MATLAB (abbreviation for MATrix LABoratory) is a numerical computing environment which offers an IDE with its own programming language (M language) for mathematical applications programming. Its a cross-platform

software with versions for Unix, Windows and MacOS X operative systems, with a Java-based GUI.

MATLAB allows matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, interfacing with programs in other languages and exporting developed code to other languages or to standalone executable applications. It provides also GUI development tools through GUIDE package and graphical multidomain simulation and Model-Based Design for dynamic and embedded systems through Simulink. Both MATLAB and Simulink can be extended by means of optional toolboxes (MATLAB) and blocksets (Simulink).

MATLAB is a widespread software, extensively used in research, both in the university and I+D Centers.

C.4 Experimental configurations

In this section we review the detailed experimental configurations of the robots for several of the experiments reported in this PhD dissertation, those related to some of the experiments done with the Lattice Associative Memories (LAM), related to the 3D ToF camera and those related with the Linked System.

C.4.1 Lattice Computing approaches to localization and mapping

In the works presented in chapter 2 two different experimental approaches where realized. The main interest of the experiments reported in section 2.3 was to try the presented approach in a real robotic mobile platform. On the other hand, in following works, reported in sections 2.4 and 2.5, the emphasis was put in testing the suitability of the proposed approach for view characterization and its performance in recognizing landmarks. Thus, an off-line experimentation approach with pre-recorded datasets was preferred, as this avoids most of the misleading and time consuming technical problems which a real robot implementation entails.

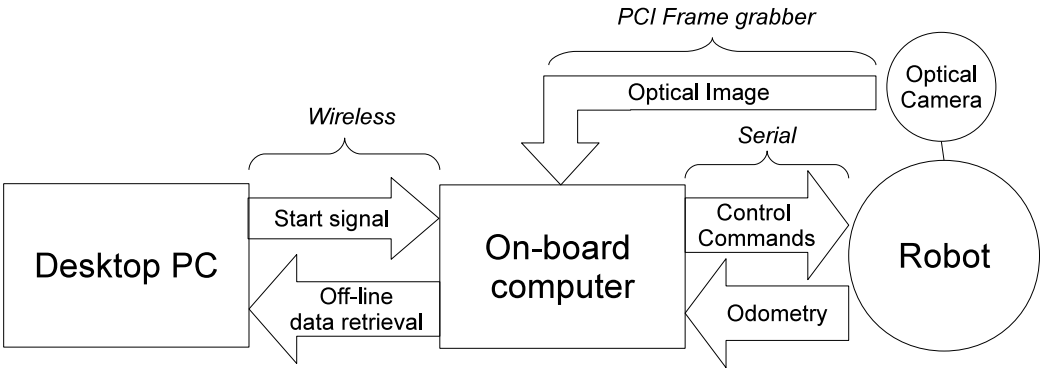


Figure C.11: Flow chart for the control program and data retrieving.

On-board experimentation

In section 2.3 a two step process was done. In the first step, the robot had to make a walk, following the path which was desired to map. The data recorded in this first walk was used to build up a map on-line. In the second step, the robot had to make several walks, following the same path as in the first walk. Then it made use of the map generated in the first step and tried to recognize the positions it was going through. This two step process required two different control programs for the two tasks: one map building program and one localization program. Both programs are programmed in C++ and made use of the ARIA libraries to control and communicate with the robot and of the PXC200 libraries for image acquisition. Both programs also run in the on-board computer in an autonomous way. A remote desktop PC was used to start the robot's program remotely through a Wi-Fi wireless connection and to retrieve the recorded experimental data for further off-line analysis. In figure C.11 the operation diagram of this process is shown. Some detail about the specifics of the two programs is presented next.

Map building The map building program had a single thread, in which the main loop controlled the movements of the robot and the image and data acquisition and recording. The path to follow was pre-programmed beforehand. Also, to avoid problems with possible blurry images caused by the robots movement and to ensure an steady separation between frames, the path was followed in a movement-stop-capture loop with 600 mm. steps. Data acquisition of each of the positions followed the steps detailed in section

2.3.1:

1. Acquire the image from the frame grabber and the position from the odometry.
2. Compute the binarization thresholds.
3. Binarize the image with the two thresholds.
4. Compute the M and W memories.
5. Store the acquired image, M and W memories and related odometry position.

Localization Localization was performed by a double-threaded program. The main thread controlled the movement of the robot and it had the same pre-programmed path that the mapping program, but in this case the path was followed in a continuous way, without the movement-stop-capture loop. A background thread was continuously acquiring odometry readings and images and comparing them with the stored map. This localization process followed the steps described in section 2.3.1:

1. Acquire the image from the frame grabber and the position from the odometry.
2. Compute the binarization thresholds.
3. Binarize the image with the two thresholds.
4. For each of the stored positions:
   (a) If the luminosity of the image is very different from the stored one, skip the position.
   (b) Use the binarized images as inputs for the W and M memories.
   (c) If the output from both memories equals 1, mark position as recognized and store the recognition.
5. If no position has been recognized, store a -1, indicating failure in localization.

Experimental data analysis After the map building and localization processes, the recorder data was retrieved from a remote desktop PC in which several statistical analyses were made. Those results are presented in section 2.3.2.

Additionally the robustness of the localization was tested, measuring the area around the positions from which its related landmark view was recognized. This was done by means of a simple program that captured one single view, generating a map position with it, and then acquiring a new frame and trying localization against the single stored position when a controller ask for it. To measure the recognition area, a 1 m^2 graduated sheet was used (figure C.12). The procedure was as following:

1. The robot is placed in the center of the graduated sheet.
2. The reference map position is acquired.
3. The robot is manually moved to the front at steps of two centimeters until it reaches the border of the sheet, making one localization step in each position.
4. The robot is placed again in the center of the sheet.
5. Repeat steps 3 and 4 moving the robot backwards, to the left and to the right.

Recognition area was calculated using as references the farthest point in every direction in which the reference position was recognized. From those experiments it was verified that localization is more robust against displacements along the optical axis than lateral translations. Therefore the positive recognition area of a map's position has an elliptical form, whose maximum elongation is collinear with the optical axis. The radiuses of this ellipse will be very dependant of the characteristics of the environment, varying between 10 and 70 cm.

Off-line experimentation

To carry out the experiments described in sections 2.4 and 2.5, a suitable dataset was required, which had to be recorded prior to any test. Those image datasets were recorder over several predefined walks in different floors of the building in which the "Facultad de Informática de San Sebastián"

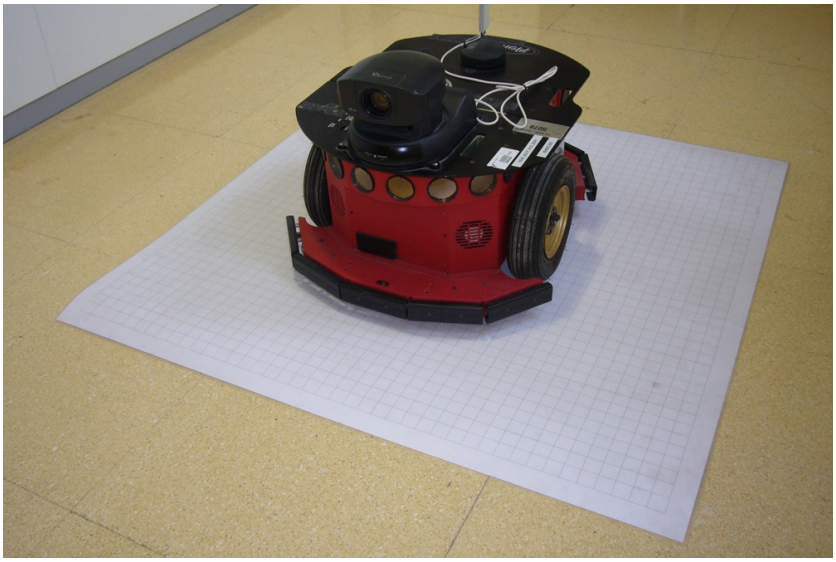


Figure C.12: Pioneer 2 DXE on the graduated sheet used to measure position recognition area.

resides. Those walks try to simulate possible paths that a robot should travel in one hypothetical navigation task in that building.

Data recording Dataset were recorded with the Pioneer 2 DXE robot and its mounted Sony EVI-D30 PTZ camera. The robot was driven manually at a constant speed with a keyboard connected to the PS2 port of the on-board computer. The robot acquired and stored one optical image from the camera each 5-6 centimeters, along with its related position as measured by odometry. The recorded data was retrieved off-line from a desktop computer via wireless connection. The control and image acquisition program was running on the on-board AMD K6-II computer and was programmed in C++ using the ARIA and PXC-200 libraries, as a client control software in the Pioneer client-server architecture (figure C.13). It had a two threaded structure, in which one of them runs a loop in the background, acquiring and storing optical images and odometry information at constant time steps, while the main thread waits for user's high-level control input through the keyboard, sends the low-level commands to the robot through serial connection and makes sure that the commands have been correctly executed by the robot.

The control of the PTZ camera was limited to check at the start of the

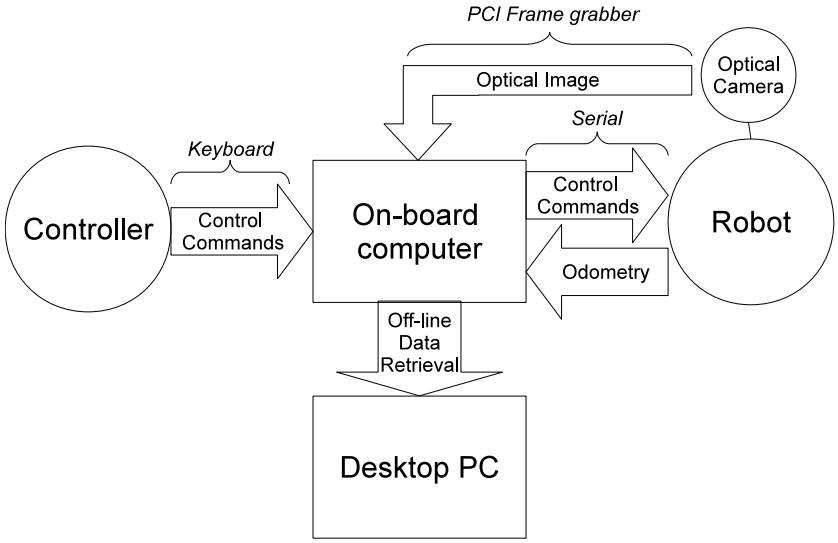


Figure C.13: Flow chart for the control and data acquisition program.

execution the centered position of the pan-tilt unit and the configuration of the zoom in the shortest focal length, in order to capture as wider field of view possible, oriented just ahead of the robot.

The resulting optical images datasets are described in appendix D.

C.4.2 Localization from 3D imaging

The approach presented in chapter 3 was also validated through off-line experimentation. This validation was done on pre-recorded 3D ToF camera data readings, compiled in several datasets described in appendix E. The settings used to acquire this datasets are detailed next.

Hardware configuration

Mounting the 3D camera on the robot presented some difficulties. It uses a USB 2.0 data connection for communication with the controlling computer. The Pioneer 3 robot used only had USB 1.1 data ports available, so the camera has to be connected to an external laptop in order to acquire data.

The ideal mounting of the camera on the robot, in the central-front part, was already occupied by the pan-tilt-zoom camera, which we did not want to unmount since complementary optical images were also desired. First

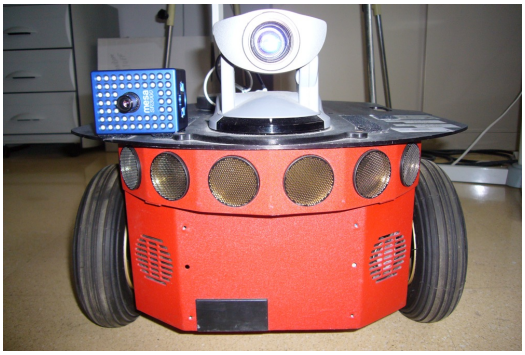


Figure C.14: 3D camera mounted on the Pioneer robot.

attempts were done mounting the 3D camera behind the pan-tilt unit, but the approach resulted unpractical with the available hardware. The resulting position was too high and suffered from great oscillations and vibrations. Placing the camera directly over the flat roof of the robot was the best solution to avoid those oscillations, but, as the pan-tilt unit occupies the central part of the robot, the 3D camera could not to be located directly on the longitudinal axis of the robot. Also, since the 3D camera has a cooling fan in the bottom, it could not be placed straight, as the airflow would be obstructed. Final position of the camera was placed, rotating it 90° clockwise, over the front part of the robot's roof and to the right of the longitudinal axis of the robot, as can be seen in figure C.14. This camera configuration has to be taken into account when processing data and analyzing the results.

The camera configuration had to be changed in order to adapt it to a moving platform. To avoid blurry data acquisition, exposition times were reduced to 1/200 s, in order to be fast enough to capture steady frames. This small exposition time also has other effects. One negative aspect is that it reduces the range of the camera, since farther rebounds have less intensity. On the other hand, it will ease the problem of rebounds beyond non-ambiguity range and specular reflections, since usually they are also of very low intensity and thus they will be less detectable with a shorter exposition time.

Robot control and data acquisition

In order to make the robot to follow an specific path along the environments that were going to be recorded, manual remote control was used. As the

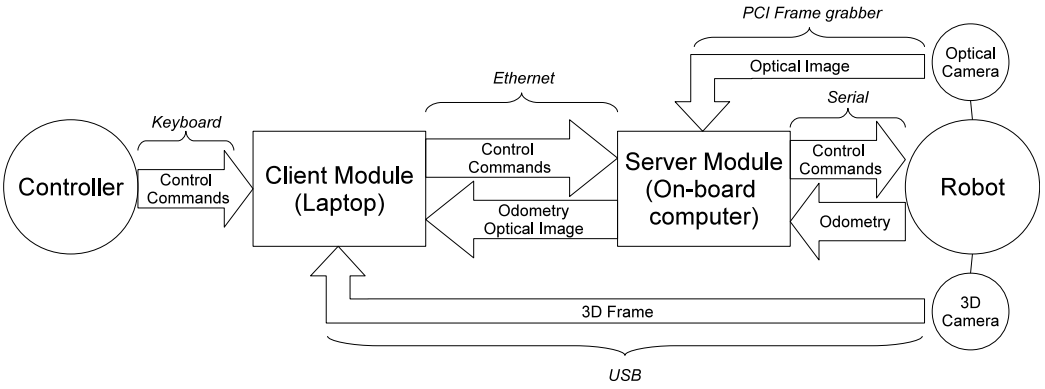


Figure C.15: Flow chart for the client-server control and data acquisition program.

3D data acquisition had to be done in an external computer because of the problems reported in C.4.2, the remote control was performed from a laptop carried by an operator, following closely the robot as the 3D camera has to be also connected to it.

To allow the remote control and data acquisition from an external laptop, a two module control-acquisition software was programmed (figure C.15), using ARIA client-server libraries provided by the robot's manufacturer:

- Client module, running in the controller's laptop, gets control commands from the operator and sends them to the robot through Ethernet. Also, when the operator hits 'space' key, the program will acquire a frame from the 3D camera and ask the robot for odometry data of the position and an image from the on-board optical camera.

- The server module runs in the robot's on-board computer waiting for commands from the client. It acts as an intermediary between the client module and the robot's hardware, sending to the robot's microcontroller the control commands received by the Ethernet and reading odometry. As the robot's on-board optical camera is connected directly to the on-board computer, the server module also manages the optical image acquisition from it. When asked for it, the server module will acquire and send via Ethernet odometry data and an optical image.

C.4.3 Multi-robot visual control

The works reported in chapter 4 are mainly experimental. However, as we were working on a new platform, the SR1 robot, first we had to learn it and build up the required hardware and software experimental environment. The experimental configurations used are reported below.

SR1 physical configuration

Being a education oriented robot, the SR1 is intended to show the students every step in the robotics research process starting from its physical construction. The SR1 robot comes, thus, unassembled. The first step in the configuration of the robotic hardware was to assemble it, including soldering the main board.

The characteristics of the environment in which the experiments where going to be performed required for some modifications in the SR1 robots. Its original yellow color is difficult to discriminate from the yellowish color of the floor's tiles, so there will be difficulties in their detection and localization by an artificial vision system. To avoid this problem and ease the segmentation process, the robots where painted in blue.

Robots also need a mechanism with which they can hook to he hose, while rotating freely below it. A practical and easy solution came with the use of cabinet casters. One of those casters screwed up on the robot's roof provide a rotating coupling for the hose. The wheel of the caster was used to avoid that the hose slips away, keeping it in place.

As most of the robot's main board is exposed, the hose had a tendency to get caught in the board's components, especially in the sonar unit. This was avoided by covering the main board with a cardboard with the corners folded to avoid as much as possible the existence of points to which the hose could get hooked.

The final robot configuration is show in figure C.16.

Command and communication protocols

A library of common command controls (advance, accelerate, brake, various turning functions, etc.) was programmed using the Basic Express environment. Communications where done through RF by means of ER-400TRS modules, connected to the serial port of the robot's microcontroller and with an USB adapter to the controller PC.

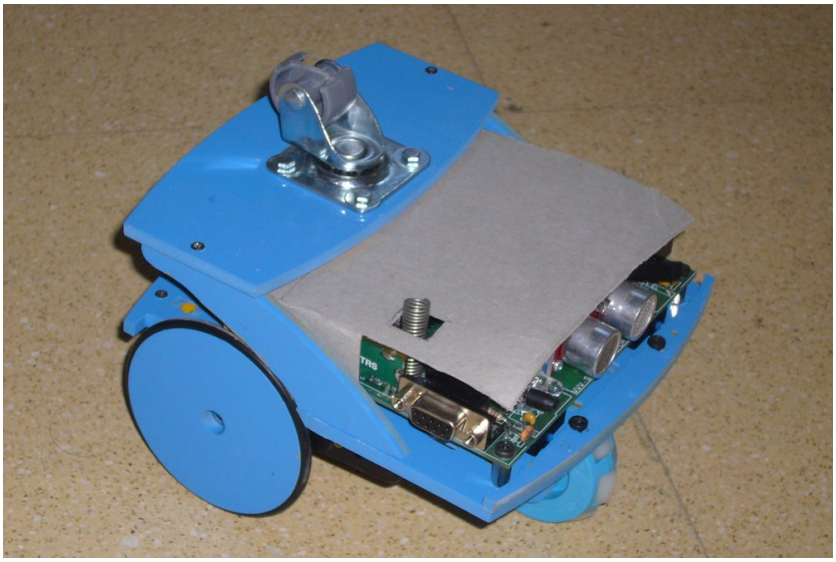


Figure C.16: SR1 robot's final configuration.

Initially, each robot used a different RF channel, with a simple command format. Early tests saw that this communication method was plagued with problems, as many command controls were lost and overlapping communication channels caused the robots to receive commands not sent to them. Also, the USB RF transceptor of the controller PC had to change constantly the channel to communicate with each and all the different robots, introducing delays that caused also synchronization problems. It was, thus, necessary to build a communications protocol robust enough, with all the robots using the same communication channel. After several try and error test, a compromise solution was found: it was estimated that merely repeating several times the desired command will be enough to cope with the problem of lost commands. To avoid the repeated command to be executed more than once, commands where labeled with an sequence ID. Following this protocol, each command packet (figure C.17) will have the following fields:

- Packet start: Packets starts with the character 'Z'.
- Robot ID: An {'1'-'9'} character identifying the robot the packet has been sent to.
- Sequence ID: A {'X','Y'} character identifying the command sequence. 'X' and 'Y' values alternate in consecutive commands.

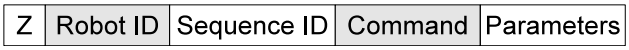


Figure C.17: Communication protocol's packet structure.

- Control command: 5 characters identifying the control command sent (e.g. 'MORAE' to command the robot to advance).
- Control command parameters: a variable number of optional parameters for some control commands.

When a robot receive a new packet (reads a 'Z' from the communication buffer), first checks if the robot ID character corresponds with its assigned number. If the message is for other robot, it discards the packet and waits for the next 'Z'. If the packet was destined to it, it checks if the sequence ID is the same than the last executed command. If it is the same, it discards the packet. If it is different, it reads the command and its parameters. If the command or the parameters are incorrect, the packet is also discarded. Finally, if the command is received correctly, the robot executes it and discards all the packets it receives until a packet with different sequence ID is read. This process is outlined in algorithm C.1.

As the packet size is quite small (8 characters for a normal command, up to 16 for the largest one), the transmission speed allowed for multiple command repetitions without introducing delays. In any case, experimental tests saw that a repetition of 3 was enough to avoid most problems from lost commands.

This protocol was implemented, besides the control program running in the robots, in a MATLAB-SR1 interface library for serial communications between the MATLAB environment and the robots. This library was used to communicate wirelessly the control program with the robots through the serial tunneling provided by the USB RF adaptor.

Robot control implementation

The control of the hose transportation system was programmed in the MATLAB environment. A GUI provided access to the configuration of a image acquisition device which provided global perception. Several parameters of the control heuristic could be changed on-line through the GUI, and the current state of the system was shown in a window (figure C.18). Commu-

Algorithm C.1 Robust communication protocol.

Given the robot with ID = # and expected command sequence ID = $.

1. Empty communication buffer until a 'Z' is received.
2. Read the robot ID field from the communication buffer.
3. If the robot ID is not #, go to 1.
4. Read the sequence ID field from the communication buffer.
5. If the sequence ID is not $, go to 1.
6. Read the command from the communication buffer.
7. Check the command. Go to 1 if the command is invalid.
8. If the command has parameters.
   (a) Read the command from the communication buffer.
   (b) Check the parameters. Go to 1 if they are invalid.
9. Execute the received command.
10. Change the expected command sequence ID.

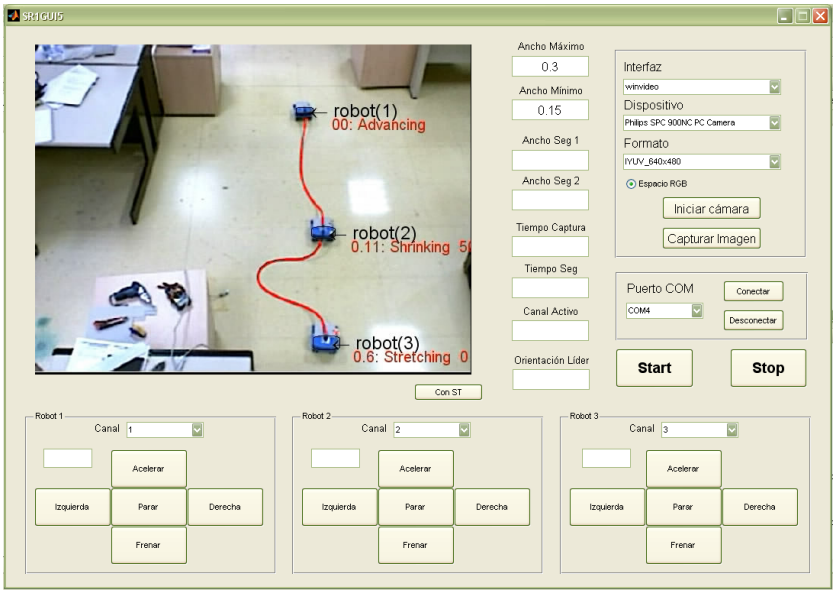


Figure C.18: GUI of the robot-hose transportation system's control program.

nications with the robots were performed with the MATLAB-SR1 interface mentioned before.

	Pioneer 2 DXE	**Pioneer 3 DX**
Physical Characteristics		
Length (cm)	44.5	44.5 (44)
Width (cm)	40	40.0 (38)
Height (cm)	24.5	24.5 (22)
Clearance (cm)	6.5	6.5 (6)
Weight (kg)	9	9
Payload (kg)	23	23 flat 14 @ 13% grade
Power		
Batteries 12VDC lead-acid	3	3
Charge (watt-hrs)	252	252
Run time (hrs)	8–10	18-24
with PC (hrs)	3-4	
Recharge time (hrs)		
Standard charger	6	12
High-Speed charger	2.4	2.4
Mobility		
Wheels	2 pneumatic 1 hard caster	2 pneumatic 1 hard caster
diam (mm)	191	191
width (mm)	50	50
Caster (mm)	75	
Steering	Differential	Differential
Gear ratio	19.7:1	38.3:1
Pushing force (kg)		6
Swing (cm)	32	32
Turn (cm)	0	0
Translate speed max (mm/sec)	1,800	1,200
Rotate speed max (deg/sec)	360	
Traversable step max (mm)	20	35
Traversable gap max (mm)	89	89
Traversable slope max (grade)	25%	25
Traversable terrains	Wheelchair accessible	Wheelchair accessible

Table C.2: Pioneer 2 DXE and Pioneer 3 DX comparative physical specifications.

	Pioneer 2 DXE	**Pioneer 3 DX**
Sensors		
Sonar Front Array	8 included 1 each side, 6 forward 20° intervals	8 included 1 each side, 6 forward 15° intervals
Rear Sonar Array	8 included 1 each side, 6 rear 20° intervals	8 optional 1 each side, 6 rear 15° intervals
Range		15cm - 5m
On-board computer		
CPU	AMD K6 II	Intel Pentium III
Frequency	400 MHz	800 MHz
RAM	128 Mb	256 Mb
SO	Microsoft Windows Me	Microsoft Windows 2000
Microcontroller		
Processor	Siemens 80C166 (20 MHz)	Hitachi H8S (33 MHz)
LCD	32 characters on 2 lines	n/a
Audio	Piezo buzzer	Piezo buzzer
Sonar inputs	2x8 (multiplexed)	16 max
Digital I/O	16 logic ports; 8 in, 8 out	8-bit I/O bus / 16 devices + PC104 I/O boards
A/D	5 @ 0-5 VDC	5 @ 0-5 VDC
FLASH PROM	32 KB	1Mb
Power switches	1 main; 1 RADIO	1 main; 2 auxiliary
Comm ports	2 RS-232 internal 1 RS-232 external	3 RS-232 serial ports

Table C.3: Pioneer 2 DXE and Pioneer 3 DX comparative sensor and electronics specifications.

Dimensions	
Length (cm)	15
Width (cm)	14
Height (cm)	10
Power	
Batteries	6 x 1.5V AA
Run Time (h)	20 without RF
Mobility	
Wheels	2 rear, 1 frontal
Diameter (cm)	7.2 rear 5 frontal
Servos	2 HS-422
Torque (kg/cm)	3.3
Sensors	1 x SRF08 ultrasonic sensor, front faced 2 x IR LED emitters, 1 receiver 2 x photoelectric cells 1 x CMPS03 digital compass 2 x contact switches, front faced 1 x inclinometer
Communications	1 x RS-232 DB-9 port 1 x ER-400TRS RF transceptor 1 x I2C communications port.
Microcontroller	Netmedia BasicX-24P

Table C.4: SR1 robot specifications.

Image Sensor	1/3" IT CCD
Effective Pixels	768(H) x 494(V) NTSC; 752(H) x 585(V) PAL
Horizontal Resolution	460 TV lines NTSC; 450 TV lines PAL
Vertical Resolution	350 TV lines NTSC; 400 TV lines PAL
Lens	12X Zoom, f=5.4 - 64.8 mm; F=1.8 - 2.7
Angle of View (H)	48.8 degrees (wide angle) - 4.3 degrees (telephoto)
Angle of View (V)	37.6 degrees (wide angle) - 3.3 degrees (telephoto)
Minimum illumination	7 lux / F1.8
Pan Angle	±100 degrees; max 80 degrees / second
Tilt Angle	±25 degrees; max 50 degrees / second
Weight	1,200 g

Table C.5: Sony EVI-D30 camera specifications.

Video Standard:	NTSC
Total Number of Pixels:	410,000 (380,000 effective pixels)
TV-Line:	460 TV L
Minimum Illumination:	6 lux (2 lux at gain-up mode)
SNR:	48 dB
Horiz, Field of View:	3 to 47.5 degrees 65 degrees with Wide angle lens adapter
White Balance:	Auto
Exposure:	Auto/Manual
Focus:	Auto/Manual
Focus Length:	4 to 64mm, F1.4 to 2.8
Zoom:	16x
Pan Angle:	±100 degrees (±170° VC-C4R)
Pan Speed:	1 to 90 degrees/sec (+10°, -90° VC-CR)
Tilt Angle:	+90/-30 degrees
Tilt Speed:	1 to 70 degrees/sec
Preset Position:	9 Positions
Controllable # by one IR:	9 Units
Cascade control:	9 Units
Control:	RS232 Serial (up to 19.2 kbps)
Power:	13V, 12W
Size (W x D x H) mm:	116 x 113 x 91
Weight:	375 g

Table C.6: Canon VC-C4 PTZ camera specifications.

Video Input Formats	NTSC, PAL, SECAM, S-Video
Video Input Signal	1 V peak to peak, 75 &
Resolution	NTSC: 640 x 480 PAL / SECAM: 768 x 576
Output Formats	Color: YCrCb 4:2:2; RGB 32, 24, 16 y 15 Grayscale: Y8
PCI PC-104 Card Dimensions	174.6 x 106.7 mm.

Table C.7: PXC-200A frame grabber specifications.

Pixel Array Size	176 x 144 (QCIF)
Field of View	47.5 x 39.6 degrees
Optical lens	f/1.4
Interface	USB 2.0
Illumination Power	1 Watt (average)
Wavelength	850nm
Housing Size	50 x 67 x 42.3 mm3
Housing Material	Aluminium
Power Supply	12V
Power Consumption	12 W, typical
Operating Temperature	-10^{o}C to +50^{o}C
Output Data (per pixel)	x, y, z coordinates i (intensity)
Camera Mounting Holes	2 x M4; 1 x $\frac{1}{4}$"
Range and Resolution	
Modulation Frequency	20MHz, standard
Non-ambiguous range	7.5 meters
Distance Resolution	1% of range, typical
Frame Rate	25 fps, typical

Table C.8: Swissranger 3000 specifications.

Appendix D

Optical image datasets

In this appendix several optical image datasets are presented. These image collections were recorded simulating several possible paths that a mobile robot should follow when performing a task inside our building. The experimental validation of the approaches reported in sections 2.4 and 2.5 of chapter 2 were done using these datasets. Detailed description of the configuration of the recording process is given in appendix C.

Due the size of these datasets we are unable to put them available online on our web site. Anyone interested in these datasets can request them through the contact address available in our wiki[1] and we will gladly send them by ordinary mail to the indicated postal address.

D.1 Procedure

Image datasets were recorder over several predefined walks in different floors of the building in which the "Facultad de Informática de San Sebastián" resides. Those walks try to simulate possible paths that a robot should travel in one hypothetical navigation task in that building.

Each walk was traveled six times, driving the Pioneer 2 robot manually through a connected keyboard at a constant speed. The robot acquired and stored one optical image from the camera each 5-6 centimeters, along with its related position as measured by odometry. Being guided manually, each one of the travels follows a slightly different path, having different oscillations in its movement and trajectory corrections as well.

[1] http://www.ehu.es/ccwintco

Original resolution	640x480
Final resolution	628x242
Format	8bit grayscale
File type	Bitmap (.bmp)

Table D.1: Format of the stored images.

D.2 Stored data format

D.2.1 Optical images

Raw NTSC frames acquired by the frame grabber have a resolution of 640x480 pixels. Those images are interleaved: each image is a composition of two frames, alternating rows from them. Frame interleaving involves a serious issue to vision based techniques when acquired from a mobile platform. When images are acquired in motion, scene captured by the camera changes between frames, and mixing different frames introduces artifacts which can be problematic for image processing. Thus, it is advisable to deinterlace the images. But deinterlacing is a difficult procedure and it does not provide a clean solution, introducing other artifacts like ghosting. As images are big enough, the easiest way to avoid problems produced by interlacing was by merely storing as an image the rows that come from the same frame. This resulted in full size images with the same resolution, but with its lower half rows without information (black). As the frame grabber driver provided this function, images were acquired in this format. Also, the NTSC format introduced black vertical rims which had to be cropped (figure D.1).

The resulting image after this process is a grayscale image of 628x242 pixels, stored in 8 bit grayscale BMP format (table D.1). As horizontal resolution more doubles vertical resolution after discarding half of the rows and the vertical rims, objects in the scene appear elongated (figure D.2). This can be an issue for some object recognition tasks, so a subsampling of the columns of the images may be required. However, the images were stored without this subsampling in order to save as much information as possible.

D.2.2 Odometry

Along with each of the travels, a text file is provided with odometry recordings corresponding with the points in which each of the stored images was

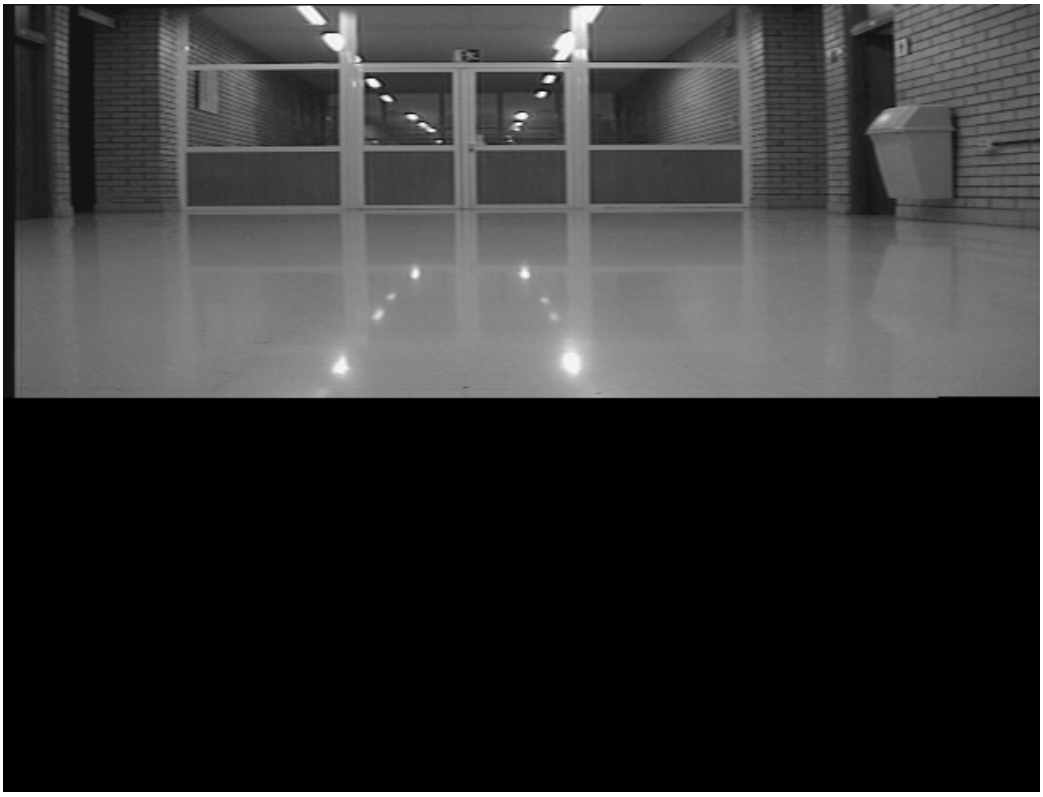


Figure D.1: Deinterlaced image as acquired.

Figure D.2: Cropped Image as stored.

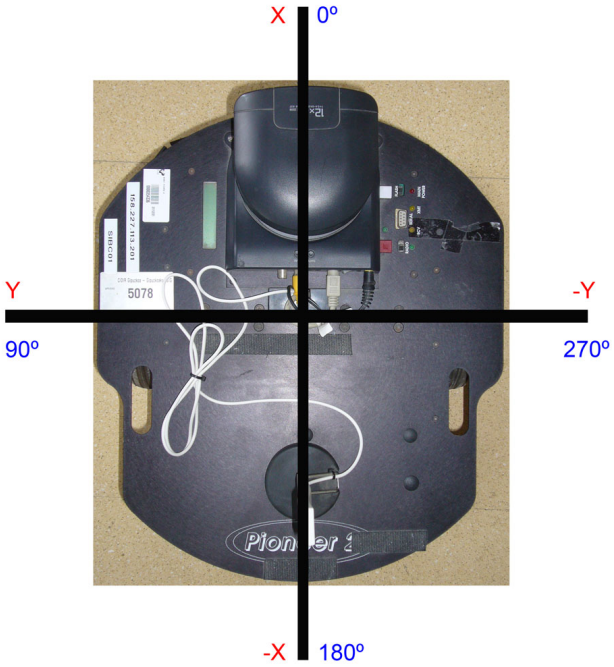


Figure D.3: Coordinate reference system in relation with original position of the robot.

taken. This text file stores one position each row, each position formed by four values separated by spaces, with the order <#image, x, y, θ>, being coordinates given in millimeters and orientation in degrees. The origin of the coordinates is in the starting position of the robot, being the X axis and the 0° orientation the direction in which the robot was facing (figure D.3).

Recorded odometry is quite imprecise, as can be seen on superposed paths show on the plans of section D.3. The greater errors relative to the nominal path of the robot are introduced at the turning points, while straight portions of the path are relatively well represented by the odometry readings.

D.3 Datasets

Five datasets are provided, recorded in two different floors. Each dataset contains six folders numbered from 01 to 06, one for each travel, and in each folder the files with the images are stored, plus a text file with the corresponding positions:

- File with the odometry readings: "posiciones.txt"
- File with the image corresponding to position number ###: "imagen_###.bmp"

D.3.1 Walk 01: Lab-Corridor-Hall

Short walk recorded in the third floor of the building. Starting from one of the laboratories, close to the starting point it turns to the right to cross a door. Then a corridor is followed up to a hall which is reached after another turn to the right. In the hall there are several furnishings (couches, cabinets, plant pots), which the path avoids to finally stop near the door to the stairs. Paths for the six walks are shown in figure D.4.

- Travelled distance: 35 m. approximately.
- $90^{\underline{o}}$ turnings: 2.
- Samples per travel: 750-800
- Illumination: Constant of artificial and natural source.

D.3.2 Walk 02: Lab-Corridor-Hall-Corridor-Hall

Long walk recorder in the third floor of the building. Starting from one of the laboratories, it turn right to go through a corridor and then again to the right to a hall. Another corridor is followed to the left up to another hall, when it turns again to the left to finally stop in front of a door. Paths for the six walks are shown in figure D.5.

- Travelled distance: 60 m. approximately.

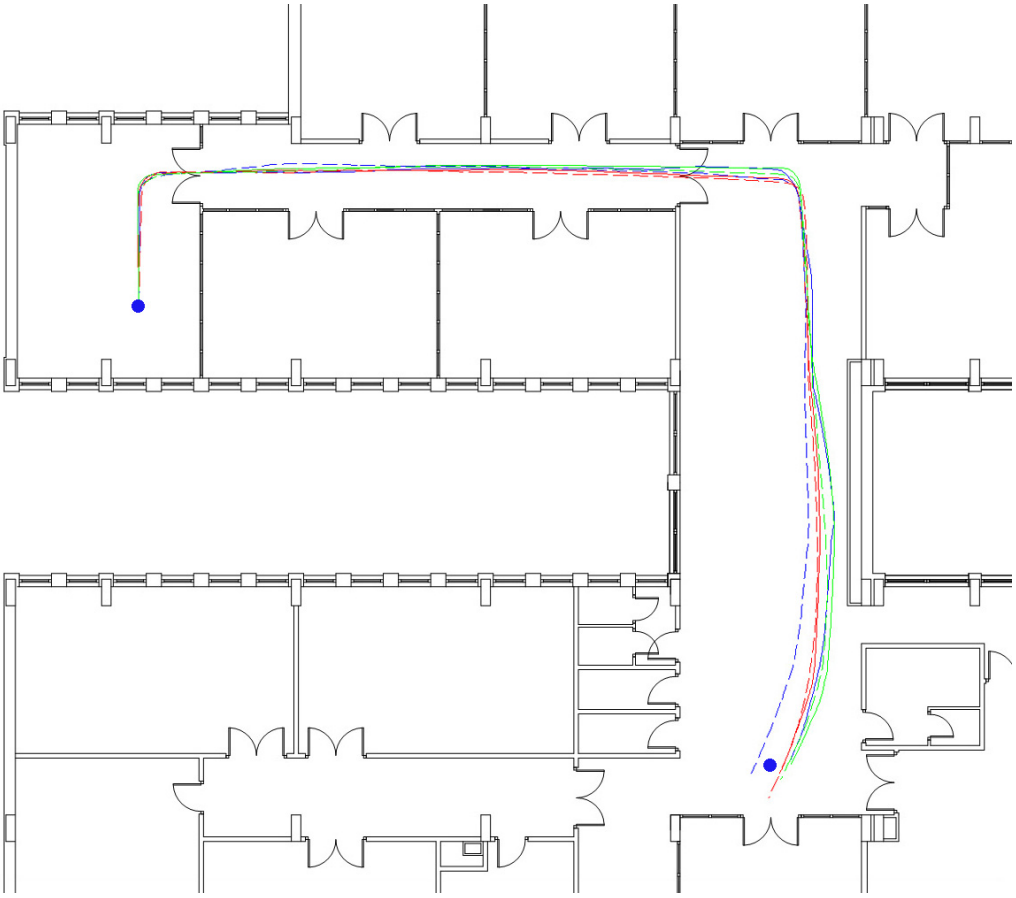


Figure D.4: Paths from the travels of walk 01.

- 90^{o} turnings: 4.
- Samples per travel: 1000-1050.
- Illumination: Constant of artificial source.

D.3.3 Walk 03: Hall-Corridor-Hall-Corridor-Lab

Same path than the walk 02, but in opposite direction. Paths for the six walks are shown in figure D.6.

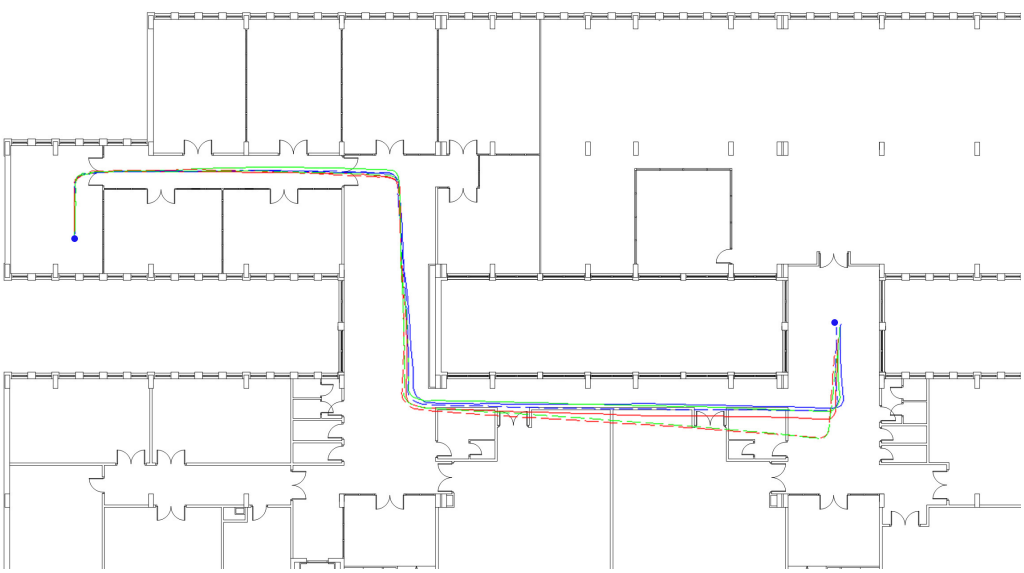


Figure D.5: Paths from the travels of walk 02.

- Traveled distance: 60 m. approximately.
- 90^{o} turnings: 4.
- Samples per travel: 1000-1050.
- Illumination: Constant of artificial source.

D.3.4 Walk 04: Hall 1

Long rectangular path in the building's first floor hall. Very regular structure, with several columns and stairs. Natural illumination with strong shadows and harsh changes between travels. Paths for the six walks are shown in figure D.7.

- Traveled distance: 40 m. approximately.
- 90^{o} turnings: 4.
- Samples per travel: 750-850.

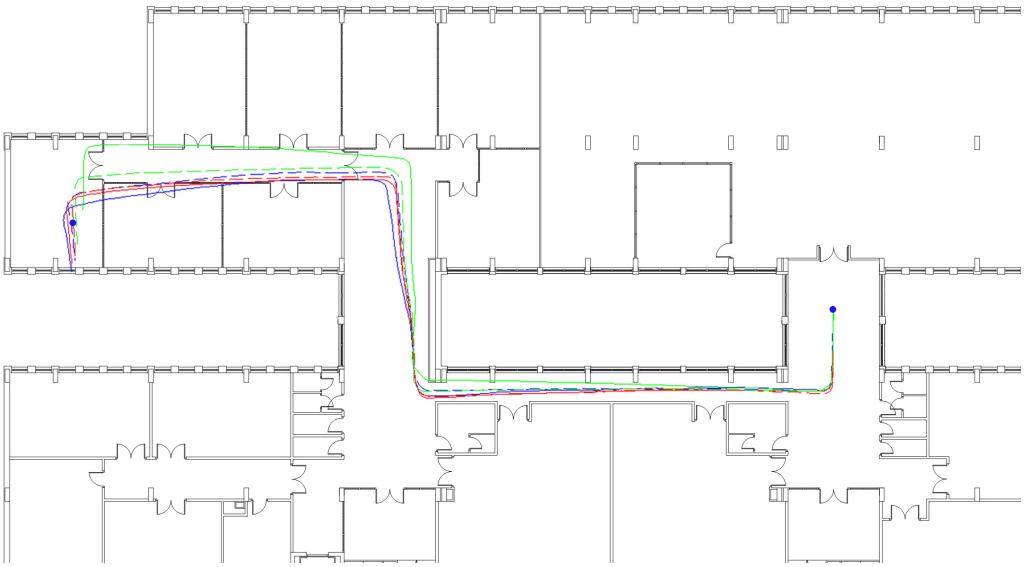


Figure D.6: Paths from the travels of walk 03.

- Illumination: Changing of natural source.

D.3.5 Walk 05: Hall 2

Rectangular path in the first floor's hall, in the opposite direction than walk 04. Similar illumination conditions. Paths for the six walks are shown in figure D.8.

- Traveled distance: 40 m. approximately.
- 90^{o} turnings: 4.
- Samples per travel: 650-700.
- Illumination: Changing of natural source.

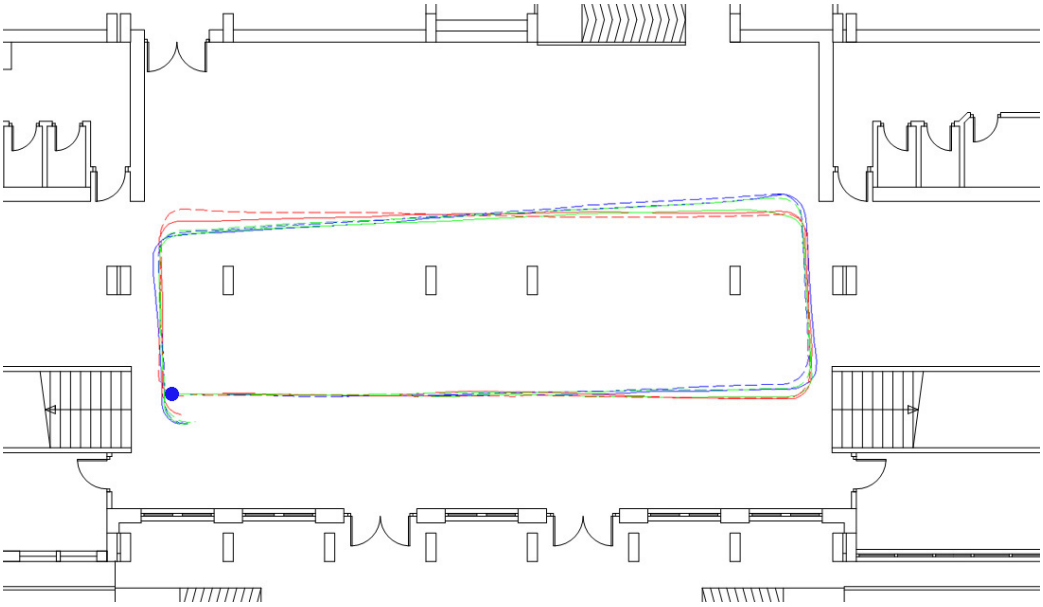


Figure D.7: Paths from the travels of walk 04.

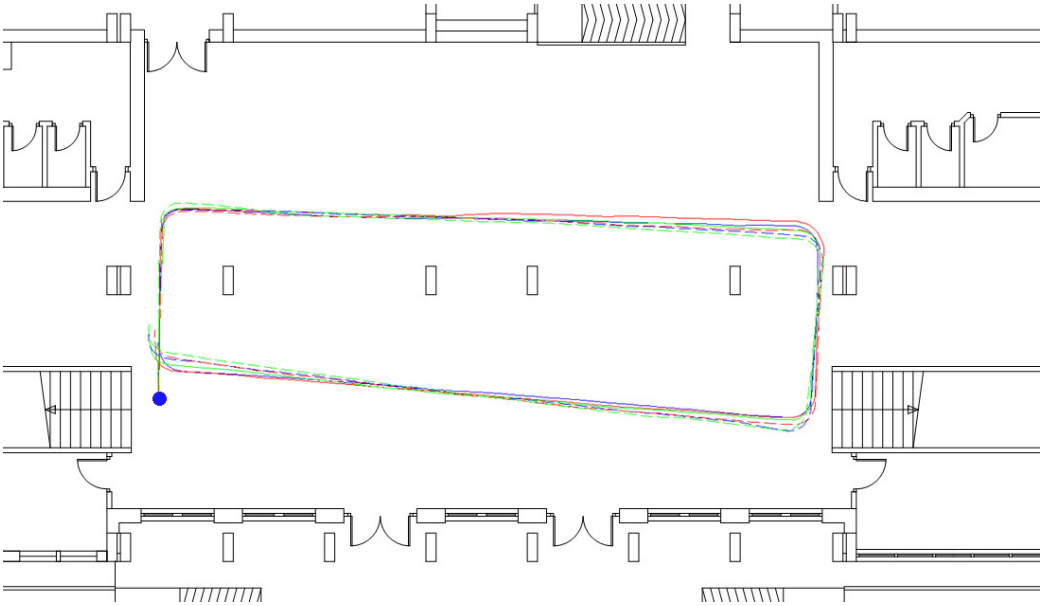


Figure D.8: Paths from the travels of walk 05.

Appendix E

3D ToF camera datasets

In this appendix we describe several datasets of measurements acquired with the Swissranger 3000 ToF camera mounted on top of a Pioneer robot. These datasets are the ones used for the off-line validation of the works presented in chapter 3. In this appendix only the specific details of the stored data and the environments they measure are described. A complete description of the hardware and software configurations used to record these datasets is given in appendix C. The datasets are freely available from the web site of the Computational Intelligence group:

http://www.ehu.es/ccwintco/index.php/Conjuntos_de_datos_3D

E.1 Environments

Datasets were recorded in rooms 125 and 126 of the first floor of the "Facultad de Informática de San Sebastián" (FISS) of the UPV-EHU, prior to reforms (figures E.1, E.2 and E.3). Both rooms are of rectangular plan, measuring 88 and 119 m^2, respectively. One of the walls in both rooms is composed of windows with radiators below (east wall in room 125 and west wall in room 126). There are projecting columns in east wall in room 125 and in both longitudinal walls in room 126.

In the north-western corner of room 125 there are piled several furnishings up to the middle of the room in wide and up to first column lengthwise. In room 126 there is also piled furnishings against the eastern wall between first and second columns, projecting up to 1.5 m from the wall.

In each of the environments three walks have been recorded, following

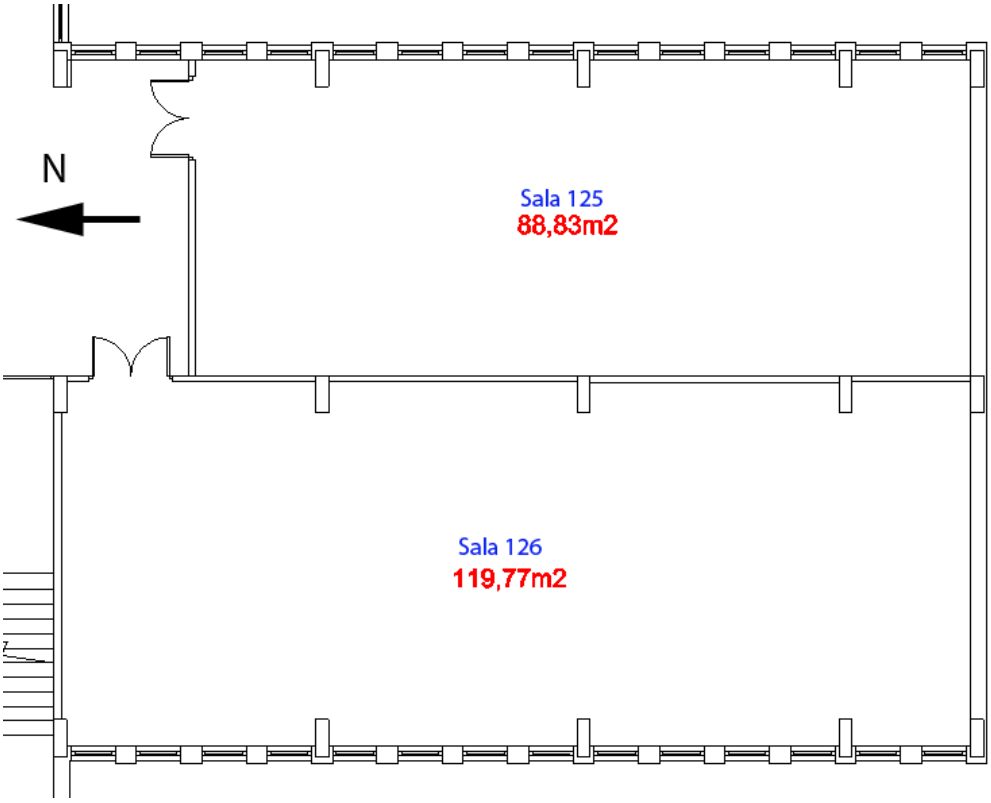
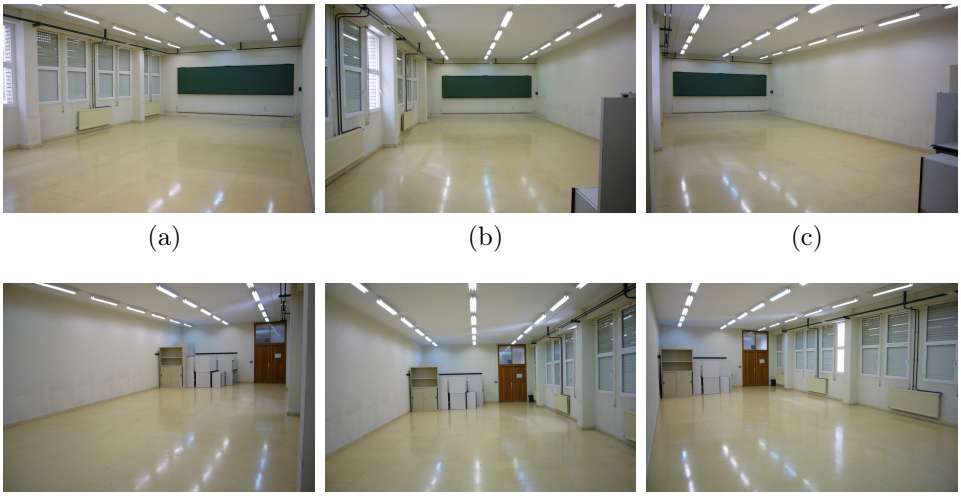


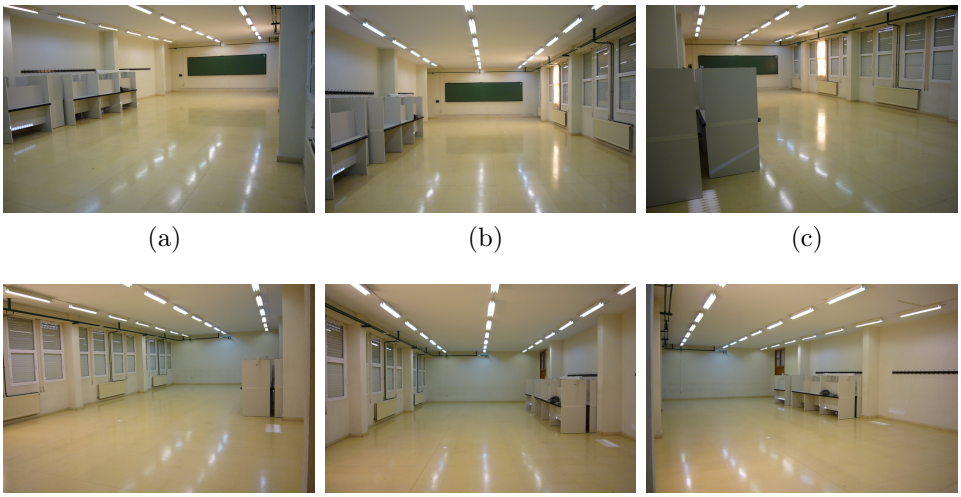
Figure E.1: Plans of rooms 125 and 126.



(a) (b) (c)

(d) (e) (f)

Figure E.2: Several views of room 125.



(a) (b) (c)

(d) (e) (f)

Figure E.3: Several views of room 126.

different paths in order to gather as complete a data set as possible from this very unique setting (which no longer exists at the time of this writing). The paths followed were:

- **Clockwise (CW):** The room is traveled following closely the walls in clockwise direction.
- **Counter Clockwise (CCW):** The room is traveled following closely the walls in counter clockwise direction.
- **Zig-Zag (ZZ):** The room is traveled crosswise following an 'S' pattern.

E.2 Stored data

As shown in section E.1 a total of 6 datasets have been recorded (2 environments, with 3 walks each). Each dataset is composed of the following data:

- **amplitudes.dat:** raw near infrared image intensity data from the 3D camera. The data is stored in a sequence of matrices in binary format, each matrix representing the amplitude image from one position. Each matrix is composed of 144×176 values of type *uint16*.
- **distancias.dat:** raw range data, the distance measurements from the 3D camera. The data is stored in a sequence of matrices in binary format, each matrix representing the distance image from one position. Each matrix is composed of 144×176 values of type *uint16*.
- **x.dat:** Values of X coordinate, computed from the raw range data from the 3D camera. The data is stored in a sequence of matrices in binary format, each matrix representing the X coordinate value corresponding to the pixel of that position. Each matrix is composed of 144×176 values of type *int16*.
- **y.dat:** Values of Y coordinate, computed from the raw range data from the 3D camera. The data is stored in a sequence of matrices in binary format, each matrix representing the Y coordinate value corresponding to the pixel of that position. Each matrix is composed of 144×176 values of type *int16*.

- **z.dat:** Values of Z coordinate, computed from the raw range data from the 3D camera. The data is stored in a sequence of matrices in binary format, each matrix representing the Z coordinate value corresponding to the pixel of that position. Each matrix is composed of 144×176 values of type *int16*.
- **posiciones.txt:** Sequence of spatial positions and orientations followed by the robot, in global coordinates, as given by robot's odometry. The origin (0,0,0) is the starting position of the robot. Text format, one position each line in the format (x, y, θ) with one space between values.
- **secuencia.mat:** Formatted data to be read by Matlab. Contains the following variables:
  - **imagenesAmp:** Cell structure of matrices of size 144×176, containing the data from *amplitudes.dat*.
  - **imagenesDist:** Cell structure of matrices of size 144×176, containing the data from *distancias.dat*.
  - **imagenesX:** Cell structure of matrices of size 144×176, containing the data from *x.dat*, in camera coordinates rotated 90° CCW.
  - **imagenesY:** Cell structure of matrices of size 144×176, containing the data from *y.dat*, in camera coordinates rotated 90° CCW.
  - **imagenesZ:** Cell structure of matrices of size 144×176, containing the data from *z.dat*, in camera coordinates rotated 90° CCW.
  - **posiciones:** Matrix n×3, containing the sequence of positions. One position each row (x, y, θ).
  - **secuenciaPuntos:** Cell containing the sequence of matrices n×3 with the point cloud for each position, integrated from *imagenesX*, *imagenesY* and *imagenesZ*, and filtered with a confidence value of 6×10^6. Each row of the matrices contains the coordinates of each point (x, y, z) in the camera reference system.
  - **secuenciaPuntosRobot:** Cell containing the sequence of matrices n×3 with the point cloud for each position, integrated from *imagenesX*, *imagenesY* and *imagenesZ*, and filtered with a confidence value of 6×10^6. Each row of the matrices contains the coordinates of each point (x, y, z) in the robot reference system.

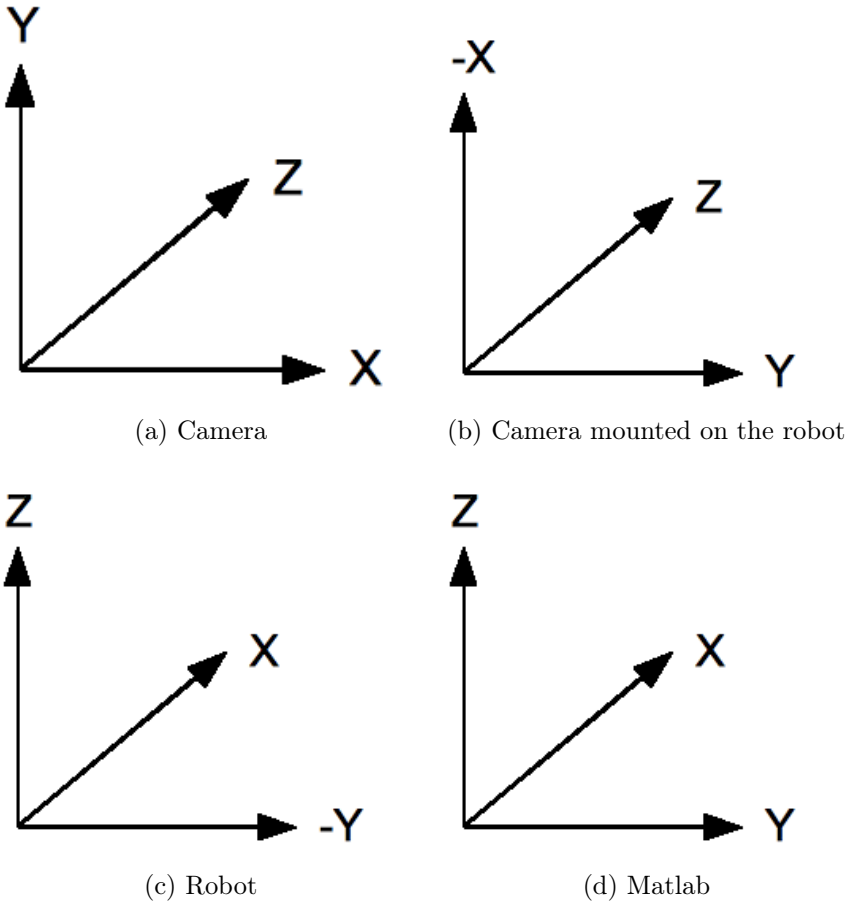
Note: Raw data is rotated 90° to the right, due the collocation of the camera in the robot, which was mounted lying down on its side, instead of being mounted straight, as explained in section C.4.2 of appendix C. In preprocessed data this has been corrected. All units are in millimeters.

E.3 Coordinate systems

When working with coordinates or displaying them as Matlab's figures, something to take into account is the reference coordinate system that is being used. Coordinate axes are situated in different order in the camera, robot and Matlab reference systems. Also has to be noted that the 3D camera is mounted on the robot rotated 90° to the right (CW), therefore, the raw data's coordinate system is also different. In Figure E.4 the different reference systems are shown.

E.4 Recorded walks

In figures E.5 to E.10 representations of the paths followed by data recording walks over the plans for the rooms are shown. Real approximate path (nominal path) is shown as a the red line. Path as recorded by odometry is shown as a blue line.



(a) Camera (b) Camera mounted on the robot

(c) Robot (d) Matlab

Figure E.4: Coordinate reference systems for (E.4a) camera, (E.4b) camera rotated 90° CW, (E.4c) robot and (E.4d) Matlab.

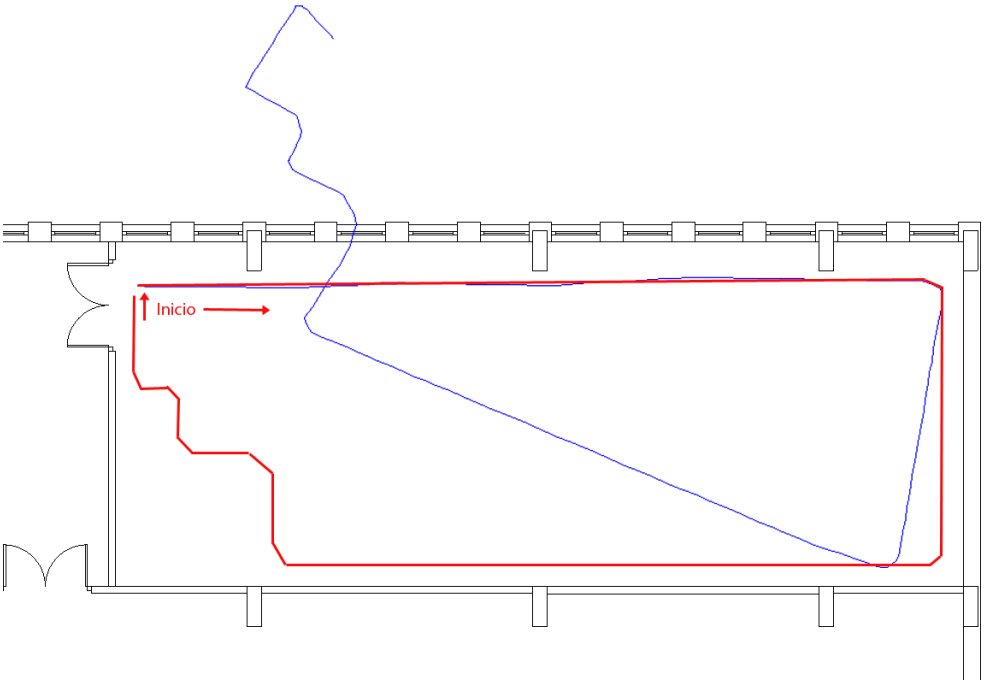


Figure E.5: Room 125. Clockwise path.

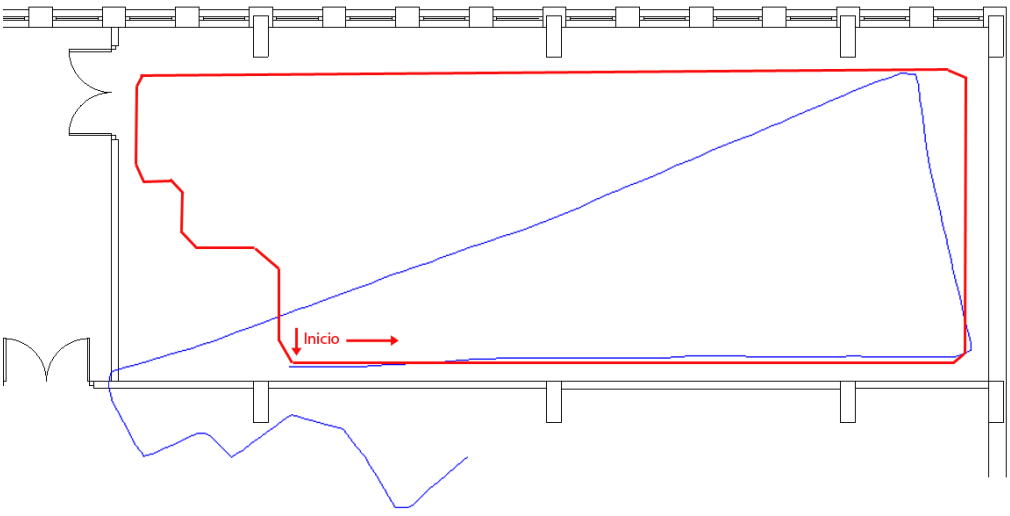


Figure E.6: Room 125. Counter Clockwise path.

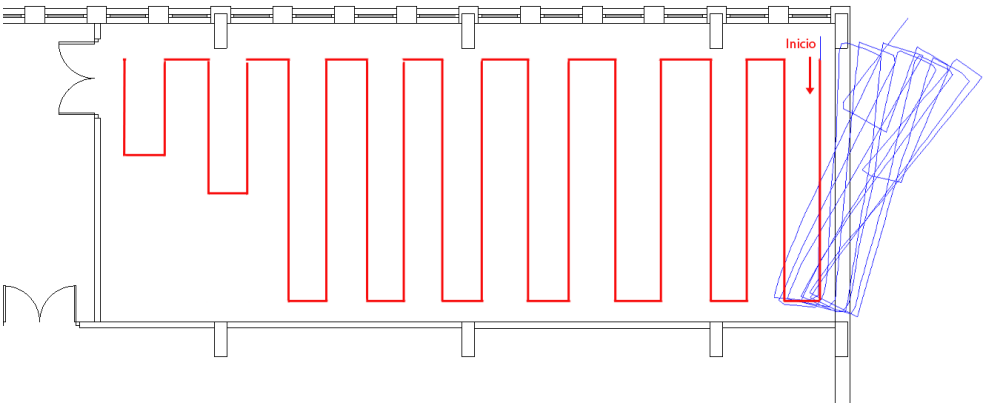


Figure E.7: Room 125. Zig-Zag path.

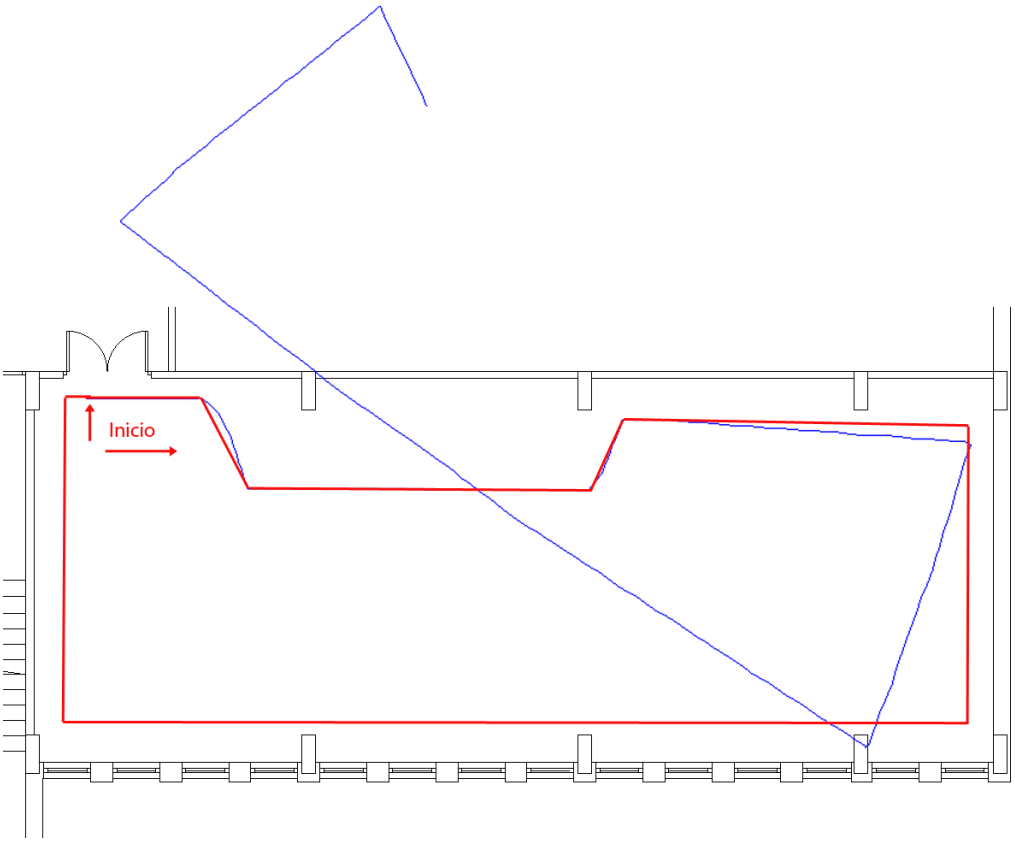


Figure E.8: Room 126. Clockwise path.

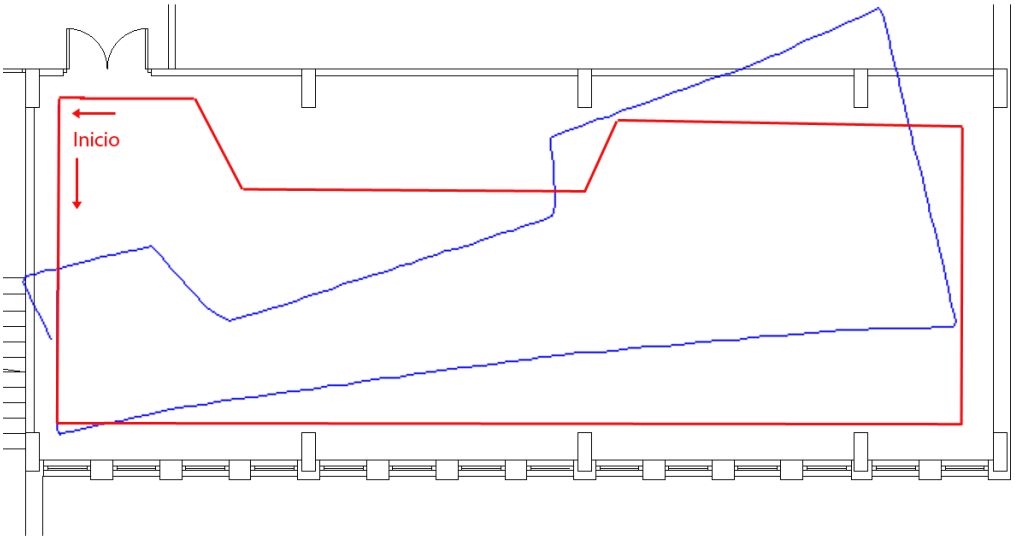


Figure E.9: Room 126. Counter Clockwise path.

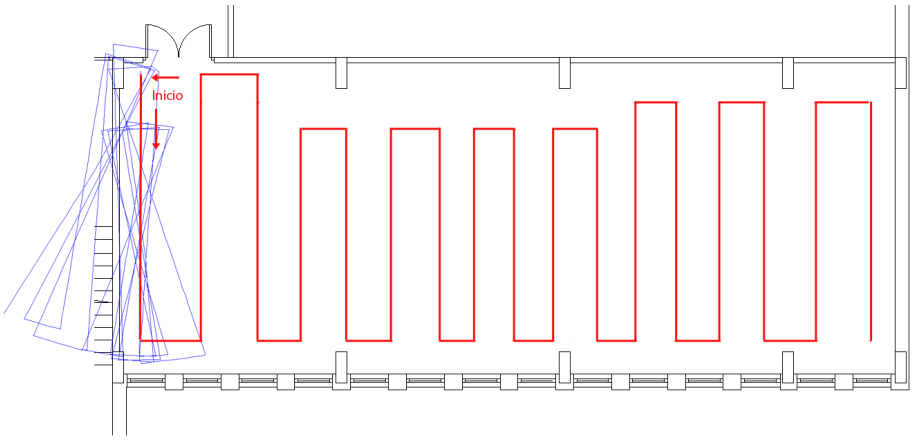


Figure E.10: Room 126. Zig-Zag path.

Bibliography

[1] H. Amano. Present status and problems of fire fighting robots. In *SICE 2002. Proceedings of the 41st SICE Annual Conference*, volume 2, pages 880–885 vol.2, Aug. 2002.

[2] G.C. Anousaki and K.J. Kyriakopoulos. Simultaneous localization and map building for mobile robot navigation. *Robotics & Automation Magazine, IEEE*, 6(3):42–53, Sep 1999.

[3] Aristotle. Politics, 350 B.C. Book 1, Part IV.

[4] Christian Balkenius and Lars Kopp. Robust self-localization using elastic templates. In T. Lindeberg, editor, *Proceedings SSAB '97: Swedish Symposium on Image Analysis 1997. Computational Vision and Active Perception Laboratory*, 1997.

[5] Garth H. Ballantyne and Fred Moll. The da Vinci telerobotic surgical system: the virtual operative field and telepresence surgery. *Surgical Clinics of North America*, 83(6):1293 – 1304, 2003. Robotic Surgery.

[6] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[7] Jon Louis Bentley. Multidimensional divide-and-conquer. *Commununications of the ACM*, 23(4):214–229, 1980.

[8] P.J. Besl and H.D. McKay. A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992.

[9] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1):3–52, March 2002.

[10] Y. Uny Cao, Alex S. Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, March 1997.

[11] R. Cassinis, D. Duina, S. Inelli, and A. Rizzi. Unsupervised matching of visual landmarks for robotic homing using Fourier-Mellin transform. *Robotics and Autonomous Systems*, 40(2-3):131 – 138, 2002.

[12] J.A. Castellanos, J.M.M. Montiel, J. Neira, and J.D. Tardos. The SPmap: a probabilistic framework for simultaneous localization and map building. *Robotics and Automation, IEEE Transactions on*, 15(5):948–952, Oct 1999.

[13] Raja Chatila. Deliberation and reactivity in autonomous mobile robots. *Robotics and Autonomous Systems*, 16(2-4):197 – 211, 1995. Moving the Frontiers between Robotics and Biology.

[14] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, 17(2):125–137, Apr 2001.

[15] Chi Kin Chow, Hung Tat Tsui, and Tong Lee. Surface registration using a dynamic genetic algorithm. *Pattern Recognition*, 37(1):105 – 117, 2004.

[16] Konstantinos Dalamagkidis, Kimon P. Valavanis, and Les. A. Piegl. Aviation history and UAS. In *On Integrating Unmanned Aircraft Systems into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations*, volume 36 of *Intelligent Systems, Control and Automation: Science and Engineering*, pages 9–28. Springer-Verlag, 2009.

[17] DARPA. Urban challenge. http://www.darpa.mil/grandchallenge.

[18] A. Davids. Urban search and rescue robots: from tragedy to technology. *Intelligent Systems, IEEE*, 17(2):81–83, March-April 2002.

[19] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410 vol.2, Oct. 2003.

[20] Mark de Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational geometry*. Springer, third edition, 2008.

[21] Guilherme N. DeSouza and Avinash C. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[22] Oxford Dictionaries. *Compact Oxford English Dictionary of Current English: Third edition revised*. OUP Oxford, new ed of 3rd revised ed edition, June 2008.

[23] Gamini Dissanayake. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[24] Marco Dorigo, Vito Trianni, Erol Sahin, Roderich Gross, Thomas H. Labella, Gianluca Baldassarre, Stefano Nolfi, Jean-Louis Deneubourg, Francesco Mondada, Dario Floreano, and Luca M. Gambardella. Evolving Self-Organizing behaviors for a Swarm-Bot. *Autonomous Robots*, 17(2):223–245, 2004.

[25] Gregory Dudek, Michael R. M. Jenkin, Evangelos Milios, and David Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, December 1996.

[26] Richard J. Duro, Manuel Graña, and Javier de Lope. On the potential contributions of hybrid intelligent approaches to multicomponent robotic system development. *Information Sciences*. Accepted for publication.

[27] Zelmar Echegoyen. *Contributions to Visual Servoing for Legged and Linked Multicomponent Robots*. PhD thesis, University of the Basque Country, 2009.

[28] Zelmar Echegoyen, Alicia D'Anjou, Ivan Villaverde, and Manuel Graña. Towards the adaptive control of a multirobot system for an elastic hose. In Vassilis Kaburlasos, Uta Priss, and Manuel Graña, editors, *Advances in Neuro-Information Processing*, volume 5506/2009 of *Lecture Notes in Computer Science*, pages 1045–1052. Springer, 2009. ISBN 978-80-244-2112-4.

[29] C. Estrada, J. Neira, and J.D. Tardos. Hierarchical SLAM: Real-time accurate mapping of large environments. *Robotics, IEEE Transactions on*, 21(4):588–596, Aug. 2005.

[30] A. Farinelli, L. Iocchi, and D. Nardi. Multirobot systems: a classification focused on coordination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5):2015–2028, Oct. 2004.

[31] Charles B. Fowler. The museum of music: A history of mechanical instruments. *Music Educators Journal*, 54(2):45–49, October 1967.

[32] Dieter Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation.* PhD thesis, University of Bonn, Germany, 1998.

[33] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *Robotics, IEEE Transactions on*, 21(2):196–207, April 2005.

[34] Zhouyu Fu, R.T. Tan, and T. Caelli. Specular free spectral imaging using orthogonal subspace projection. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 812–815, 2006.

[35] Keinosuke Fukunaga. *Introduction to statistical pattern recognition.* Academic Press, 1990.

[36] Rafael C. Gonzalez and Richard Eugene Woods. *Digital image processing.* Prentice Hall, August 2007.

[37] Manuel Graña. A brief review of Lattice Computing. In *Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence). IEEE International Conference on*, pages 1777–1781, June 2008.

[38] Manuel Graña and Alicia D'Anjou. Feature extraction by linear spectral unmixing. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3213/2004 of *Lecture Notes in Computer Science*, pages 692–698. Springer-Verlag, 2004.

[39] Manuel Graña, Alicia D'Anjou, and Francisco Xabier Albizuri. *ESANN 2005*, chapter Morphological memories for feature extraction in hyperspectral images, pages 497–502. dFacto press, 2005.

[40] Manuel Graña and Josune Gallego. Associative morphological memories for endmember induction. In *Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International*, volume 6, pages 3757–3759 vol.6, July 2003.

[41] Manuel Graña, Maite García-Sebastián, Ivan Villaverde, and Elsa Fernandez. *Proceedings of the Lattice-Based Modeling Workshop, in conjunction with The Sixth International Conference on Concept Lattices and Their Applications*, chapter An approach from Lattice Computing to fMRI analysis, pages 33–44. 2008.

[42] Manuel Graña, Bogdan Raducanu, Peter Sussner, and G. Ritter. On endmember detection in hyperspectral images with morphological associative memories. In *Advances in Artificial Intelligence - IBERAMIA 2002*, volume 2527/2002 of *Lecture Notes in Computer Science*, pages 526–535. Springer, 2002.

[43] Manuel Graña, Peter Sussner, and Gerhard Ritter. Associative morphological memories for endmember determination in spectral unmixing. In *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, volume 2, pages 1285–1290 vol.2, May 2003.

[44] Manuel Graña, Ivan Villaverde, Jose Manuel López Guede, and Borja Fernández. Review of hybridizations of Kalman filters with fuzzy and neural computing for mobile robot navigation. In E. Corchado, E. Oja X. Wu, A. Herrero, and B. Baruque, editors, *Hybrid Artificial Intelligence Systems*, volume 5572/2009 of *Lecture Notes in Computer Science*, pages 121–128. Springer, 2009.

[45] Manuel Graña, Ivan Villaverde, Jose Orlando Maldonado, and Carmen Hernandez. Two Lattice Computing approaches for the unsupervised segmentation of hyperspectral images. *Neurocomputing*, 72(10-12):2111–2120, 2009.

[46] Manuel Graña, Ivan Villaverde, Ramón Moreno, and Francisco Xabier Albizuri. *Computational Intelligence Based on Lattice Theory*, chapter

Convex Coordinates From Lattice Independent Sets for Visual Pattern Recognition, pages 99–126. Springer-Verlag, 2007. ISBN 978-3-540-72686-9.

[47] H.-M. Gross, A. Koenig, H.-J. Boehme, and Ch. Schroeter. Vision-based Monte Carlo self-localization for a mobile service robot acting as shopping assistant in a home store. In *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, volume 1, pages 256–262 vol.1, 2002.

[48] J.E. Guivant and E.M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *Robotics and Automation, IEEE Transactions on*, 17(3):242–257, Jun 2001.

[49] J.E. Guivant and E.M. Nebot. Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms. *Robotics and Automation, IEEE Transactions on*, 19(4):749–755, Aug. 2003.

[50] Sigurjón A. Guðmundsson, Rasmus Larsen, and Bjarne K. Ersbøll. Robust pose estimation using the SwissRanger SR-3000 camera. In *Image Analysis*, volume 4522/2007 of *Lecture Notes in Computer Science*, pages 968–975. Springer, 2007.

[51] Robert M. Haralick and Linda G. Shapiro. *Computer and robot vision.* Addison-Wesley Pub. Co., 1992.

[52] Homer. The iliad, 800 B.C. Book XVIII.

[53] Berthold Horn. *Robot vision.* MIT Press, 1986.

[54] Luca Iocchi, Daniele Nardi, and Massimiliano Salerno. Reactivity and deliberation: A survey on Multi-Robot systems. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, volume 2103/2001 of *Lecture Notes in Computer Science*, pages 9–32. Springer, 2001.

[55] J.L. Jones. Robots at the tipping point: the road to iRobot Roomba. *Robotics & Automation Magazine, IEEE*, 13(1):76–78, March 2006.

[56] Vassilis G. Kaburlasos. *Towards a Unified Modeling and Knowledge-Representation based on Lattice Theory*. Springer Verlag, 2006.

[57] Vassilis G. Kaburlasos and Gerhard X. Ritter. *Computational Intelligence Based on Lattice Theory*. Springer Verlag, 2007.

[58] N. Keshava. A survey of spectral unmixing algorithms. *Lincoln Laboratory Journal*, 14(1):55–78, 2003.

[59] N. Keshava and J.F. Mustard. Spectral unmixing. *Signal Processing Magazine, IEEE*, 19(1):44–57, Jan 2002.

[60] T. Kohonen and T. Honkela. Kohonen network. *Scholarpedia*, 2(1):1568, 2007.

[61] Teuvo Kohonen. *Self-Organizing Maps*. Springer, 2001.

[62] R. Lange and P. Seitz. Solid-state Time-of-Flight range camera. *IEEE J. Quantum Electronics*, 37 (3)(3):390–397, March 2001.

[63] C.L. Lawson and R.J. Hanson. *Solving least squares problems*. Englewood Cliffs : Prentice-Hall, 1974.

[64] Salvatore Livatino and Claus B. Madsen. Autonomous robot navigation with automatic learning of visual landmarks. In *Symposium on Intelligent Robotics Systems (SIRS*, 1999.

[65] Jose Manuel Lopez-Guede, Manuel Graña, Ekaitz Zulueta, and Oscar Barambones. Economical implementation of control loops for multi-robot systems. In *Advances in Neuro-Information Processing*, volume 5506/2009 of *Lecture Notes in Computer Science*, pages 1053–1059. Springer, 2009.

[66] R. Madhavan and H. F. Durrant-Whyte. Natural landmark-based autonomous vehicle navigation. *Robotics and Autonomous Systems*, 46(2):79 – 95, 2004.

[67] Francesco Marando, Maurizio Piaggio, and Alessandro Scalzo. Real time self localization using a single frontal camera. In *International symposium on intelligent robotic systems No9, Toulouse , FRANCE*, pages 435–444. LAAS-CNRS, Toulouse, FRANCE, 2001.

[68] Stephen Marsland, Ulrich Nehmzow, and Tom Duckett. Learning to select distinctive landmarks for mobile robot navigation. *Robotics and Autonomous Systems*, 37(4):241 – 260, 2001.

[69] T. M. Martinetz and K. J. Schulten. *Proc. International Conference on Artificial Neural Networks*, chapter A neural-gas network learns topologies, pages 397–402. North-Holland, Amsterdam, 1991.

[70] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten. 'Neural-gas' network for vector quantization and its application to time-series prediction. *Neural Networks, IEEE Transactions on*, 4(4):558–569, Jul 1993.

[71] Bruce A. Maxwell, Lisa A. Meeden, Laura Brown Nii Addo, Paul Dickson, Jane Ng, Seth Olshfski, Eli Silk, and Jordan Wales. Alfred: The robot waiter who remembers you. In *Proceedings of AAAI Workshop on Robotics*, July 1999.

[72] Stephen McPhail. Autosub6000: A deep diving long range AUV. *Journal of Bionic Engineering*, 6(1):55–62, March 2009.

[73] NASA. Mars exploration rover mission. http://marsrovers.jpl.nasa.gov.

[74] NASA. National space science data center. NSSDC ID: 1966-006A, http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=1966-006A.

[75] NASA. National space science data center. NSSDC ID: 1970-060A, http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=1970-060A.

[76] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Transactions on*, 17(6):890–897, Dec 2001.

[77] Nils J. Nilsson. Shakey the Robot. Technical Report no. 323, SRI International, April 1984.

[78] Shimon Y. Nof. *Handbook of industrial robotics*. John Wiley and Sons, 1999.

[79] T. Oggier, M. Lehmann, R. Kaufmannn, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, and N. Blanc. An all-solid-state optical range camera for 3D-real-time imaging with sub-centimeter

depth-resolution (SwissRanger). In *Proc. SPIE*, volume 5249, pages 634–545, 2003.

[80] I. Ohya, A. Kosaka, and A. Kak. Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. *Robotics and Automation, IEEE Transactions on*, 14(6):969–978, Dec 1998.

[81] C.F. Olson. Landmark selection for terrain matching. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 2, pages 1447–1452 vol.2, 2000.

[82] C.F. Olson. Probabilistic self-localization for mobile robots. *Robotics and Automation, IEEE Transactions on*, 16(1):55–66, Feb 2000.

[83] Martin J. Pearson, Anthony G. Pipe, Chris Melhuish, Ben Mitchinson, and Tony J. Prescott. Whiskerbot: A robotic active touch system modeled on the rat whisker sensory system. *Journal of Adaptive Behavior*, 15(3):223–240, 2007.

[84] E. Potemkin, P. Astafurov, A. Osipov, M. Malenkov, V. Mishkinyuk, and P. Sologub. Remote-controlled robots for repair and recovery in the zones of high radiation levels. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 80–82 vol.1, May 1992.

[85] B. Raducanu, M. Grana, and P. Sussner. Morphological neural networks for vision based self-localization. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 2059–2064 vol.2, 2001.

[86] Bogdan Raducanu, Manuel Graña, and F. Xabier Albizuri. Morphological scale spaces and associative morphological memories: Results on robustness and practical applications. *Journal of Mathematical Imaging and Vision*, 19(2):113–131, 2003.

[87] Bogdan Raducanu, Manuel Graña, and Peter Sussner. *Biologically Inspired Robot Behavior Engineering*, volume 19 of *Studies in Fuzziness and Soft Computing*, chapter Steps towards one-shot vision-based self-localization, pages 265–294. Springer-Verlag, 2002.

[88] L. H. Randy and E. H. Sue. *Practical Genetic Algorithms*. Wiley-Interscience, 2nd edition, May 2004.

[89] J. Reuter. Mobile robot self-localization using PDAB. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 4, pages 3512–3518 vol.4, 2000.

[90] Jessica Riskin. The defecating duck, or, the ambiguous origins of artificial life. *Critical Inquiry*, 29(4):599–633, 2003.

[91] G. X. Ritter, J. L. Diaz de Leon, and P. Sussner. Morphological bidirectional associative memories. *Neural Networks*, 12(6):851 – 867, 1999.

[92] Gerhard Ritter and Paul Gader. Fixed points of Lattice Transforms and Lattice Associative Memories. volume 144 of *Advances in Imaging and Electron Physics*, pages 165 – 242. Elsevier, 2006.

[93] Gerhard X. Ritter, Gonzalo Urcid, and Laurentiu Iancu. Reconstruction of patterns from noisy inputs using Morphological Associative Memories. *Journal of Mathematical Imaging and Vision*, 19(2):95–111, 2003.

[94] Gerhard X. Ritter, Gonzalo Urcid, and M.S. Schmalz. Autonomous single-pass endmember approximation using Lattice Auto-Associative Memories. *Neurocomputing*, 72(10-12):2101–2110, 2009.

[95] G.X. Ritter, P. Sussner, and J.L. Diza-de Leon. Morphological Associative Memories. *Neural Networks, IEEE Transactions on*, 9(2):281–293, Mar 1998.

[96] Alessandro Saffiotti and Leonard Wesley. Perception-based self-localization using fuzzy locations. In L. Dorst, M. Van Lambalgen, and F. Voorbraak, editors, *Reasoning with Uncertainty in Robotics*, volume 1093/1996 of *Lecture Notes in Computer Science*, pages 368–385. Springer, 1996.

[97] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent ASIMO: system overview and integration. In *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2478–2483 vol.3, 2002.

[98] Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578 – 596, 2007.

[99] Daisuke Sekimori, Tomoya Usui, Yasuhiro Masutani, and Fumio Miyazaki. High-Speed obstacle avoidance and Self-Localization for mobile robots based on omni-directional imaging of floor region. In *RoboCup 2001: Robot Soccer World Cup V*, volume 2377/202 of *Lecture Notes in Computer Sciencie*, pages 79–118. Springer, 2002.

[100] Steven A. Shafer. Using color to separate reflection components. *Color Research and Aplications*, 10:43–51, april 1984.

[101] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice Hall, January 2001.

[102] N. E. Sharkey and A. J. C. Sharkey. Electro-mechanical robots before the computer. In *Proc. IMechE Part C: Journal of Mechanical Engineering Science*, volume 223, pages 235–241, 2009.

[103] Noel Sharkey. I ropebot. *The New Scientist*, 195(2611):32 – 35, 2007.

[104] Shigetoshi Shiotani, Tetsuya Tomonaka, Keiichi Kemmotsu, Shin Asano, Ken Oonishi, and Ryouta Hiura. World's first full-fledged communication robot "Wakamaru" capable of living with family and supporting persons. Technical Review Vol. 43 No. 1, Mitsubishi Heavy Industries, Ltd., 2006.

[105] Peter Sussner. Fixed points of autoassociative morphological memories. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 5, pages 611–616 vol.5, 2000.

[106] Mario Taddei. *Leonardo da Vinci's Robots*. Leonardo3, 2007.

[107] Robby T Tan, Ko Nishino, and Katsushi Ikeuchi. Separating reflection components based on chromaticity and noise analysis. *IEEE Trans Pattern Anal Mach Intell*, 26(10):1373–1379, Oct 2004.

[108] T.T. Tan, K. Nishino, and K. Ikeuchi. Illumination chromaticity estimation using inverse-intensity chromaticity space. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–673–I–680vol.1, 18-20 June 2003.

[109] S. Thrun, C. Martin, Yufeng Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard. A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots. *Robotics and Automation, IEEE Transactions on*, 20(3):433–443, June 2004.

[110] Sebastian Thrun. *Exploring Artificial Intelligence in the New Millenium*, chapter Robotic Mapping: A Survey. Morgan Kaufmann, 2002.

[111] Nicola Tomatis, Illah Nourbakhsh, and Roland Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, 44(1):3 – 14, 2003. Best Papers of the Eurobot '01 Workshop.

[112] J. Vesanto, E. Alhoniemi, J. Himberg, K. Kiviluoto, and J. Parviainen. Self-Organizing Map for data mining in Matlab: the SOM toolbox. *Simulation News Europe*, 25(54):54, March 1999. http://www.cis.hut.fi/projects/somtoolbox/.

[113] Ivan Villaverde. *Actas de las I Jornadas de Inteligencia Computacional - JIC'07*, chapter 3D Camera for Mobile Robot SLAM, pages 248–253. Basque Country University Editorial, 2007. ISBN 978-84-9860-019-3.

[114] Ivan Villaverde, Alicia D'Anjou, and Manuel Graña. Morphological Neural Networks and vision based simultaneous localization and mapping. *Integrated Computer-Aided Engineering*, 14(4)(14):355–363, 2007. IOS Press.

[115] Ivan Villaverde, Zelmar Echegoyen, and Manuel Graña. Neuro-evolutive system for ego-motion estimation with a 3D camera. *Australian Journal of Intelligent Information Systems*, 10(1):59–70, 2008. ISSN 1321-2133.

[116] Ivan Villaverde, Zelmar Echegoyen, and Manuel Graña. Neuro-evolutive system for ego-motion estimation with a 3D camera. In

M. Köppen, editor, *Advances in Neuro-Information Processing*, volume 5506/2009 of *Lecture Notes in Computer Sciences*, pages 1021–1028. Springer-Verlag, 2009.

[117] Ivan Villaverde and Manuel Graña. A hybrid intelligent system for robot ego-motion estimation with a 3D camera. In Emilio Corchado, Ajith Abraham, and Witold Pedrycz, editors, *Hybrid Artificial Intelligence Systems*, volume 5271 of *Lecture Notes in Artificial Inteligence*, pages 657–664. Springer-Verlag, 2008.

[118] Ivan Villaverde and Manuel Graña. An improved evolutionary approach for egomotion estimation with a 3D TOF camera. In J. Mira et al., editor, *Bioinspired Applications in Artificial and Natural Computation*, volume 5602/2009 of *Lecture Notes in Computer Science*, pages 390–398. Springer, 2009. ISBN 978-3-642-02266-1.

[119] Ivan Villaverde, Manuel Graña, and Alicia D'Anjou. Morphological Neural Networks and vision based mobile robot navigation. In *Artificial Neural Networks - ICANN 2006*, volume 4131 of *Lecture Notes in Computer Science*, pages 878–887. Springer-Verlag, 2006. ISBN 3-540-38625-4.

[120] Ivan Villaverde, Manuel Graña, and Alicia D'Anjou. Morphological Neural Networks for localization and mapping. In *Proceedings of the IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 06)*, pages 9–14. IEEE Press, 2006. ISBN 1-4244-0245-X.

[121] Ivan Villaverde, Manuel Graña, and Alicia D'Anjou. Morphological Independence for landmark detection in vision based SLAM. In F. Sandoval, A. Prieto, J. Cabestany, and M. Graña, editors, *Proc. of IWANN 2007*, volume 4507 of *Lecture Notes in Computer Science*, pages 847–854. Springer-Verlag, 2007. ISBN 3-540-73006-0.

[122] Ivan Villaverde, Manuel Graña, and Jose Luis Jiménez. Lattice Independence and vision based mobile robot navigation. In Bruno Apolloni, Robert J. Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems, KES 2007*, volume 4693/2009 of *Lecture Notes in Artificial Inteligence*, pages 1196–1203. Springer-Verlag, 2007. ISBN 978-3-540-74826-7.

[123] Ivan Villaverde, Sergio Ibáñez, Francisco Xabier Albizuri, and Manuel Graña. Morphological Neural Networks for real-time vision based self localization. In Ajith Abrham, Y. Dote, T. Furuhashi, M. Köpen, A. Ohuchi, and Y. Ohsawa, editors, *Soft Computing as transdisciplinary Science and Techonology, Proc. WSTST'05*, volume 29/2005 of *Advances in Soft Computing*, pages 70–79. Springer-Verlag, 2005. ISBN 3-540-25055-7.

[124] Cang Ye and G.-P.M. Hegde. Robust edge extraction for SwissRanger SR-3000 range images. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2437–2442, May 2009.

[125] Kuk-Jin Yoon, Yoojin Choi, and In So Kweon. Fast separation of reflection components using a specularity-invariant image representation. In *Image Processing, 2006 IEEE International Conference on*, pages 973–976, 8-11 Oct. 2006.

[126] T. Zinsser, H. Schnidt, and J. Niermann. A refined ICP algorithm for robust 3-D correspondences estimation. In *International Conference on Image Processing*, pages 695–698, 2003.