UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

Evolutionary-based Global Localization and Mapping of three-dimensional environments

Autor:
Fernando Martín Monar
Director:
Luis Moreno Lorente
Codirector:
Santiago Garrido Bullón

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA
Leganés, Febrero 2012

TESIS DOCTORAL (THESIS)

Evolutionary-based Global Localization and Mapping of three-dimensional environments

Autor (Candidate): Fernando Martín Monar

Director (Adviser): Luis Moreno Lorente
Codirector (Co-Adviser): Santiago Garrido Bullón

Tribunal (Review Committee)

Presidente (Chair): Carlos Balaguer Bernaldo de Quirós	
Vocal (Member): Jose Ángel Castellanos López	
Vocal (Member): Antonio Giménez Fernández	
Vocal (Member): Pedro Lima	
Secretario (Secretary): Dolores Blanco Rojas	
Suplente (Substitute): Jose María Sebastián y Zuñiga	
Suplente (Substitute): Fernando Matía	

Título (Grade): Doctorado en Ingeniería Eléctrica, Electrónica y Automática
Calificación: ____________________

Leganés, 7 de Febrero de 2012

Abstract

A fully autonomous robot must obtain and interpret information about the environment to execute several tasks. The mobile robot mapping or SLAM problem is closely related to these abilities. It consists of interpreting the information perceived by its sensors in order to build map and localize itself in it. There are many other robot skills that depend on this task; thus, it is one of the most important problems to be solved by a truly autonomous robot. The objective of this work is to design various specific tools related to the mapping problem in order to improve the autonomy of MANFRED-2, which is a mobile robot fully developed by the Robotics Lab research group of the Systems Engineering and Automation Department of the Carlos III University of Madrid.

The localization problem in mobile robotics can be defined as the search of the robot's coordinates in a known environment. If there is no information about the initial location, we are talking about global localization. In this work, we have developed an algorithm that solves this problem in a three-dimensional environment using Differential Evolution, which is a particle-based evolutionary algorithm that evolves in time to the solution that yields the cost function lowest value. The proposed method has many features that make it very robust and reliable: thresholding and discarding mechanisms, different cost functions, effective convergence criteria, and so on. The resulting global localization module has been tested in numerous experiments. The high accuracy of the method allows its application in manipulation tasks.

If the environment information is given by laser readings, it is essential to correct the local errors between pairs of scans to improve the map quality, which is called registration or scan matching. We have implemented a scan matching algorithm for three-dimensional environments. It is also based on the Differential Evolution method. The high accuracy and computational efficiency of the proposed method have been demonstrated with experimental results.

The last problem addressed here consists of detecting when the robot is navigating through a known place (loop detection). After that, the accumulated error can be minimized to give consistency to the global map (loop closure). We have developed a loop detection method that compares features extracted from two different scans to obtain a loop indicator. This approach allows the introduction of very different characteristics in the descriptor. First, the *surface* features include the geometric forms of the scan (lines, planes, and spheres). Second, the *numerical* features describe other several properties (volume, average range, curvature, etc.). The algorithm has been tested with real data to demonstrate its efficiency. All true loops are correctly detected and no false detections are appreciated when the mobile robot is covering a long trajectory. The results are similar or even better than those obtained by other research groups. In addition, it is a more versatile method because it admits a wide variety of scan properties and different weights in the comparison formula.

Resumen

Un robot completamente autónomo debe ser capaz de obtener e interpretar la información del entorno para ejecutar diversas tareas. El problema de mapeado o SLAM para robots móviles está estrechamente relacionado con estas habilidades. Consiste en interpretar la infomación percibida por sus sensores para construir un mapa y localizarse. Hay muchas otras tareas que dependen del mapeado, luego este es uno de los problemas más importantes para un robot móvil. El objetivo de este trabajo es el desarrollo de varias herramientas específicas relacionadas con el mapeado de entornos tridimensionales. Con ellas se mejorará la autonomía del robot manipulador MANFRED-2, que es un robot móvil desarrollado íntegramente en el Robotics Lab del Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid.

El problema de localización para un robot móvil puede ser definido como la búsqueda de las coordenadas del robot dentro de un entorno conocido. Si no hay información sobre la localización inicial, el problema se denomina localización global. En este trabajo se ha desarrollado un módulo que soluciona este problema para entornos tridimensionales utilizando el algoritmo *Differential Evolution*, el cual es un filtro evolutivo basado en partículas que evolucionan con el tiempo hacia la solución que tiene asociado un mejor valor para una función de coste dada. El algoritmo desarrollado tiene diversas características que lo hacen muy robusto y fiable: mecanismos de umbralización y descarte, diferentes funciones de coste, criterios de convergencia efectivos, etc. El módulo de localización global se ha probado en múltiples experimentos. La elevada precisión de este método permite que el robot sea utilizado en tareas de manipulación.

Si la información del entorno viene dada por barridos de un láser, es muy importante que se pueda corregir el error local entre pares de barridos para mejorar la calidad del mapa. Este proceso se conoce como registro o *scan matching*. Hemos implementado un algoritmo que resuelve este problema en entornos tridimensionales. Está también basado en el *Differential Evolution*. Si se elige la función de forma adecuada es posible resolver el problema de *scan matching* utilizando este método. La elevada precisión y la eficiencia computacional se han demostrado en los resultados experimentales.

El último problema abordado aquí consiste en detectar cuando el robot está navegando por un entorno conocido. Después de esto se podrá minimizar el error acumulado para aumentar la consistencia del mapa. La tarea de detección se llama usualmente detección de bucles, mientras que la minimización del error es el cierre del bucle. Se ha desarrollado un algoritmo de detección que extrae las características más importantes de dos barridos del láser para obtener un indicador que es usado como umbral para detectar si el robot está en un lugar que ha visitado previamente. Nuestro método permite tener en cuenta características muy diferentes. Primero, las características de superficie permiten incluir las formas geométricas presentes en el barrido (líneas, planos y esferas). Segundo, las características numéricas permiten describir diversas propiedades (volumen, rango medio, curvatura, etc.). El algoritmo ha sido probado con datos reales para demostrar su eficiencia. Todos los bucles son detectados correctamente y no se aprecian falsos positivos cuando el robot está navegando por una trayectoria larga con varios bucles. Los resultados son parecidos o mejores que los que obtienen otros grupos de investigación. Además, este es un método muy versátil pues admite multitud de variables y diferentes pesos en la fórmula de comparación.

A mi hermano Alberto
y a mis padres.

Agradecimientos

Cuando me puse a pensar para escribir estos párrafos me di cuenta de que es muy difícil poner nombre y apellidos a todos los que me han ayudado durante todo este tiempo. Así que quiero empezar disculpándome con aquellos a los que no cite aquí.

Primero quería darle las gracias a mi director de tesis, Luis, por el tiempo dedicado y las ideas aportadas durante la realización de este trabajo. Es mucho el tiempo que hemos invertido y aquí está el fruto. Realmente es un placer trabajar contigo.

Luego me tengo que acordar de mi familia, que siempre me ha apoyado en todas mis decisiones: mis padres Paco y Lolina, mi hermano Alberto y su mujer Ari, mi sobrina Lucía (alguna vez lo leerá y seguro que se enfada si no la menciono aquí), mis abuelas Lola y Manuela, mis tíos y primos...

Posteriormente quiero centrarme en mis compañeros del Robotics Lab y del Departamento de Ingeniería de Sistemas y Automática. Sobre todo tengo mucho que agradecerle a mi codirector, Santiago, por todo lo que me ha ayudado y enseñado en esta tésis y, en general, en mi actividad como investigador. También quiero resaltar a Dolores, pues su ayuda ha sido también muy importante y he aprendido mucho de ella. También están: Paolo, Álvaro, Carla, Javi, Concha, María, Marisa, Santi, Alberto, Ángela, Edu, Sonia, Carlos, Miguel Ángel, Ramón, Mohamed, Carlos P., Jorge, Alex, David, Fran, Alberto, César, Ana, Jose, Fernando, Nico, Juan, Teresa (ya sé que no vas aquí, pero tenía q meterte en algún lado)... Me olvido de varios seguro, pero es que son muchos los que han aportado algo durante todos estos años.

A continuación quería darle las gracias a Rudolph Triebel por permitirme trabajar con él en el ETH de Zürich (Swiss Federal Institute of Technology), dirigido por Roland Siegwart, donde realicé una estancia de tres meses y medio en la que aprendí y progresé mucho de cara a la realización de esta tesis.

También hay un lugar en estas líneas para mis amigos que no tienen nada que ver con el trabajo, que aunque no lo crean y la mayoría no tengan ni idea de este tema, me han ayudado mucho. Primero los de toda la vida, en riguroso orden alfabético para que no se enfaden: Ancor, Arexe, David, Javi, Marci, Mario, Matías, Mingo, Nico... Nos vamos haciendo mayores, a partir de ahora me tienen que tener respeto. Luego los que he conocido en Madrid durante todos estos años. Sobre todo a Juanma (un segundo hermano), Juan Molín, Carlos, Iulian... También Paolo y Álvaro, que repiten, pues aunque sean compañeros de la universidad también tienen que estar aquí citados. Y finalmente quiero citar a los entrenadores, jugadores y demás componentes que han coincidido conmigo en el Baloncesto Getafe. Realmente no supe lo que era Getafe hasta que empecé a entrenar en este club, y eso que antes estuve viviendo allí durante siete años mientras estudiaba la carrera. Sobre todo me acuerdo de Fran, que ya son muchos años dando vueltas por los pabellones madrileños y haciendo todo tipo de "freakadas", que seguro que es una palabra que no existe, pero es la adecuada.

Uno no se puede olvidar de aquellos organismos públicos que con su financiación hacen posible el desarrollo de la investigación. Particularmente me acuerdo del Ministerio de Ciencia e Innovación (este es el nombre que me parece más adecuado de los que ha tenido, aunque ahora tenga un nombre diferente) y de la Comunidad de Madrid.

Y no puedo terminar sin nombrar a Mathworks. Gracias por hacerme la vida más fácil.

Abbreviations

2D - two-dimensional
3D - three-dimensional
6D - six-dimensional
CCD - Charge-Coupled Device
DAC - Digital-to-Analog Converters
DE - Differential Evolution
DOF - Degrees of Freedom
DSP - Digital Signal Processor
EA - Evolutionary Algorithms
EIF - Extended Information Filter
EKF - Extended Kalman Filter
ELF- Evolutionary Localization Filter
EM - Expectation Maximization
GA - Genetic Algorithms
GPS - Global Positioning System
ICP - Iterative Closest Points
IDC - Iterative Dual Correspondence
KNN - K-Nearest Neighbours
LPI - Loop Probability Indicator
LSM - Least Squares Method
MAP - Maximum A Posteriori
MC - Monte Carlo
MLE - Maximum Likelihood Estimation
MLS - Multi-Level Surface
NDT - Normal Distributions Transform
NN - Nearest Neighbour
PD - Probability of Detection
PFA - Probability of False Alarm
PLC - Programmable Logic Controller
PSM - Polar Scan Matching
PSO - Particle Swarm Optimization

R&D - Research and Development
RELF-3D - Rejection Evolutionary Localization Filter in Three Dimensions
RWS - Roulette Wheel Selection
SEIF - Sparse Extended Information Filter
SIFT - Scale-Invariant Feature Transform
SLAM - Simultaneous Localization And Mapping
SURF - Speeded Up Robust Feature
SUS - Stochastic Universal Sampling
UC3M - Carlos III University of Madrid
VRML - Virtual Reality Modeling Language

Contents

Abstract **i**

Resumen **iii**

Agradecimientos **vii**

Abbreviations **ix**

1 Introduction **1**

2 State of the Art **11**
- 2.1 Mapping/SLAM . . . 18
  - 2.1.1 Planar mapping . . . 19
  - 2.1.2 Volumetric mapping . . . 20
- 2.2 Global Localization . . . 25
- 2.3 Scan Matching . . . 29
- 2.4 Loop Detection . . . 33

3 Global Localization **39**
- 3.1 Localization Problem Formulation and Solution . . . 43
  - 3.1.1 Localization as a MAP optimization problem . . . 47
  - 3.1.2 Recursive formulation of the optimization problem . . . 49
- 3.2 Evolutionary Localization Filter (RELF-3D) . . . 50
  - 3.2.1 Mutation, crossover, and selection . . . 52
  - 3.2.2 Thresholding mechanism . . . 57
  - 3.2.3 Discarding mechanism . . . 59
  - 3.2.4 Fitness function . . . 61
  - 3.2.5 Convergence conditions . . . 65
  - 3.2.6 Initial population size determination . . . 68
- 3.3 The Localization Method . . . 77

4 3D Mapping **81**
4.1 Mapping Method 85
4.2 Registration: Scan Matching 90
4.2.1 Preprocessing: data reduction 91
4.2.2 Acceleration using k-d trees 92
4.2.3 Fitness function 96
4.2.4 DE-based scan matching (6 DOF) 100
4.2.5 Simplified version: plane scan matching 103
4.2.6 Scan matching associated difficulties 105
4.3 Loop Detection 107
4.3.1 Surface features 107
4.3.2 Numerical features 109
4.3.3 Expression analysis 112

5 Experimental Results **115**
5.1 Global Localization in 3D Environments 117
5.1.1 Effectiveness of the algorithm 118
5.1.2 Accuracy of the method 121
5.1.3 Thresholding and discarding effect 122
5.1.4 Fitness function options 124
5.1.5 Complexity and computational cost 127
5.1.6 Initial population estimation 129
5.2 Mapping in 3D Environments 136
5.2.1 Data sets 136
5.2.2 Scan matching 136
5.2.3 Loop detection 139

6 Conclusions and Future Developments **145**
6.1 Global Localization 147
6.2 Scan Matching 148
6.3 Loop Detection 149
6.4 Future Developments 149

A Experimental Platform **151**
A.1 Mechanical Design - Robot Structure 154
A.2 Sensory System 160
A.3 Control System 164
A.4 Software 166
A.4.1 MATLAB 166
A.4.2 ROS 167

Bibliography **169**

List of Tables

1.1 Historic robots. 4
1.2 Well-known robots. 5

3.1 Different crossover operations. 55

5.1 Errors localizing the robot at different points. 121
5.2 L1-norm and L2-norm position error. 127
5.3 Computation time for a fixed number of iterations. 128
5.4 Computation time for a fixed population of 50 elements. 128
5.5 Initial population estimates, large environment. 131
5.6 Initial population estimates, medium-size environment. 132
5.7 Highest PD with minimum PFA, *max*. 142
5.8 Highest PD with minimum PFA, *mean*. 142
5.9 Highest PD with minimum PFA, average of *max*. 143

A.1 MANFRED-2's weight. 156
A.2 SICK PLS technical characteristics. 161

List of Figures

1.1 Different robots 6
1.2 MANFRED-2 8

2.1 Topological and metric maps. 15
2.2 Planar map. 18
2.3 6D SLAM by Nüchter *et al.* 22
2.4 6D SLAM by Borrmann *et al.* 23
2.5 Elevation and MLS maps by Triebel *et al.* 23
2.6 VRML model using 2D laser and panoramic camera by Biber *et al.* . 24
2.7 Optimization-based global localization by Moreno *et al.* 27
2.8 ICP scan matching by Nieto *et al.* 30
2.9 IDC scan matching by Lu and Milios. 31
2.10 FAB-SLAM by Cummings and Newman. 34
2.11 Loop detection using the NDT by Magnusson *et al.* 35
2.12 Loop detection by Magnusson *et al.* 36
2.13 Loop detection by Bosse and Roberts. 37

3.1 Corridor of the UC3M and laser reading. 42
3.2 Approximations of probability density functions. 45
3.3 New population member generation for a single candidate. 53
3.4 Simulated environment and laser scan from the true pose. 62
3.5 Three consecutive laser beams over a grid map. 71
3.6 Overlapping between two consecutive laser beams. 73
3.7 Discontinuity effect on the fitness function minimum. 75

4.1 General mapping process. 85
4.2 3D reading obtained by MANFRED-2 86
4.3 Odometry model. 88
4.4 2D *k*-d tree. 93
4.5 3D *k*-d tree. 94
4.6 ICP flow chart. 97
4.7 Scan matching example with 2D data. 104

4.8 Scan matching example with 3D data. 105
4.9 Scan matching associated difficulties. 106

5.1 3D simulated map. 117
5.2 Example of localization in a corridor. 119
5.3 Steps to localization vs. iterations/step. 120
5.4 Error vs. sensor noise level. 123
5.5 Best fitness evolution value vs. iterations. 123
5.6 Algorithm solution in a symmetrical environment. 125
5.7 Best fitness function value vs. iterations. Noisy case. 126
5.8 Map of the environment (large area). 130
5.9 Map of the environment (medium-size map). 132
5.10 Probability of a correct global localization vs. population size. 134
5.11 Scan Matching. Different local maps obtained after registration. . . . 137
5.12 Scan Matching. Map of an area composed of 500 laser scans. 138
5.13 Loop detection matrix between 375 consecutive scans. 140
5.14 Probability histogram between 150 different scans. 141
5.15 PD and PFA as a function of t_1 144

A.1 MANFRED-2, mobile manipulator with robotic arm. 154
A.2 MANFRED-2, lateral view. 157
A.3 Power supply system. 158
A.4 LWR-UC3M-1(robotic arm). 159
A.5 Hokuyo UTM-30LX (laser range finder). 160
A.6 SICK PLS (laser range finder). 161
A.7 Color cameras. 162
A.8 Time-of-flight camera: Kinect. 163
A.9 JR3 67M25A-U560 (force/torque sensor). 163
A.10 PMAC2-PCI (controller card). 164
A.11 ACC-8E. Interface between the PMAC2-PCI and the devices. 165

Chapter 1

Introduction

Isaac Asimov was the first writer that used the word *robotics*. It was in his short story “Liar!”, published in 1941 in *Astounding Science Fiction*. However, Asimov did not know that he was inventing the term. He assumed that robotics was the science and technology of robots because electronics was the science and technology of electrical devices.

Isaac Asimov (born Isaak Yudovich Ozimov, Petróvichi, USSR, January 2, 1920 - New York, April 6, 1992) is one of the fathers of science fiction. He was a prolific writer with more than 500 books, but also a professor of biochemistry in Boston University. His informative science books were also very popular. He was especially interested in the future and the planetary exploration, issues that he considered closely related to robotics. In 1942, he formulated his Three Laws of Robotics in the short story “Runaround”:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.

3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

It is amazing to think about how these thoughts of a science fiction writer of the early twentieth century are closely related to the current research topics in this area.

The first person who used the word *robot* was Karel Capek. He was a Czech writer, also considered one of the founders of science fiction. He wrote about a factory that manufactured artificial people called *robots* in his play “R.U.R.” (*Rossum's Universal Robots*, 1920). Nevertheless, Karel Capek declared that the man who actually invented the word was his brother Josef Capek.

This science was included into a science fiction scope in the early twentieth century because the level of technological development was too low. However, it is not true today. Robotics is an important research area where important resources are invested in research and development.

Robotics can be defined as the branch of technology that deals with the design, construction, operation, structural disposition, manufacture, and application of robots (Oxford Dictionaries[1]). It is related to other sciences such as electronics, engineering, mechanics, mathematics, physics, and software.

William Grey Walter developed the first first electronic autonomous robot between 1948 and 1949. His first robots, Elmer and Elsie, are often described as tortoises because they looked like these animals and their movement was slow. They were

[1]http://oxforddictionaries.com/definition/robotics

capable of finding their way to a recharging station when the battery power was low. Willam Grey's work inspired subsequent generations of robotics researchers. The first digitally operated and programmable robot installed in the industry was the Unimate (1961). Its task consisted of lifting hot metal pieces from a die-casting machine and stacking them. In Table 1.1, the reader can see some historic examples of robots.

More recently, important advances have been made in the robotics research area. Some of the most important examples are presented in Table 1.2.

An illustrative example of robots that have been designed by multiple research groups is shown in Figure 1.1.

The number of industrial and commercial robots has increased substantially in recent years. They can execute specific tasks more cheaply than human beings or can work with high accuracy and reliability. They are also useful tools to be used en dangerous environments which are not safe for human beings. Robots are used in multiple areas: earth and space exploration, automotive industry, manufacturing, transport, surgery, military industry, education research, etc.

Table 1.1: Historic robots.

Inventor (Year)	Name	Characteristics
Heron of Alexandria and others (First century AD)	-	Descriptions of more than 100 machines and automata, in *Pneumatica and Automata*
Al-Jazari (1206)	-	Humanoid automata
Leonardo da Vinci (1495)	Mechanical knight	Designs for a humanoid robot
Jacques de Vaucanson (1738)	Digesting Duck	Mechanical duck that was able to eat, flap its wings, and excrete
Nikola Tesla (1898)	Teleautomaton	First radio-controlled vessel
Westinghouse Electric Corporation (1930s)	Elektro	Humanoid robot exhibited at the 1939 and 1940 World's Fairs
William Grey Walter (1948)	Elsie and Elmer	Simple robots exhibiting biological behaviors
George Devol (1961)	Unimate	First installed industrial robot
KUKA Robot Group (1973)	Famulus	First industrial robot with six electromechanically driven axes
Victor Scheinman (1975)	PUMA	Programmable universal manipulation arm. Unimation company

The countries with more operational stock of multipurpose industrial robots in 2010 (number of robots between parentheses) are enumerated in the following list

Table 1.2: Well-known robots.

Inventor (Year)	Name	Characteristics
Hiroshi Makino (1978)	SCARA	Robotic arm
Waseda University (1984)	Wabot-2	Capable of playing the organ
MIT (1989)	Genghis	Hexapodal robot
Carnegie Mellon University (1994)	Dante	Dante II entered Mt. Spurr and sampled the gases within the volcano
David Barret, MIT (1996)	RoboTuna	Designed to swim
Dr. John Adler (1994)	Cyberknife	Surgery robot
Honda (1996)	P2	Humanoid robot
Mars Pathfinder mission (1997)	Soujouner	Semi-autonomous operations on the surface of Mars
Sony (1999)	AIBO	Robotic dog capable of interacting with humans
Honda (2000)	ASIMO	Most advanced result of their humanoid project
MDA Space Missions (2001)	Canadarm2	It was launched into orbit and attached to the International Space Station
Northrop Grumman (2001)	UAV Global Hawk	First autonomous non-stop flight over the Pacific Ocean
iRobot (2002)	Roomba	Robotic vacuum cleaner
NASA (2003)	Spirit and Opportunity	They landed on the surface of Mars. Opportunity is still operating
Carnegie Mellon University (2005)	Stanley	It won the 2005 DARPA Grand Challenge
Cornell University (2006)	Starfish	4-legged robot. Self modeling and learning to walk after having been damaged

[2]: Japan (285800), North America (173174), Germany (148195), Republic of Korea (101080), Italy (62378), China (52290), France (34495), Spain (28868), and Taiwan (26896).

After some ramblings and history, we are going to address the problem that has been studied in this work.

There is not a unique definition for robot. According to the Oxford Dictionary[3], a robot is a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer. An autonomous robot can be defined as a robot that can perform specific tasks in unstructured environments without continuous human guidance. There are many kinds of robots with different types and levels of autonomy. Besides, the level of autonomy needed by a robot depends on the field and its tasks.

[2]`http://www.worldrobotics.org`

[3]`http://oxforddictionaries.com/definition/robot`

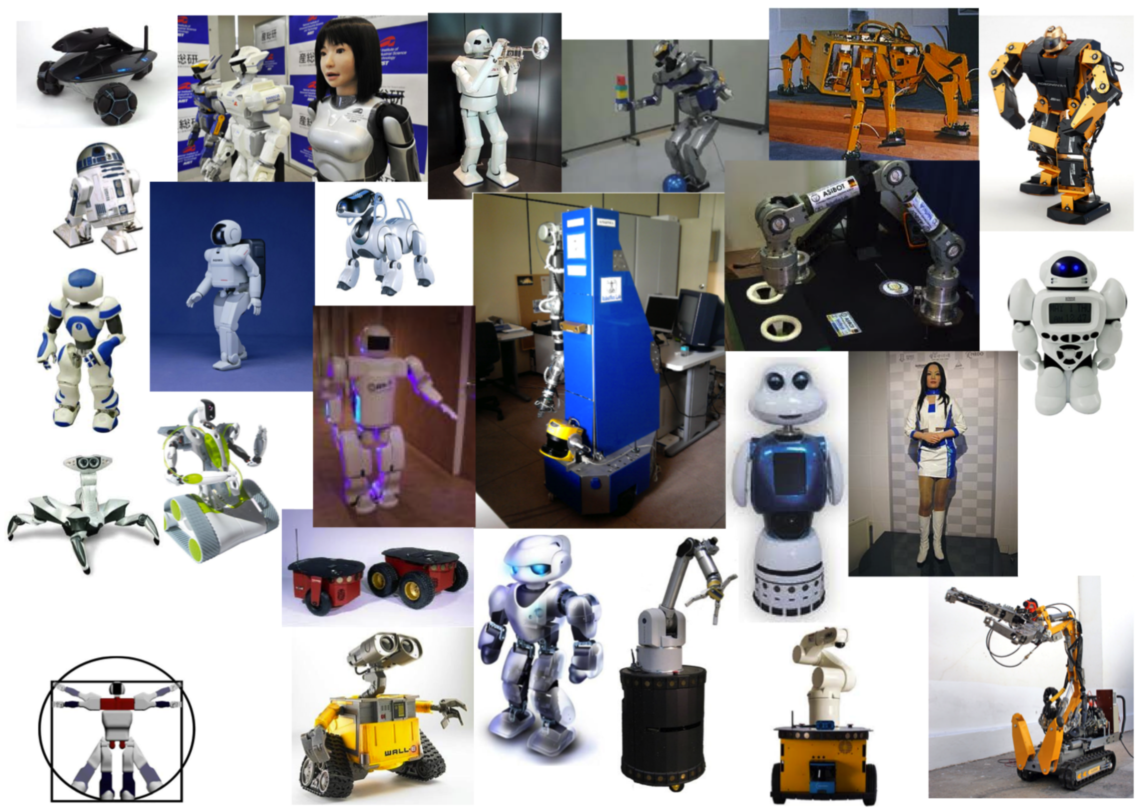

Figure 1.1: Different robots.

A fully autonomous robot must have the following abilities: obtain information about the environment, work for a long period without human intervention, move throughout the environment without human assistance, avoid dangerous situations, and maintain its own survival without breaking the previous rules. It may also learn how to execute new tasks that are not included in the initial configuration.

It is very challenging to design a fully autonomous robot. A useful simplification that is usually assumed is to consider specific workspaces. However, the problem is still a difficult task because the variables that must be considered are often complex, chaotic, and unpredictable. The final purpose is to design a robot capable of working in a robust, reliable, accurate, and safe way.

The mobile robot MANFRED-2 (Figure 1.2) has been fully developed by the Robotics Lab research group of the Systems Engineering and Automation Department of the Carlos III University of Madrid (UC3M). It is an autonomous mobile manipulator designed to operate efficiently in environments where human manipulation capabilities are required. This mobile manipulator uses a sensorial system based on vision, thee-dimensional (3D) laser telemetry, and 3D time-of-flight data (laser and RGB color) to perceive and model 3D environments. It includes navigation, localization, and obstacle avoidance modules in order to execute safely multiple tasks.

As said previously, a fully autonomous robot must obtain information about the environment and learn or gain new capabilities. The mobile robot mapping problem is closely related to these abilities. It consists of interpreting the information perceived by its sensors in order to build map and localize itself in it. There are many other robot skills that depend on this task; thus, it is one of the most important problems to be solved by a truly autonomous robot. Although effective algorithms have been implemented for static, structured, and small or medium-sized environments, it is difficult to be successful in unstructured, dynamic, or large-scale environments.

The objective of this work is to design various specific tools related to the mapping problem in order to improve MANFRED-2 autonomy. We are going to introduce these tools in the following paragraphs.

The localization problem in mobile robotics can be defined as the search of the robot's coordinates in a known environment. If there is no information about the initial location, we are talking about global localization. In this work, we have developed an algorithm that solves this problem in a 3D environment using evolutionary computation concepts (Differential Evolution (DE) [1]). The method has been called RELF-3D and has many features that make it very robust and reliable: thresholding and discarding mechanisms, different cost functions, effective convergence criteria, and so on. The resulting global localization module has been tested in numerous experiments and the most important improvement obtained is the accuracy of the method, allowing its application in manipulation tasks.

If the information is given by laser readings, it is essential to correct the local

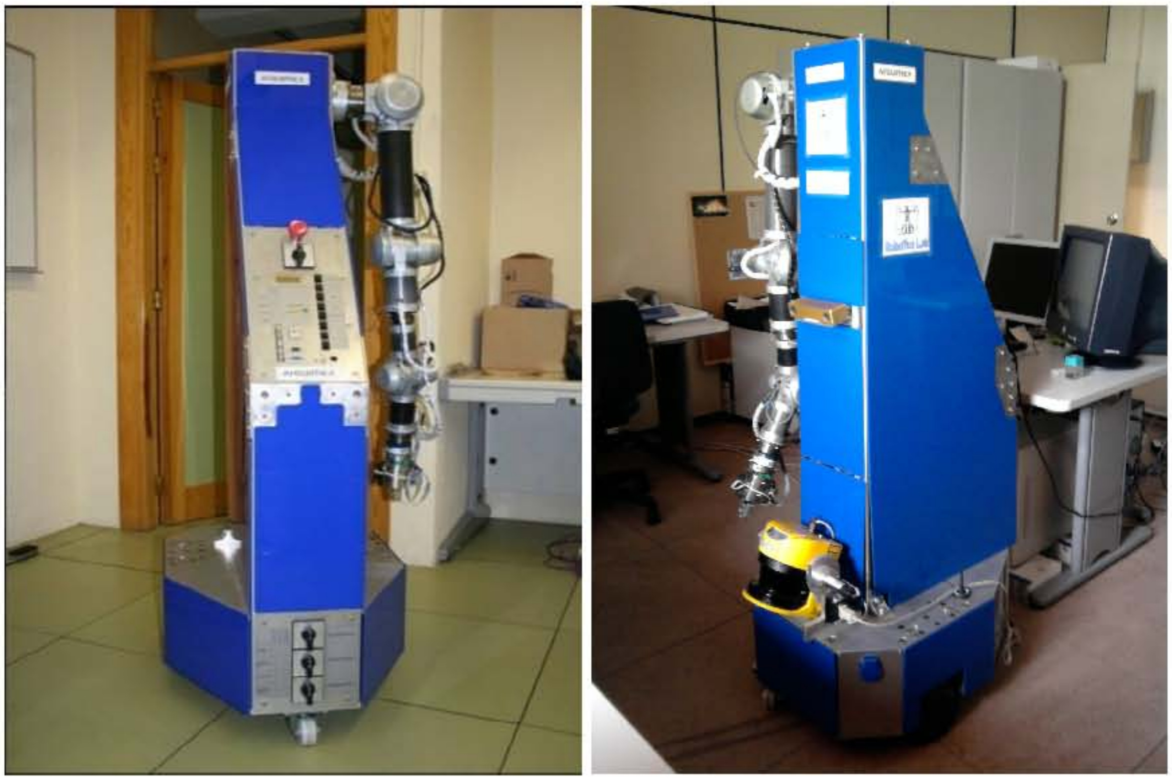

Figure 1.2: MANFRED-2: mobile manipulator developed by the Robotics Lab of the UC3M.

errors between pairs of scans to improve the map quality (our robot uses a 3D laser range finder), which is called registration or scan matching. We have implemented a scan matching algorithm for 3D environments. It is based on the DE method, which is a particle-based evolutionary algorithm that evolves in time to the solution that yields the cost function lowest value. If the cost function is properly chosen, it is possible to solve the scan matching problem using this method. The high accuracy and computational efficiency of the proposed method have been demonstrated with experimental results.

The last problem addressed in this work consists of detecting when the robot is navigating through a known place. This task is often called loop detection. After that, the accumulated error can be minimized to give consistency to the global map, which is called loop closure. We have developed a loop detection method that compares features extracted from two different scans to obtain a loop indicator. It has ben called LPI. This approach allows the introduction of very different characteristics in the descriptor. First, the *surface* features include the geometric forms of the scan (lines, planes, and spheres). Second, the *numerical* features describe several numerical properties: volume, average range, curvature, and so on. The algorithm has been tested with real data to demonstrate that it is an efficient tool to be used in mapping problems. The results are similar or even better than those obtained by other research groups. The introduction of different types of features and weights in the comparison formula, and the uncertainty band improve the algorithm performance and make it a more versatile method because it admits different settings.

This document is organized as follows. The state of the art is reviewed in Chapter 2. In Chapter 3, the global localization method is explained. The scan matching and loop detection methods are presented in Chapter 4. The experimental results are shown in Chapter 5 and, finally, the most important conclusions and future developments are summarized in Chapter 6. The experimental platform MANDFRED-2 is presented in Appendix A .

Chapter 2

State of the Art

Three different methods that solve three different problems are the main contributions of this document. These problems are global localization, registration via scan matching, and loop detection. The global localization problem can be defined as the search of the robot's coordinates relative to its environment, assuming that it is provided by a map and there is no knowledge about the robot's initial position. The registration or scan matching consists of estimating the metric relation between different scans in order to correct the robot's pose. The laser scan matching (the perceptive sensor is a laser scan) consists of the current scan pose correction until the best overlap with the reference scan or model is achieved. It is basic to detect when the robot is navigating through a previously visited place, which is usually referred to as loop detection.

All these concepts can be included in the mapping or Simultaneous Localization And Mapping (SLAM) problem. A typical statement assumed when designing an autonomous robot is that there is no initial knowledge about the environment and the map is gradually discovered during navigation. The perceptive sensor measurements that are continuously received by the robot must be therefore related to previously perceived ones and integrated in a consistent way. The result is a map representing the environment characteristics. The relation between the newly acquired measurements and the already built model must be known. In other words, the robot's location (localization problem) in the map must be known in order to incorporate new measurements when exploring the environment. Therefore, incremental mapping and localization are connected concepts that can be studied together. This problem, which is one of the biggest challenges in our field, is called SLAM.

The SLAM problem was originally developed by Leonard and Durrant-White [2] basing on an earlier work by Smith *et al.* [3]. It consists of building a map of an unknown environment while navigating through it at the same time, using this map. The mapping is closely related to the SLAM problem because map-learning cannot be separated from localization.

The state of the art of the topics studied in this thesis is discussed in this chapter. Since all of them are related to the SLAM problem, this is the starting point from which we start.

The robot has to interpret the information perceived by its sensors in order to build map and localize itself in it. Despite significant progress in this area during the last decades, it is still an open problem with great challenges.

The first question to be asked in robotic mapping is how complex the environment representation must be regarding the available information. The map complexity can be higher or lower depending on the robot purposes. For example, some blind insects do not have a spatial representation of the world, but they survive because they have developed a triggered response to the events that they perceive. However, it is not the case of autonomous robots because they need a more complex representation of the environment to successfully execute their tasks. One possibility is to build a map composed of the information obtained by the robot's sensors from different locations. This information must be merged into a global coordinate system. The environment computation process of the human beings is commonly referred to as cognitive map. A more complex representation is based on the construction of cognitive maps. These maps allow the execution of different tasks (path planning, obstacle avoidance, events memorization, danger evaluation) depending on the map information.

The path planning is a basic problem that is closely linked to mapping. It consists of computing the best path from the starting point A to the final point B. The initial and final locations can be given by coordinates or features depending on the map type. The path planning algorithms are measured by different variables, such as: time, computational complexity, distance, or energetic efficiency. The environment characteristics must be considered when calculating an optimal path; thus, an efficient motion requires a high knowledge about the environment. In general, if the mapping algorithms provide us with efficient and robust maps, the possibility of finding optimal paths will increase.

The information obtained by a mobile robot can be provided by *idiothetic* or *allothetic* sources. The word *idiothetic* literally means self-proposition, while the word *allothetic* means being centered in people or places other than oneself. In other words, this information is usually provided by two types of sensors: proprioceptive sensors, such as wheel encoders with motion information, and exteroceptive sensors, such as laser range finders receiving environment information.

When the robot is under motion, it can measure its absolute position by different methods. One option is to track the number of revolutions of its wheels. It uses an *idiothetic* source with internal information. However, this information is strongly influenced by a cumulative error that can grow quickly. The *allothetic* sources obtain external information. This information is given by perceptive sensors such as cameras, laser range finders, and sonars. The problem here is that two different places can be perceived as the same (*perceptual aliasing*). It occurs in some cases where it is not possible to distinguish between different locations considering only the external information given by the exteroceptive sensors. For example, all the corridors may look the same in an office building.

It is possible to classify the map as metric or topological depending on the internal representation of the map:

- Metric map: it considers a n-dimensional space where the objects are placed with precise coordinates. It is the most common representation for humans, but it can be influenced by noise and it can be difficult to calculate the distances accurately. These maps are addressed in this work.

- Topological map: it is based on the relations between places. The distances

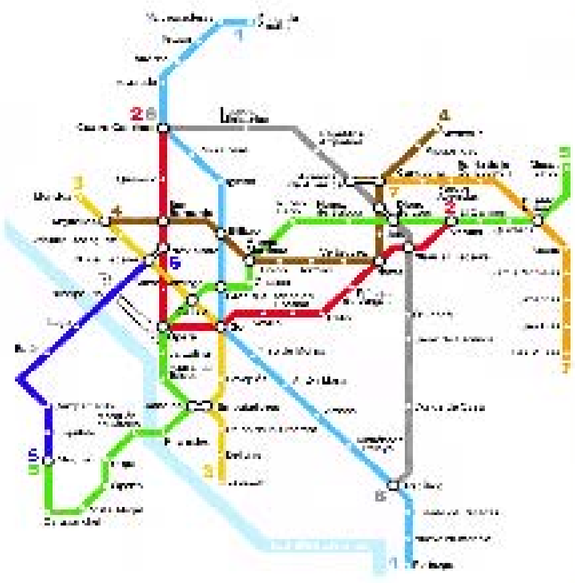
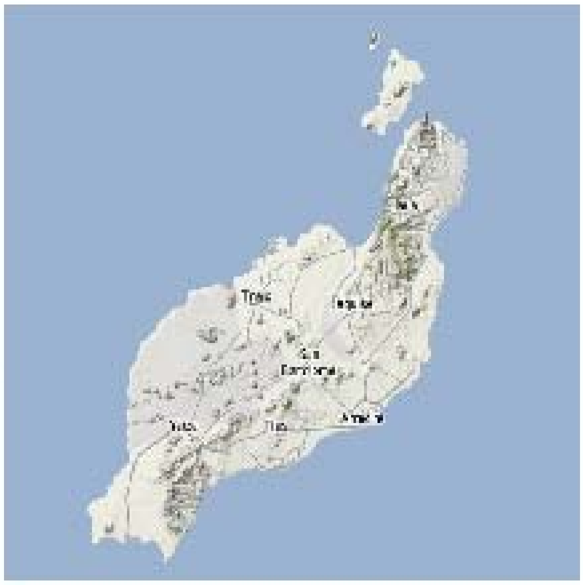

Figure 2.1: Madrid metro topological map and Lanzarote Island metric map.

between a set of different places are stored. The map is composed of a graph with nodes (places) and arcs (paths).

An example of both maps is shown in Figure 2.1. The Madrid metro map is composed of the underground stations (nodes) and the connections between them (paths). The Lanzarote Island map contains a two-dimensional (2D) spatial model.

Many research groups have studied the mapping problem from a probabilistic point of view [4] in order to deal with the uncertainty. For example, the map can be divided into grids where each cell has a different occupancy probability. The map is composed of three types of cells: free spaces, objects, and unknown places (high uncertainty). This representation makes it possible to change the map resolution when more accuracy is required.

According to the handbook of Siciliano and Khatib [5], the SLAM problem can be divided into two different approaches. The first one is called *full* SLAM problem and it is based on the estimation of the posterior probability of the whole path of the robot together with the map:

$$p(X_t, m|Z_t, U_t), \tag{2.1}$$

where X_t is the path up to time t; m is the map; $Z_t = \{z_1, ..., z_t\}$ contains the perceptive sensors measurements (external information); and $U_t = \{u_1, ..., u_t\}$ contains the motion information. It can be noticed that the variables of the right part are all available. It is necessary to estimate X_t and m.

The second one is the *online* SLAM problem. It consists of the estimation of the current location of the robot:

$$p(x_t, m|Z_t, U_t), \tag{2.2}$$

where x_t is the robot's pose (robot position and orientation) at the present time, not being necessary to estimate the whole path.

As can be observed in Equations 2.1 and 2.2, the robot needs two models in order to successfully accomplish this task. The first one is a mathematical model that converts wheel displacements or odometry[4] information (u_t) into robot's locations (x_t), and the second one is a mathematical model that relates the sensor measurements (z_t) to the robot's locations and the map (m).

The mapping problem has been studied in the past considering a 2D world due to the available sensors and the simplicity of this approach. However, the recent developments make it possible to obtain a 3D representation suitable for multiple applications. The computational cost when building a 3D world has been traditionally considered as a drawback. However, there are now 3D sensors and powerful computers that minimize this shortcoming. Besides, the model is much richer because it contains a larger amount of information. Every aspect covered in this thesis is related to 3D environments. Therefore, most of the related works cited in this chapter will focus on this type of maps.

As said previously, various concepts related to the mapping problem have been addressed in this document. An overview of the most significant SLAM algorithms is given in Section 2.1.

We are talking about global localization when the map of the environment is known but there is no *a priori* information about the robot's pose. We have developed an algorithm that solves this problem obtaining a high accuracy. The global localization methods are reviewed in Section 2.2.

The mapping process has been divided here into several points that represent the tasks that should be solved to generate a consistent map. These tasks are the following ones:

1. Robot's pose estimation: the robot's pose with respect to the last one must be computed, which means that the motion information has to be updated. An odometry model is used to compute the robot's pose according to the motion information given by the wheel encoders. It is well-known that these measurements eventually diverge. The accumulated error is specially important when

[4]Odometry can be defined as the use of data from moving sensors to estimate change in position over time.

the robot is located far away from the initial position. Hence, the accumulated error should be corrected by other methods.

2. Pose correction via scan matching: the accumulated motion error when only considering odometry information has to be corrected using the perceptive sensor information. The pose is corrected by matching the last acquired scan against the previous one. The scan matching methods can be regarded as local localization methods because they compare single scans. A detailed review of these methods is presented in Section 2.3.

3. Loop detection and loop closure: the model obtained is not consistent after registration because the accumulated error due to local small errors can be very important. If the robot can detect when it is navigating through a previously visited place, it will be possible to minimize this error. The global map will be more consistent. The detection task is usually referred to as loop detection, and the global error minimization is called loop closure. The loop detection methods related to our approach are described in Section 2.4.

   There are several algorithms that correct the pose errors after registration [6, 7]. They distribute the accumulated pose error of pairwise registered scans in order to build a consistent map once the robot has detected that it is in a pre-visited place. However, the loop detection task is still an open problem in robotics. That is the reason why this topic has been studied in this thesis.

A more detailed explanation of the mapping method will be given in Chapter 4.

2.1 Mapping/SLAM

Mapping or SLAM is a very common topic in mobile robotics. It is very difficult to review this subject without forgetting important contributions. Since this work focuses on 3D environments, one possibility is to categorize the mapping algorithms into subgroups depending on the map dimensions and the robot's pose Degrees of Freedom (DOF).

Depending on the map dimensions, there are planar (2D) and volumetric (3D) maps. Besides, the SLAM approaches can be classified by the robot's pose DOF. A pose estimate in a 2D world contains three parameters that represent the robot's position (x, y) and orientation (θ). An example of this type of map is shown in Figure 2.2. If the environment is 3D, it can be necessary to define six different parameters: the x, y, z coordinates, and the the roll, yaw, pitch angles. However, there are different techniques to build volumetric maps considering the available sensors and the DOF of the localization method:

- Planar localization method (3 DOF) combined with 2D perceptive sensors from different angles.
- Six-dimensional (6D) pose estimate combined with 2D perceptive sensor.

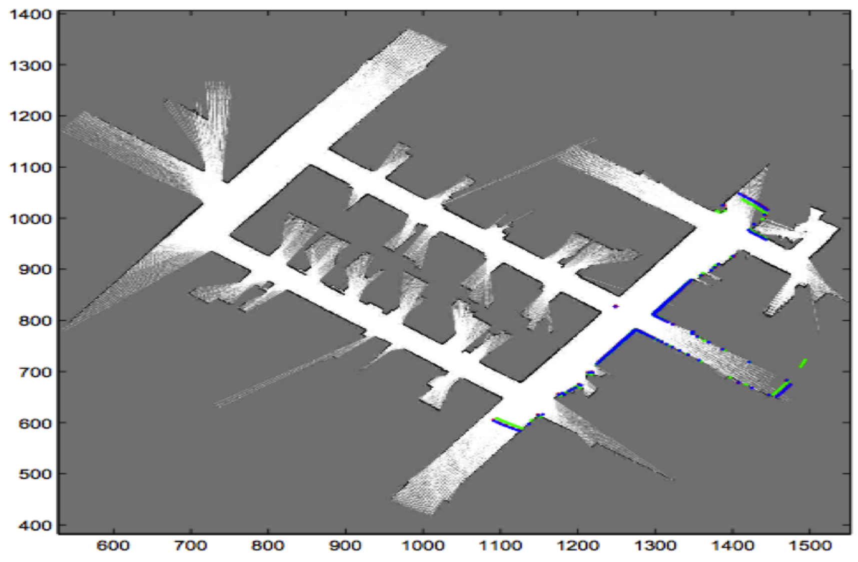

Figure 2.2: Example of planar map obtained by Moreno *et al.* [8].

- 6D localization method combined with 3D perceptive sensor.

All these methods are described in the following sections.

2.1.1 Planar mapping

Planar metric maps are often built using probabilistic techniques where there are two sources of information: probabilistic motion and environmental perception with uncertainty. It is possible to localize the robot by integrating both components. The first approaches were carried out by Smith *et al.* [3] and Leonard and Durrant-Whyte [2].

The first question to be solved is how to estimate the robot's pose. After that, the map can be built. The consistency of the global map can be increased by detecting known places. However, there is no guarantee of a correct model. There are different techniques that solve the SLAM problem (a complete review can be found in the work by Thrun [9]):

- Maximum Likelihood Estimation (MLE) [10, 11]:

  It is a method for estimating the parameters of a statistical model according to a fixed set of observations. It assumes that the robot's pose can be given by a statistical model. In general, the MLE method obtains the model parameters that maximize the likelihood function. These parameters define a probability distribution. In other words, the greatest probability is given to the observed data. It can be very useful when the model can be defined by well-known distributions such as the normal one. However, this method present difficulties when the probability distribution is unsuitable or does not exist.

- Expectation Maximization (EM) [12]:

  The EM algorithm is an iterative method proposed by Dempster *et al.* [13] that finds maximum likelihood estimates of parameters in statistical models. The method is composed of two steps. First, the Expectation (E) step computes the expectation of the log-likelihood evaluated using the current estimates. After that, the Maximization (M) step estimates the parameters that maximize the expected log-likelihood found in the E step. These parameters will be used again in the next iteration E step.

- Extended Kalman Filter (EKF) [14]:

  The EKF is the nonlinear version of the Kalman filter which linearizes about the current mean and covariance. The EKF has been considered the most common method in the theory of nonlinear state estimation. Most SLAM algorithms are based on EKFs.

A Kalman filter [15] is an optimal recursive data processing algorithm. It integrates all information that can be provided to it. The Kalman filter computes all available measurements (with uncertainty) to estimate the current value of the unknown variables of interest.

- Sparse Extended Information Filter (SEIF) [16]:

  The Extended Information Filter (EIF) is the dual of the EKF. Both algorithms are computationally equivalent, but the EKF considers the covariance matrix while the EIF considers the inverse covariance matrix, also known as information matrix. The SEIF is a sparse variant of the EIF. The maps are represented by graphical networks of features that are locally interconnected. The relations between pairs of neighbor features and the robot's pose relative to the map are stored in the links between features.

There are also other approaches that are important in planar mapping. The FastSLAM [17] method approximates the posterior probabilities (robot's poses) by particles. The method proposed by Lu and Milios [18] is based on the on the Iterative Dual Correspondence (IDC) scan matching. This type of maps was also addressed by our research group in the past [8]. We applied DE to planar SLAM.

2.1.2 Volumetric mapping

Planar localization method (3 DOF) combined with 2D perceptive sensors from different angles

Some research groups have simplified the volumetric mapping by using 2D laser range finders instead of 3D scanners. If the 2D scanner obtains information from different angles, it is possible to obtain volumetric maps.

The first possibility is to mount one scanner horizontally and the other one vertically. The second one obtains information in a vertical plane that is transformed into 3D coordinates according to the robot's pose that is computed depending on the horizontal scan. The map accuracy depends on the robot's pose estimation. This method has been applied by Thrun *et al.* [17] and Früh and Zakhor [19]. Zhao and Shibasaki [20] have used two additional scanners shifted by 45° to reduce occlusions.

Other groups have developed rotating scanners for the same purpose. Although they obtain complete information about the 3D world, they do not consider all the six DOF in localization because the robot's pose is obtained by sensors that use three DOF (x, y, θ). Wulf *et al.* [21] utilize a scanner that rotates around the vertical axis. The 3D laser data are acquired while moving. The pose estimate is given by inertial sensors, i.e. gyros.

The algorithms developed in this work can use six DOF at maximum. However, the map obtained by the experimental platform MANFRED-2 can be included into this type of maps because the robot's pose is given by three coordinates (x, y, θ). Therefore, all the experiments that use our robot will be simplified considering the available information. This platform is described in Appendix A. The motor mounted laser scanner is shown in Figure A.6. We have also included other experiments with different data sets where the robot's pose is given by six DOF.

6D pose estimate combined with 2D perceptive sensor

A different option consists of utilizing 2D laser scanners and 6D pose estimation algorithms. This is sometimes used in mobile robot navigation systems. The Stanford Racing Team [22] used this technique for high speed terrain classification in the DARPA Grand Challenge, which is a race for driverless vehicles organized by the Defense Advanced Research Projects Agency (research organization of the United States Department of Defense). The participating vehicles must travel between two different points of the United States (the first edition covered 240 km) without human intervention. The location of the vehicle must be accurately estimated with expensive sensors.

6D localization method combined with 3D perceptive sensor

There are multiple research groups that focus on the mapping problem considering that the robot is in 3D environments with six DOF. This scene holds more information than the 2D world. This strategy is appropriate for several tasks such as object manipulation or grasping and robot motion and localization on natural surfaces. Due to the available sensors and the increase of the computer computational capabilities, this procedure must play an important role in SLAM for mobile robots.

It is important to distinguish between outdoor and indoor maps because there are sensors, e.g. the GPS (Global Positioning System), that provide the mobile robot with a precise measurement of its pose when it is located outdoors. This is not an easy task when the mobile robot is located in an indoor environment. The measurements are not accurate, so additional mechanisms should be implemented to improve the map quality.

Nuchter *et al.* [23] have developed a mapping method based on the alignment of 3D laser range scans using the Iterative Closest Points (ICP) [24] scan matching method combined with a heuristic for loop detection and a global relaxation method. Furthermore, they accelerate the data association process using k-d trees, which is a technique that have also been applied to our methods and will be explained later in this document. They have also implemented a data association system called "Cached

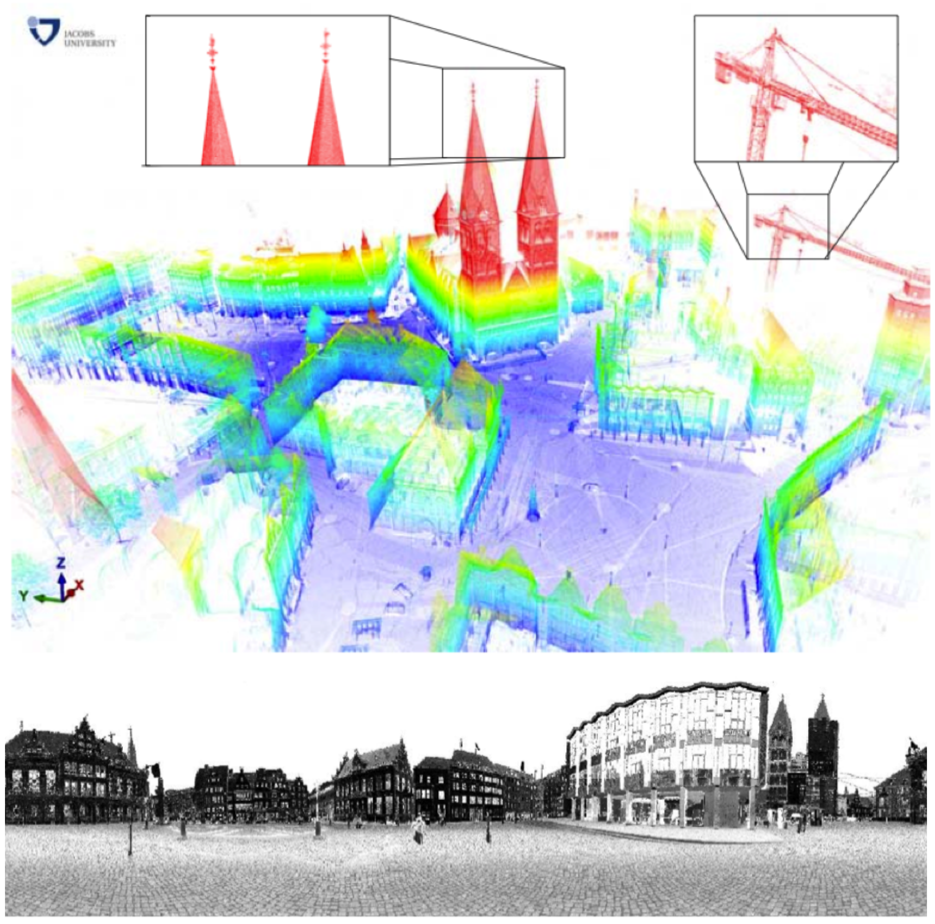

Figure 2.3: 6D SLAM of an outdoor environment by Nüchter *et al.* [25].

k-d tree Search". Their recent work includes a city map [25, 26]. As can be observed in Figures 2.3 and 2.4, they obtain impressive results.

Triebel *et al.* [27] work with 3D maps building Multi-Level Surface (MLS) maps. An elevation map stores the height of the corresponding area in each cell of a discrete grid. The main shortcoming of this map is that it is not possible to represent vertical structures with multiple levels. The MLS maps allow the storage of multiple surfaces in each cell, making it possible to include bridges or buildings in the map. Their experiments demonstrate that their method is convenient for representing large-scale outdoor environments. Their results can be seen in Figure 2.5, where it is possible to distinguish between elevation and MLS maps.

Cole *et al.* [28] demonstrate that the traditional methods used to solve the SLAM

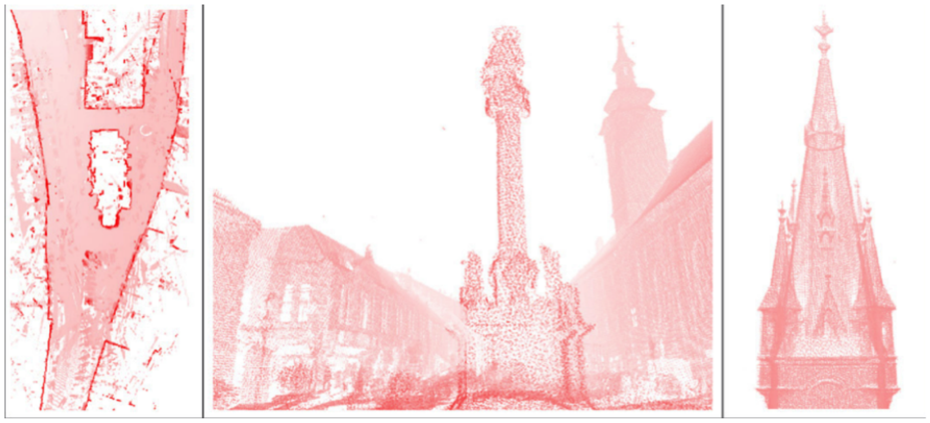
Figure 2.4: 6D SLAM of an outdoor environment by Borrmann *et al.* [26].

problem in planar environments can also be extended to perform 6D SLAM in more difficult conditions, e.g. undulating outdoor areas. Their technique is based on the registration of scans that are obtained from different places.

Sequeira *et al.* [29] have worked in the RESOLVE project to model interiors for virtual reality and tele-presence. They have used a laser range finder called RIEGL. The scan matching method adopted is the ICP.

Allen *et al.* [30] have worked in the AVENUE project for modeling urban environments. The scanner that they have used is called CYRAX. They have designed a feature-based scan matching approach for registering the 3D scans. However, in their

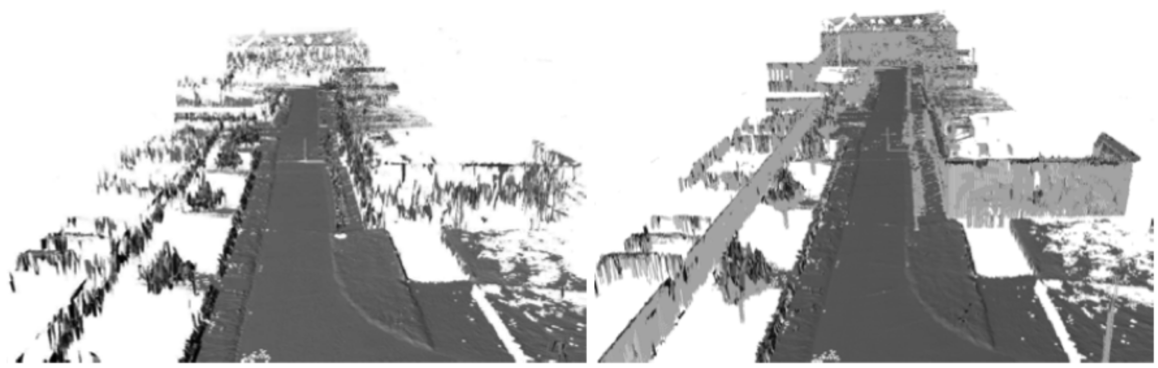
Figure 2.5: Elevation map (left) and MLS map (right) obtained by Triebel *et al.* [27].

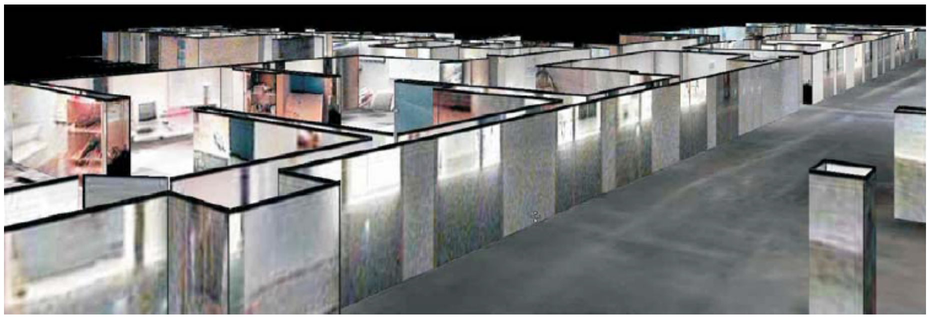

Figure 2.6: VRML model using 2D laser and panoramic camera by Biber *et al.* [35].

recent work the localization system does not make use of laser data [31].

Hebert's group [32] have built maps using the Zoller+Fröhlich laser scanner. They have developed two different methods: surface matching between observations and feature matching between individual features extracted from the observations. They do not use odometry information or initial pose estimates.

Magnusson and Duckett [33] have proposed an alignment method that is based on the Normal Distributions Transform (NDT) [34].

Other approaches

Another way to accomplish this is to use computer vision sensors such as CCD (Charge-Coupled Device) cameras. Their principal disadvantage is that changing light conditions make them difficult to use.

Biber *et al.* [35] have developed a mapping method based on a 2D laser scanner and a panoramic camera. They use the laser scanner to solve the SLAM problem and to extract walls. After that, the walls textures are built from the panoramic camera. The model obtained after being exported to VRML (Virtual Reality Modeling Language) format can be observed in Figure 2.6.

The mobile robot by Se *et al.* [36] uses scale-invariant visual landmarks to localize itself and build a 3D map at the same time. The 3D landmarks are tracked using Kalman filters. The landmarks position and uncertainty are stored in a data base.

2.2 Global Localization

In general, the autonomous robots that need to travel from an initial location to a destination point must solve two fundamental problems: navigation and localization. The robot is correctly localized when it can determine its position and orientation. The navigation skill involves finding a feasible path and avoiding the obstacles. It is straightforward that it is not possible to navigate without the localization module. In robotics, these problems have been among the most important research areas for the last decades.

The global localization systems can be separated into two different types depending on the information source: positioning systems and self-localization systems.

The positioning systems are based on signals emitted by an external source (beacons). These signals are detected by the robot's sensor system. A different possibility is that the robot emits the signal that is captured by one or more sensors located in the environment. These sensors communicate the location to the robot.

The positioning systems use different variants of triangulation methods to estimate the robot's location. You can find different types of positioning systems. The best known is the GPS, which is a global navigation satellite system that returns location and time information where there is an unobstructed line of sight to four or more GPS satellites. It is more difficult to develop positioning systems for indoor environments because a great number of emitters and receivers is required to have a good signal coverage. Systems based on radio-frequencies (Wifi and Zigbee), vision, laser or ultrasonics are obtaining promising results. Their main disadvantage is that they depend strongly on the emitters and receivers positional distribution. Although it is not necessary to have a detailed map, the beacons' coordinates must be known *a priori*.

The self-positioning systems are based on sensor systems implemented in the robot. They do not require external beacons because they use the measurements obtained by the sensors. Some typical examples are localization modules based on laser range finders that obtain 2D or 3D scans. This approach requires a predefined map that is usually more complex than the map of the positioning systems. The self-positioning systems have been studied in this document because it is a more interesting problem from a theoretical point of view and our robot works in indoor places.

The self-positioning associated problems depend on the initial information about the map:

- Global localization and re-localization: there is an *a priori* knowledge about the map.

- SLAM: there is no initial information and the robot is learning the map while navigating at the same time.

The localization problem can be separated into two different types depending on the initial information about the robot's location:

- Re-localization (or tracking) systems: they maintain a reliable estimate of the robot's position and orientation while it is navigating. It is assumed that the initial position is known (with uncertainty) and the tracking algorithm objective is to maintain a reliable estimate while the robot is executing a specific mission. Most of the existing localization algorithms address this problem because it is less complex than global localization. There are algorithms based on the EKF that obtain good solutions. The Kalman filters were initially applied to re-localization during the nineties [37, 38, 39]. These methods are computationally very efficient, but they require a good initialization and a good knowledge of the statistical properties of the system.

- Global localization systems: these modules do not assume any initial knowledge about the initial position and orientation. The search must be done not only in a limited zone, which is sufficient for re-localization, but in the whole map (at least in a large area). This problem is more difficult because the information integration and the mathematical models are more complex.

There are different families of algorithms that can provide a solution to the global localization problem in 2D maps:

- Bayesian-based methods: they operate in two steps. In a first step, movement and perceptive probabilistic information are integrated into the *a posteriori* density function. The estimate is computed in a second step according to a specific criterion such as the maximum density point or the average value. These methods focus on the generation of accurate models for the density function in order to represent the most feasible areas. When the convergence is reached, all the distribution is concentrated in a small area. This group of solutions has been widely studied. Grid-based probabilistic filters [40, 41, 42] and Monte Carlo (MC) localization methods [43, 44] can be included here. Burgard *et al.* [40] have studied the problem assuming that the robot is in a grid map and Fox *et al.* [41] have developed a Markov localization module in dynamic environments.

- Optimization-based methods: all the information is integrated to generate a fitness or cost function that is minimized in each motion-perception cycle. The estimate is the element with best fitness value. The DE [45] and Particle Swarm

Optimization (PSO) [46] filters can be included here. Vahdat *et al.* [46] apply two evolutionary methods (DE and PSO) and compare both of them with MC.

A global localization algorithm for 3D environments that is based on optimization techniques (DE) has been developed here. In our previous work, it was successfully applied to planar environments [45]. An example is shown in Figure 2.7, where an initially generated population evolves to the true solution basing on the optimization of a fitness function.

- Hybrid methods (multi-hypotheses Kalman filters): these methods are not

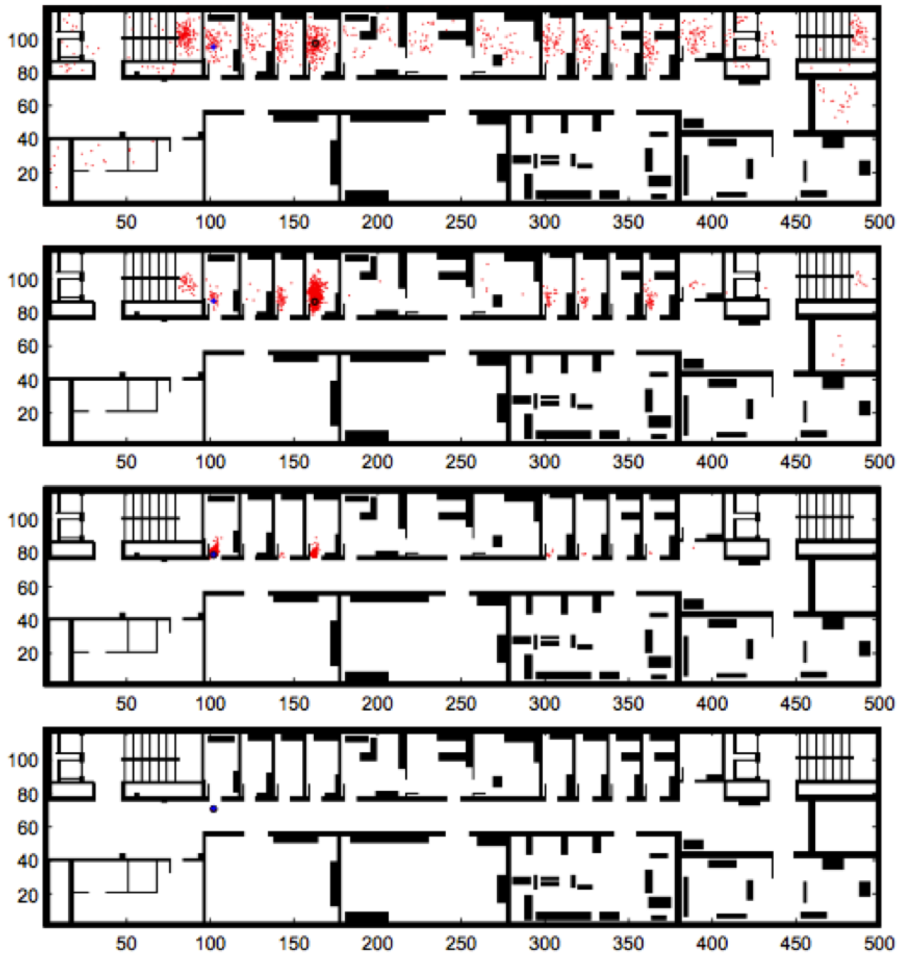

Figure 2.7: Global localization using optimization-based methods (Moreno *et al.* [45]). It is based on DE. Population members in red and best solution in blue.

purely Bayesian. Although they contain a set of solutions composed of normal probability distributions, the generation and elimination of solutions are based not only on probability distributions but on decision trees and geometric constraints. Some examples are [47, 48, 49, 50].

Most of these methods have been applied to 2D environments. If we talk about the localization problem in a 3D map, we find less information in the literature. For example, Kümmerle *et al.* [51] have worked with MC in outdoor applications. They apply a particle filter to estimate the full 6D state of the robot. These authors solve the localization problem in outdoor environments by matching laser range measurements to a given map of the environment using MLS maps.

Lingemann *et al.* [52] have developed an algorithm called HAYAI (High-speed And Yet Accurate Indoor) that localizes the robot matching features between the data set and the model set. Their contributions lie in fast filtering and extraction of natural features in laser scans and a closed-form solution for computing the pose shift. Tsai *et al.* [53] fuse inertial and ultrasonic sensors and apply an EKF-based algorithm to estimate the current posture of a mobile robot navigating over indoor uneven terrain. Lai *et al.* [54] have designed a non-linear method for a system composed of a laser range finder and four artificial reflectors.

There are also many groups that are using computer vision techniques [55, 56, 57, 58, 59]. Se *et al.* [58] consider global localization as a place recognition problem, and solve it by matching the SIFT (Scale-Invariant Feature Transform) [60] features detected in the current frame to the pre-built SIFT database map. More recently, a different descriptor called SURF (Speeded Up Robust Feature) [61] has been also applied to object recognition. Ho *et al.* [56] demonstrate how the robot can perform global localization using a panoramic mirror in conjunction with a rich 3D model of the environment and a particle filter for localization. Royer *et al.* [57] also use a previously built map to compute robot localization in real time using vision algorithms.

Finally, it has to be remarked that the localization problem is a particular aspect of the SLAM problem; thus, all the approaches related to SLAM (Section 2.1) can be enclosed in this section.

2.3 Scan Matching

An important tool that must be developed when building consistent maps is the registration or scan matching module. This is because the pose estimate considering only motion information is not accurate enough and the local error must be minimized comparing the last acquired perceptive measurements to the map. The current scan pose is corrected until the best overlap with the reference scan or model is achieved.

There are different classifications of the scan matching methods. These classifications and some ideas about scan matching are introduced in the next paragraphs.

The first one depends on the environment dimensions. The scan matching methods found in the literature can be developed to work with 2D or 3D data. The last case is addressed in this work. The main difference between them is the computational cost. The computational cost of the algorithms that use 3D data is higher because they use more information.

There are also local [18] and global [62] methods. The local methods match single scans. The disadvantage of this approach is that the final map is inconsistent because the accumulated error can be important. However, its advantage is that this inconsistency can be minimized by relaxation mechanisms. The loop detection scope is included here. It is possible to close the loop by minimizing the accumulated error when it is detected. The global methods consider the current scan and the global model. Their associated shortcoming is that one single mistake is fatal because a wrong measurement can be included in the model, not being possible to correct this error by additional mechanisms.

It is also possible to distinguish between feature-based, point-based, or mixed approaches. The first case requires a feature extraction before the scan matching. The most common features are line segments, planes, or corners. It is very important to choose the correct features because they must be contained in the laser scan; otherwise, the scan matching will not obtain the desired results. The point-based approach does not require any distinguishable structure in the environment and the mixed method seeks the correspondence between points and features. A point-based local method has been developed in this work.

Some examples of point-based scan matching are detailed below:

- Iterative Closest Point (ICP):

  The ICP method [63] is an algorithm that is used to minimize the spatial distance between two scans. The ICP is the most common scan matching method. It is also used to reconstruct planar or volumetric surfaces using multiple scans, to localize robots, etc.

  This method is quite simple and its computational cost makes it possible to use it in real-time. It receives two clouds of points, an initial estimate of the

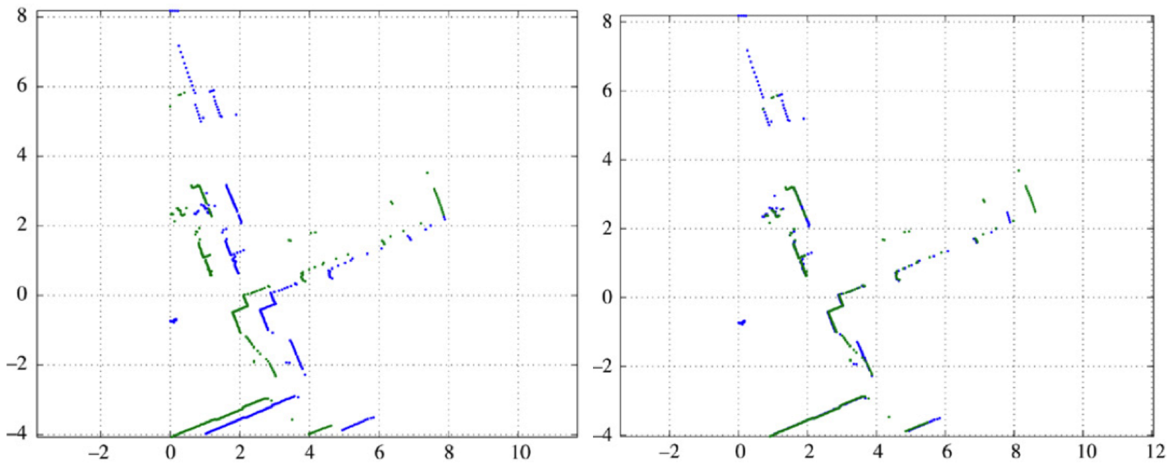

Figure 2.8: ICP scan matching by Nieto *et al.* [64]. Left: initial coordinates of the scans. Right: results after scan matching.

translation and rotation, and the stopping criteria. It iteratively estimates the transformation (translation, rotation) that minimizes the distance between the clouds.

Besl and Mac Kay [24] have developed this method to register 3D shapes. It has been successfully applied to SLAM. An example can be found in the Recursive scan matching SLAM method developed by Nieto *et al.* [64]. An illustration of their results for a pair of scans is shown in Figure 2.8.

Many variants of this method have been proposed. A very interesting comparison of several methods depending on different parameters such as the selection and matching of points and the minimization strategy can be found in the work by Rusinkiewicz *et al.* [65]. Some mechanisms of the registration method that has been developed in this work are based on the ICP variant proposed by Triebel *et al.* [27]. The cost function is changed in order to match points that belong to similar surfaces.

- Iterative Matching Range Point (IMRP):

  The IMPR method, proposed by Lu and Milios [18], is based on the limitation of the maximum translation and rotation. The correspondences are found in a neighborhood of the model, not considering the whole space.

- Iterative Dual Correspondence (IDC):

The IDC method, also proposed by Lu and Milios [18], combines ICP and IMRP. The translation is computed by the ICP method and the rotation is estimated by the IMRP method. Their results are presented in Figure 2.9.

- Polar Scan Matching (PSM):

  The PSM method does not need to find correspondences between points. It assumes that model and data are sorted in the same way and only points with the same bearing are matched. Its advantage is that it is not necessary to transform laser measurements into cartesian coordinates (the original distances can be compared). This advantage is the key of the method proposed by Diosi and Kleeman [66].

A different method was proposed by Weiss and Puttkamer [67]. It starts extracting line segments that connect consecutive points. After that, it generates orientation histograms of the line segments. These histograms are used to estimate the orientation of the current scant with respect to the model scan. The translation is estimated by calculating histograms of the points' cartesian coordinates.

Thrun et. al [17] consider that the free space in the current model will remain free in the future. They increase the information that is extracted from the laser scan.

Biber *et al.* [34] have proposed an alternative representation based on the NDT. The map is divided into cells and each cell is assumed to be defined by a normal distribution that represents the probability of being a laser measurement (occupied cell). The transformed scan is composed of a set of probability distributions that can be used to match another scan. This method does not require the corresponding points computation.

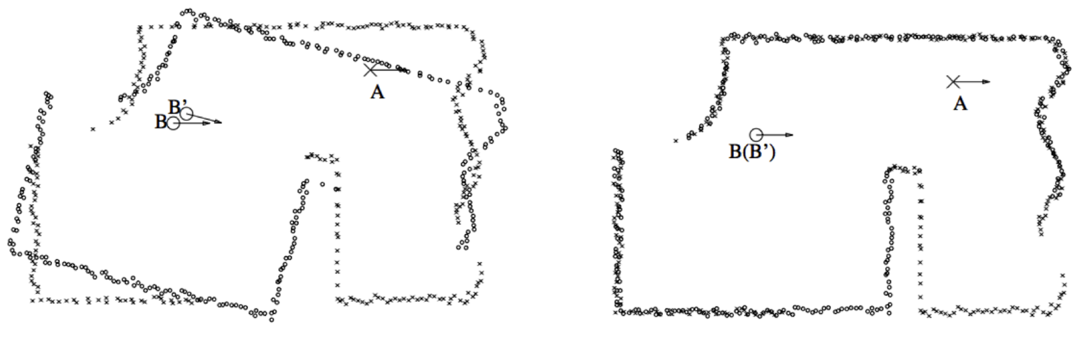

Figure 2.9: IDC scan matching by Lu and Milios [18]. Left: initial coordinates of the scans. Right: results after scan matching.

Another important task is to estimate the quality of the match. Lu and Milios [18] assume that the scan points contain white Gaussian noise. They use this information to estimate the uncertainty of the match. Bengtsson and Baerveldt [68] have developed two different methods based on the calculation of the covariance matrix. They differ in the calculation method.

2.4 Loop Detection

The maps generated after registration are not consistent enough to conclude that the mapping task has been successfully achieved. The small errors in the initial estimation and the matched pairs can cause a huge accumulated error at the end of the robot's path. It is essential to improve the map consistency after registration. One way to do it is to detect when the robot is navigating through a known place. After that, the accumulated error can be minimized or eliminated.

Several feature-based techniques that have been successfully applied to loop detection will be explained here. It is important to say that these techniques must be rotation-invariant because the places have different appearances depending on the robot's orientation.

The sensors that are more commonly used in loop detection are cameras or 2D/3D laser range finders.

Ramos *et al.* [69] have applied feature-based detection to close loops in unstructured, outdoor environments. These features are created from images and scans that are used together. First, the laser scan is used to detect the region of interest. The algorithm can work in real-time because only these regions are processed. The process can be divided into four different parts: laser clustering, sub-image processing, dimensionality reduction, and classification. They utilize non-linear probabilistic regression models to match landmarks basing on position and appearance. They get good performance in cases with not too many features. Their method has been tested in the Victoria Park, which they define as an outdoor environment with dynamic objects, irregular terrain, and different illumination conditions. The available features are sparsely planted trees. They have reported that their method is able to correctly close loops of more than 400 m while associating 120 different features. An interesting characteristic of their method is that it has the capacity to cope with occasional failures. Their appearance models also correct erroneous position estimates.

Cummings and Newman [70, 71, 72] have developed an algorithm called FAB-SLAM that is a probabilistic approach that detects known places basing on their appearance. FAB-SLAM analyzes new observations to identify seen and unseen places. Their maps are represented by a set of visual words that are detected using a SURF descriptor. The previously visited places are detected when a new image contains a word that is already in the set. They have reported that their method avoids the *perceptual aliasing* problem, is robust in visually repetitive environments, and can be applied to online loop detection and closure. As can be observed in Figure 2.10, their results are suitable in large outdoor environments. In their recent work [73], they have obtained good results with trajectories of $1,000$ km.

Two important methods for loop detection considering 3D surfaces have been proposed by Johnson and Huber.

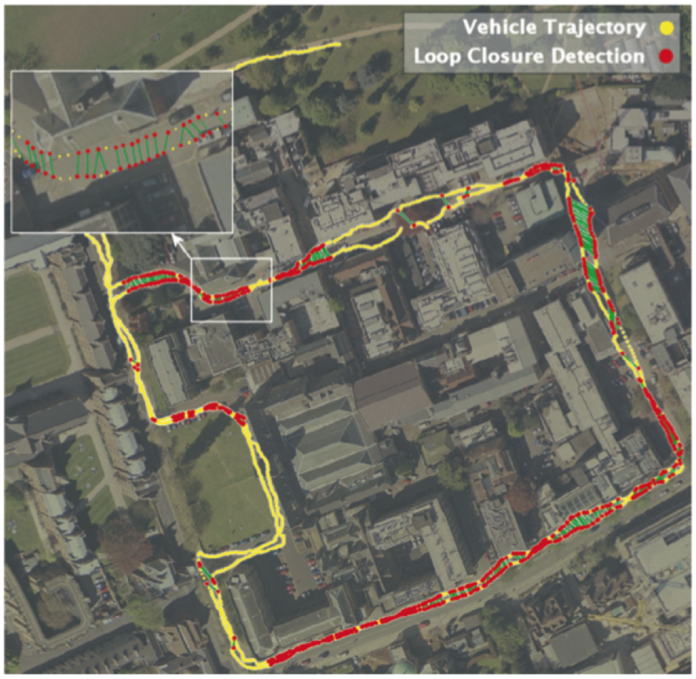

Figure 2.10: FAB-SLAM method by Cummings and Newman [70]. The circular path is traversed twice. It is composed of 2, 474 images that cover a 2 km trajectory. The images are collected from the yellow points. The seen places are marked in red and joined by green lines.

Johnson [74] has developed a method based on “spin images”, which he defined as local feature descriptors that represent the local surface shape around an oriented point. First, an oriented point of the surface is considered. After that, several parameters of the neighbor points are calculated. Finally, the spin image of this oriented point is composed of the accumulation of the neighbor points on the surface. This method requires to calculate correspondences between points. The loops are detected when two surfaces share many corresponding points. Johnson wrote that he chose this name because “the image generation process can be visualized as a sheet spinning about the normal of a point”. The results are accurate enough to obtain the whole models of objects. Besides, their method can also be applied to object recognition in environments with clutter and occlusions.

A similar method that is also based on spin images was proposed by Huber [75]. He has developed a global registration algorithm that is closely related to the loop detection problem. Firstly, a model graph is built by registering (scan matching)

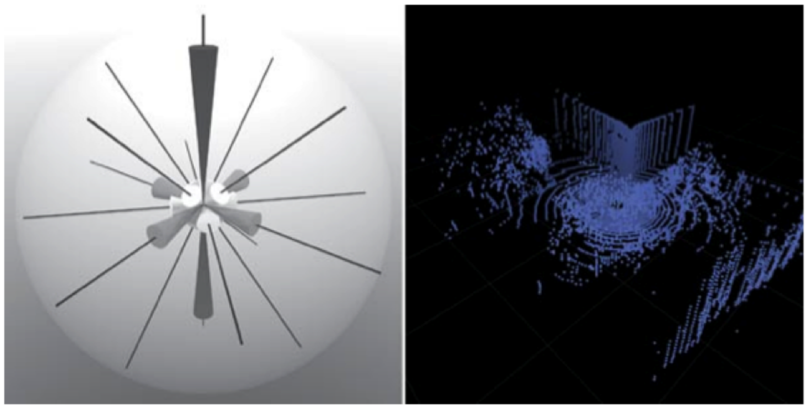

Figure 2.11: Feature-based loop detection using the NDT (Magnusson *et al.* [76]). Left: histogram. Right: laser scan. The thin black lines correspond to the directions. The cones are scaled according to the number of points.

all scan pairs using spin images. This model is used to detect correct and incorrect matches. The surface consistency is improved by computing sequences of matches. Huber has used this method to automatically build models of various types of scenes.

A detailed explanation of a geometric descriptor for 3D metric maps can be found in the work developed by Magnusson *et al.* [76]. Our loop detection method is based on their work. The environment is divided into cells and each one is represented by its NDT (eigenvectors are calculated for each cell and features are defined depending on the relationship between them). Each feature contains the number of cells that belong to an specific surface. The laser range data are described with feature histograms based on the surface orientation and smoothness, and the loops are detected by matching feature histograms. A visual example of the histogram is represented in Figure 2.11.

As said in the cited paper, it is necessary to define an efficient descriptor which has to be rotation-invariant. This descriptor compares two different scans using a specific formula that returns a difference value. This value is compared to a threshold to determine if the loop exists. Their method is able to detect loops without false positives (Figure 2.12).

Granström *et al.* [77] have used a machine learning algorithm called AdaBoost [78] to learn a classifier from previously extracted features. The feature descriptor is composed of 41 different features, all of them invariant to rotation. The first 33 are numeric values representing different properties such as: volume, difference, curvature, centroid, etc. This type of features has also been included in the loop

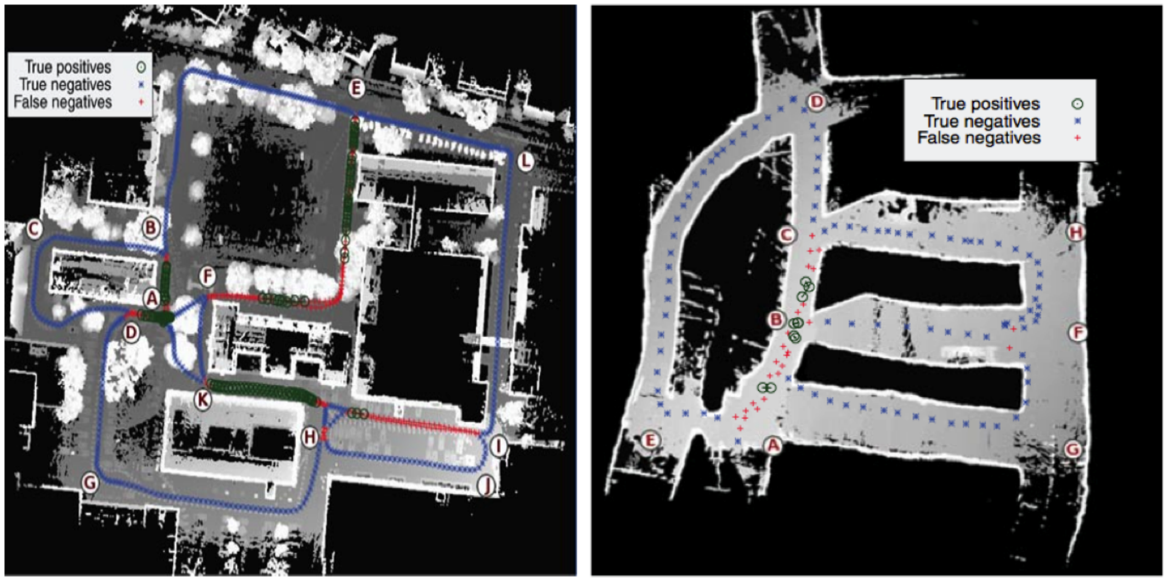

Figure 2.12: Loop detection by Magnusson *et al.* [76]. There are no false positives and all loops are detected.

detection algorithm defined in this work. The last nine are range histograms with different bin sizes. Their descriptor gets good performance when detecting loops from pairs of scans. However, a previous training is needed to build the classifier. A huge data set is necessary to build a robust descriptor. Therefore, it can be difficult to utilize this method in exploration or SLAM.

Bosse and Roberts [79] have developed a histogram-based technique for planar maps with two main components. The first one is an orientation histogram used to compute the rotation difference. The second one is a set of projection histograms that determine the distance between scans. The loops are detected by multiplying the peak value of the orientation histogram by the peak value of the projection histograms and comparing the result with a threshold. In Figure 2.13, their method results are shown in a large outdoor environment.

Finally, we have developed a loop detection algorithm that extracts the most important features from two different 3D laser scans in order to obtain an indicator that is used to detect when the robot is visiting a known place. Our approach allows the introduction of very different characteristics in the descriptor. First, the *surface* features include the geometric forms of the scan (lines, planes, and spheres). Each component is equal to the number of elements that belong to each feature. These components are similar to the features calculated by Magnusson's method. Second,

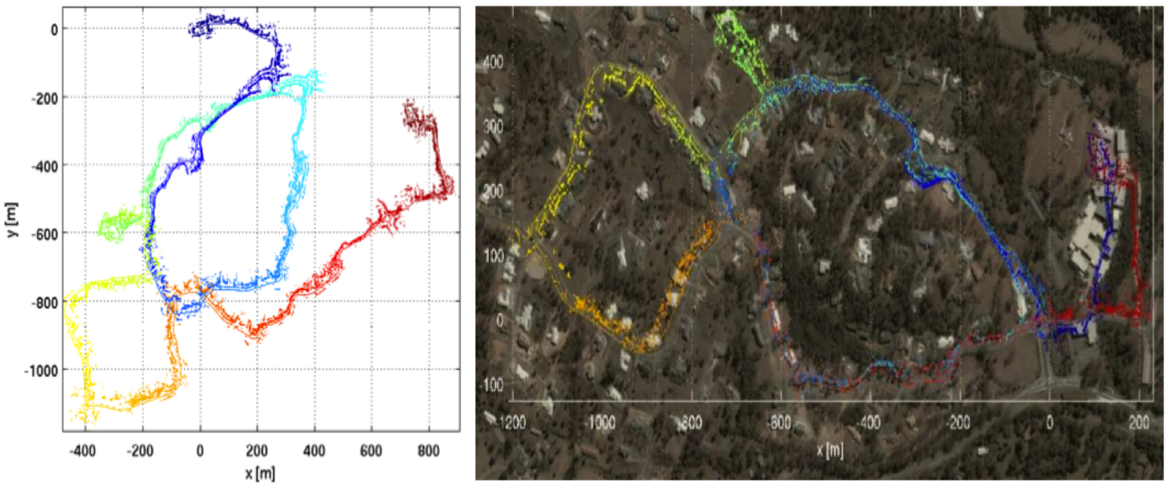

Figure 2.13: Loop detection by Bosse and Roberts [79]. Left: map before loop detection. Right: map corrected by loop detection.

the *numerical* features are values that describe other several numerical properties of the measurements: volume, average range, standard deviation of range, etc. This idea was taken from the work by Granström. The algorithm has been tested with real data to demonstrate that it is an efficient tool to be used in mapping problems.

Chapter 3

Global Localization in 3D Environments

There are many tasks in which an autonomous robot needs to know accurately where it is to carry them out successfully. The localization problem is crucial in robotics and can be defined as the search of the robot's coordinates relative to its environment, assuming that it is provided by a map. For instance, it is required to execute a geometrical path successfully during navigation. We can distinguish between two different situations: the re-localization problem (the robot knows its initial position, at least approximately) and the global localization problem (no knowledge about the robot's initial position). The second case will be studied in this work.

In our first approaches, we tried to solve the referred problem in a 2D map with an evolutionary filter [45]. The algorithm developed was called ELF (Evolutionary Localization Filter) and solves the global localization problem in a satisfactory way. Due to the available sensors and the accuracy requirements of certain tasks, the extension to 3D environments and an improvement in the method were developed after that [80]. In our ELF approach it was assumed that the robot was in a 2D map, and three parameters were estimated, corresponding to the position (x, y) and orientation (θ). We are now working on 3D maps, and four coordinates that represent the coordinate system of MANFRED-2 must be calculated, corresponding to the position (x, y, z), and horizontal orientation (θ). The DOF stated is 4 for simplicity (because MANFRED-2 works in 4 DOF) but the robot could work in 6 DOF $(x, y, z, \varphi, \theta, \psi)$ that represent position and orientation (roll, pitch, and yaw) when the map is three dimensional. The current method can be easily expanded to 6 DOF without increasing the computational cost.

The new algorithm has been called RELF-3D (Rejection Evolutionary Localization Filter in Three Dimensions). A detailed explanation of its final version and a complete study of the last developments and experimental results are presented in this document. This has been published in [81].

The RELF-3D method is based on the representation of the robot's location by a set of possible location estimates weighted by a fitness function. The state is recursively estimated using a set of results selected according to the weight associated to each possible solution included in the set. The solutions set evolves in time to integrate the sensor information and the robot motion information. The adaptation engine of the RELF-3D method is based on an evolutionary adaptation mechanism, which combines a stochastic gradient search together with a probabilistic search to find the most promising pose candidates.

Our research group is working on the development of the experimental platform MANFRED-2 (Figure 1.2). This robot has some built-in sensors which have been used in this work.

The first one is a laser range finder (SICK PLS), which provides us with 3D information about the environment. The original sensor measurements are 2D, but

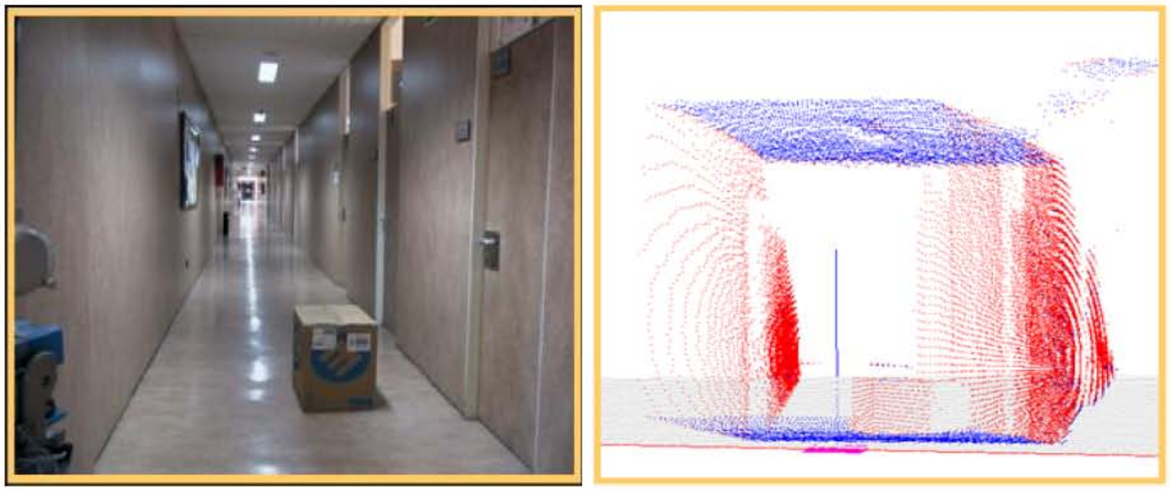
Figure 3.1: Corridor of the Department of Engineering Systems and Automation of the UC3M and corresponding laser reading.

we have added a motor that lets it rotate up and down, being able to obtain measurements in three dimensions. To understand this, imagine that the laser rotation movement is like a pan-tilt camera. The maximum horizontal (tilt) resolution, which is the minimum distance between two consecutive measurements within a scan in two dimensions, is 0.25°, and the amplitude of a 2D scan is 190°. The vertical (pan) resolution, which is the minimum distance between two consecutive 2D scans, is 1°.

The 3D laser reading is used by our localization module to try to locate the robot in a familiar environment. A typical environment and a real 3D laser reading can be seen in Figure 3.1. We have also implemented a simulated laser which works in 13 vertical scans separated by 5°, and each vertical scan contains 61 horizontal readings separated by 3°.

However, we need more information to address our problem completely. When the robot moves (the global localization problem has been solved, but the re-localization problem starts), we need information about this movement. In order to get it, the robot is also equipped with an optical encoder (HEDS-5540), a sensor necessary to obtain the odometry information. This information gives us an idea about the movement but has several drawbacks: the precision is very low and the error accumulates with the time.

3.1 Localization Problem Formulation and Solution

From a Bayesian point of view, the localization problem can be formulated as a probability density estimation problem where the robot seeks to estimate posterior distribution over the space of its poses conditioned on the available data. The robot pose at time t is defined as x_t.

The sensor data can be divided into two groups: environment measurements given by the perceptive sensors and pose information obtained by the motion sensors (encoders). The available information at time t is defined as:

$$Y_t \equiv \{z_{0:t}, u_{1:t}\} \equiv \{z_0, u_0, z_1, u_1, ..., z_{t-1}, u_{t-1}, z_t\},$$

where $z_{0:t}$ contains the perception sensor measurements and $u_{1:t}$ contains the odometry information. The components z_t and u_t contain the information at time t.

The posterior probability density function can be defined as $p(x_t|Y_t, m)$, where m is a known variable that represents the map of the environment. The term m will not be included from now on.

It is assumed that the process has the Markov property in order to estimate $p(x_t|Y_t, m)$. A stochastic process has the Markov property when the probability distribution of the futures states does not depend on the past states but on the present state.

The recursive determination of the posterior probability density can be computed in following two steps:

- *Measurement update.* Applying Bayes's rule to the last element of the measurement vector Y_t and assuming that the observation z_t is conditionally independent of the previous measurements given the state x_t, yields

$$\begin{aligned} p(x_t|Y_t) &= \frac{p(z_t|x_t, Y_{t-1})p(x_t|Y_{t-1})}{p(z_t|Y_{t-1})} \\ &= \frac{p(z_t|x_t)p(x_t|Y_{t-1})}{p(z_t|Y_{t-1})}, \qquad (3.1) \\ p(z_t|Y_t) &= \int_{\Re^n} p(z_t|x_t)p(x_t|Y_{t-1})dx_t, \qquad (3.2) \end{aligned}$$

  where the denominator of Equation (3.1) is obtained by marginalization.

- *Prediction.* The effect of a time-step on the state given the observations up to time t is obtained by observing that

$$p(x_{t+1}|Y_t) = \int_{\Re^n} p(x_{t+1}|x_t, u_t)p(x_t|Y_t)dx_t, \qquad (3.3)$$

where the assumption that the process x_t is Markovian, and then x_{t+1} is independent of Y_t, has been considered.

Equations 3.1, 3.2, and 3.3 provide the solutions to the bayesian recursive estimation problem. In general, the multidimensional integrals of these equations have no explicit analytical solutions when the models are not linear or Gaussian. It is necessary to describe $p(x_{t+1}|x_t, u_t)$ and $p(z_t|x_t)$.

The first conditional probability $p(x_{t+1}|x_t, u_t)$ is often named probabilistic motion model. This probability distribution will be obtained from the state space model of the mobile robot:

$$x_{t+1} = f(x_t, u_t) + \nu_t,$$

where ν_t is the motion noise. The motion model is a probabilistic generalization of the robot kinematics. It is described by a posterior density of the next possible states x_{t+1} given the state x_t and the control input u_t. The motion noise is typically modeled as gaussian and it is added to the movement information.

The second conditional probability $p(z_t|x_t)$ is often named probabilistic observation model. The uncertainty about the environment information perceived by sensors is expressed by this measure:

$$z_t = h(x_t) + \epsilon_t,$$

where ϵ_t quantifies the perceptive sensor noise. It is typically modeled as a gaussian which is added to the distance weighted by the sensors. Since it is assumed that the map is known and the robot's pose is x_t, it is possible to estimate z_t computing the measurements that should be observed when the robot is situated in x_t. The observation model $p(z_t|x_t)$ describes the posterior density over the possible sensor measurements z_t.

Different types of filters are obtained depending on the representation method of the probability density function $p(x_t|Y_t)$. Before the explanation of the proposed method, it is advisable to know a little bit about them. It is necessary to evaluate integrals to apply the Bayesian recursive filter. These integrals can be initially evaluated because they are composed of known functions which are defined in the problem. However, it is laborious to know the component that will be used in the next iteration *a priori* probabilities. It is because the estimate is used in the next iteration and it has an associated error. The next iteration inputs will be composed of *a priori* values that are not known analytically. Besides, one of the main disadvantages of the

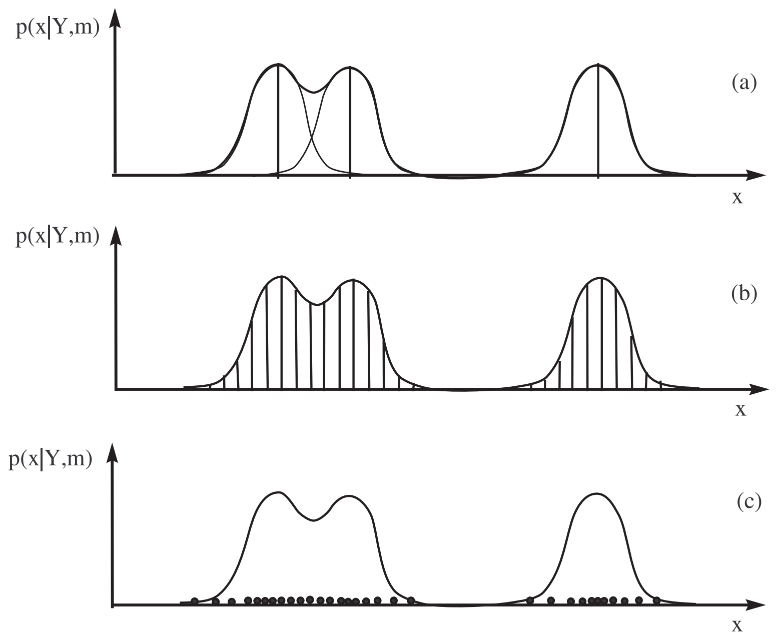

Figure 3.2: Approximations of probability density functions: (a) Mixture of Gaussians, (b) Piecewise, (c) Monte Carlo.

numerical integration in a n-dimensional space is the associated computational cost. It can be observed that the computational requirements grow exponentially with the state dimension.

There are different methods that approximate the Bayesian estimate function by an equivalent problem (Figure 3.2). The common idea is that the propagation of the continuous density function $p(x_t|Y_t)$ is replaced by the propagation in a finite set distributed over the region of interest. These methods are the following ones:

- Mixture of Gaussians: the whole distribution is approximated by a sum of weighted Gaussian distributions, thus the integrals are conceptually replaced by sums. The posterior density function is equal to the weighted sum of the probabilities induced by each Gaussian function:

$$p(x_t|Y_t) \approx \sum_{i=1}^{N} \gamma_i \mathcal{N}(x_t; \mu_i, \sigma_i^2), \tag{3.4}$$

  where γ_i is the weight of the Gaussian distribution i, μ_i is the mean, and σ_i^2 is the variance.

The advantage of this approach is that the multiplication and convolution of Gaussian distributions generate Gaussian distributions. It has been successfully applied by several authors[49, 39, 50]. The Multi-Hypothesis EKFs can be included in this type. It is not possible to apply traditional EKFs to multimodal distributions. One possibility is to utilize an independent EKF to track each possible pose.

- Piecewise approximation: the state space is divided into cells and the probabilities are referred to each region [40, 41, 82]. The density function is calculated for each node of the cell grid, thus the integral is approximated by a finite sum of the cell probabilities:

$$p(x_t|Y_t) \approx \sum_{i=1}^{N} p(x_t^i|Y_t)\delta_i(x_t - x_t^i). \tag{3.5}$$

  Each one of the N points of the grid has an associated probability $p(x_t^i|Y_t)$. This technique starts with an uniform probability distribution over the grid and then the integral is replaced by the sum of the cell probabilities.

- MC approximation: this method is based on particles. There are multiple groups that utilize this idea [43, 44, 83]. The probability distributions are replaced by a set of N sampled points that are called particles. These particles are sampled according to the probability distributions. It is possible to replace the integrals by sums of weighted samples. The particle set is distributed over the state space and the posterior density function can be defined as:

$$p(x_t|Y_t) \approx \sum_{i=1}^{N} w_t^i \delta(x_t - x_t^i), \tag{3.6}$$

  where w_t^i are the normalized weights.

  The most important shortcomings of this method are the computational cost and the relatively slow convergence of the algorithm.

The previously explained probabilities (Equations 3.2 and 3.3) are used to estimate the solution of the localization problem, but they are difficult to handle in general problems that are not linear or Gaussian. Each candidate parameter value in $\Re^n$ yields a value of $p(x_t^i|Y_t)$, reflecting the posterior probability of the robot pose given the data up to time t. The density function is calculated using all the available information about the system, but it needs to be weighted according to a given criterion to determine an estimate $\hat{x}_t$ of the true pose value. Different estimates can be obtained depending on the cost function that is chosen to discriminate between poses. Two common choices are the Least Squares Method (LSM) and the *Maximum A Posteriori* (MAP) estimator.

The solution of the LSM is defined as

$$\hat{x}_t^{LS} = \arg\min_{x_t^*} \int_{\Re^n} (x_t - x_t^*)^T p(x_t|Y_t)(x_t - x_t^*)dx = \int_{\Re^n} x_t p(x_t|Y_t)dx. \quad (3.7)$$

The optimal estimate calculated by the LSM is the conditional mean. Since the posterior probability distribution is multi-modal in the global localization problem, the estimate of the LSM is inconvenient.

The localization algorithm that we have developed, which is probabilistic but not Bayesian-based, concentrates on obtaining the better MAP estimator:

$$\hat{x}_t^{MAP} = \arg\max_x p(x_t|Y_t). \quad (3.8)$$

This approach is less dependent on statistical assumptions, has a simpler implementation, is robust from a statistical point of view, and has a lower computational cost than Bayesian methods.

3.1.1 Localization as a MAP optimization problem

The localization problem is basically an optimization problem, where the robot seeks to estimate the pose which maximizes the posterior probability density:

$$\begin{aligned}
\hat{x}_t^{MAP} &= \arg\max_x p(x_t|Y_t) \\
&= \arg\max_x p(z_t|x_t, u_{t-1}, Y_{t-1})p(x_t|x_{t-1}, u_{t-1}, Y_{t-1}) \\
&= \arg\max_x p(z_t|x_t)p(x_t|x_{t-1}, u_{t-1})p(x_{t-1}|Y_{t-1}) \\
&= \arg\max_x \prod_{i=1}^{t} p(z_i|x_i) \prod_{i=1}^{t} p(x_i|x_{i-1}, u_{i-1})p(x_0). \quad (3.9)
\end{aligned}$$

This expression requires the definition of $p(z_i|x_i)$ and $p(x_i|x_{i-1}, u_{i-1})$. These probability distributions will be obtained from the state space model of the mobile robot.

The MAP estimate expression can be easily stated as an optimization problem subject to constraints (the motion and observation models of the robot). The MAP estimate is the solution of the following problem under noise conditions:

$$\hat{x}_t^{MAP} = \arg\max_x \prod_{i=1}^{t} p(z_i|x_i) \prod_{i=1}^{t} p(x_i|x_{i-1}, u_{i-1}) p(x_0), \tag{3.10}$$

where $p(z_i|x_i)$ expresses the observation density function including the perceptive sensor noise, and $p(x_i|x_{i-1}, u_{i-1})$ defines the motion density function considering the motion sensor noise. Equation 3.10 can be modified by taking logarithms:

$$\hat{x}_t^{MAP} = \arg\max_x [\sum_{i=1}^{t} p(z_i|x_i) + \sum_{i=1}^{t} p(x_i|x_{i-1}, u_{i-1}) + \log p(x_0)]. \tag{3.11}$$

In general, this function has not an analytical solution in optimization problems that are neither Gaussian nor linear, and it has to be solved iteratively in order to avoid the difficulties associated with the optimization problem. These difficulties are:

1. It is highly non-linear. The observations and the motion information have non-linear components that are propagated to the density function.

2. The environment symmetries make the objective function multimodal. The objective function admits a high number of solutions at initial stages. It is specially important in highly symmetrical environments like offices and buildings. If the robot is stopped, the results depend only on $\sum_{i=1}^{t} p(z_i|x_i)$. This component presents a high number of potential solutions in this type of environments.

3. Another source of symmetries is originated by the sensor limitations. The range and angular resolution of the sensor add observation symmetries.

A set of possible solutions has to be initially generated to solve Equation 3.11. This set has to be maintained or modified according to the perceptive and motion information included in the cost function. The problem can be simplified if it is assumed that the initial distribution is a Gaussian because the optimization problem is now unimodal. It is even possible to obtain an analytical solution. This solution will be equivalent to the well known EKF in tracking problems.

The objective function to maximize will be noted as $f_0(x_t)$ from now on. The problem consists of estimating the pose $\hat{x}_t$ that maximizes $f_0(x_t)$ given all possible poses that satisfy the conditions $x_{t+1} = f(x_t, u_t) + \nu_t$ and $z_t = h(x_t) + \epsilon_t$. In other words, it is necessary to find the optimal value between the feasible solutions. A pose is feasible if the constraints f and h are satisfied. There are multiple optimal values at initial stages, thus the method has to be capable of handling multiple solutions. The Bayesian methods use the posterior density function. However, the method that is proposed here uses a different approximation. The main idea is to keep all feasible solutions. The solution set will evolve in time to the true solution according to the available information.

3.1.2 Recursive formulation of the optimization problem

The MAP problem formulation described by Equation 3.11 is not efficient from a computational point of view. In order to implement the global localization algorithm in a robot, a recursive formulation is required. The objective function can be reformulated in a more convenient form:

$$\begin{aligned} f_0(x_t) =& \sum_{i=1}^{t} \log p(z_i|x_i) + \sum_{i=1}^{t} p(x_i|x_{i-1}, u_{i-1}) + \log p(x_0) \\ =& \log p(z_t|x_i) + \sum_{i=1}^{t-1} \log p(z_i|x_i) \\ & + p(x_t|x_{i-1}, u_{i-1}) + \sum_{i=1}^{t-1} p(x_i|x_{i-1}, u_{i-1}) + \log p(x_0) \\ =& \log p(z_t|x_t) + \log p(x_t|x_{t-1}, u_{t-1}) + f_0(x_{t-1}) \end{aligned} \tag{3.12}$$

The MAP optimization problem can be written as

$$\hat{x}_t^{MAP} = \arg\max_x [\log p(z_t|x_t) + \log p(x_t|x_{t-1}, u_{t-1})]. \tag{3.13}$$

Then, by perturbing and searching new solutions to Equation 3.13, we obtain a recursive version of the MAP estimate.

An evolutionary algorithm that obtains the MAP estimate for the global localization problem is presented in the next section.

3.2 Evolutionary Localization Filter (RELF-3D)

The global localization algorithm is based on evolutionary optimization techniques. These techniques are probabilistic, but without derivatives or probability density functions to estimate the best solution to the localization problem. The evolutionary method that we have implemented is explained in this section. The original method has been improved with different ideas that are also explained here.

In our evolutionary algorithm, there are elements that correspond to possible solutions, and the fitness function value represents the error between real and estimated data. The method is population-based and each population member represents a possible solution (pose) to the global localization problem. The fitness function compares real data received by sensors (observation vector from real pose) with estimated data that must be obtained by sensors when the robot is located in the candidate pose. It is possible to estimate the measurements that must be observed from candidates because the map is known. The population set will evolve in time to the true solution by minimizing the cost function.

The stochastic search of the robot's coordinates is done using the DE method proposed by Storn and Price [1] for global optimization problems over continuous spaces, which is explained in this section. In order to do that, the reader can see Algorithm 1. Besides, a complete explanation of the DE algorithm can be found in our published work [45].

First of all, it is necessary to comment on some details about the environment and the robot's characteristics.

The environment has been modeled geometrically as an occupancy grid map in three dimensions and the robot's pose (the robot's pose is defined as the robot's position and orientation: x, y, z, and yaw) is represented with the cartesian coordinates and the horizontal orientation. In most of our experiments, each cell is a cube of 12.1 cm side and the whole map contains $500 \times 119 \times 25 = 1,487,500$ cells. It represents a 3D environment with a 871 m^2 area and a height of 3 m. If we want to determine the robot's localization, four coordinates must be estimated, defining a state space with four DOF in a 3D map. It is necessary to remark that it is also possible to work with six DOF (including roll and pitch) without increment of the computational cost. The environment considered from now on will have four DOF because the mobile manipulator MANFRED-2 works in a environment that can be modeled using this assumption. The simulated environment and the whole algorithm have been implemented using MATLAB, developed by MathWorks. More information can be found in Section A.4.1.

The search starts with a population of N_P candidates (the algorithm is population-based and N_P represents the number of elements), which are introduced in the localization module and evolve with time to the best solution. Each candidate is a

Algorithm 1 RELF-3D

```
1: function $RELF-3D(...)$
2:     for $i = 1 : N_P$ do
3:         $estimated\_dist\_3d(i) \leftarrow dist\_est\_3d(...)$
4:         $cost(i) \leftarrow fitness\_3d(estimated\_dist\_3d(i), real\_dist\_3d)$
5:     end for
6:     while (CONVERGENCE CONDITIONS) do
7:         for $i = 1 : N_P$ do
8:             $MUTATION$
9:             $CROSSOVER$
10:            $SELECTION \quad with \quad THRESHOLDING$
11:            $estimated\_dist\_3d(i) \leftarrow dist\_est\_3d(...)$
12:            $cost(i) \leftarrow fitness\_3d(estimated\_dist\_3d(i), real\_dist\_3d(i))$
13:                              ▷ cost function value calculation for next generation
14:        end for
15:        $DISCARDING$
16:        $[error, ind\_best] \leftarrow min(cost)$
17:        $bestmem \leftarrow pop(ind\_best)$
18:        $conv\_conditions\_checking(...)$
19:    end while
20: end function                        ▷ return bestmem, error and population
```

possible solution to the global localization problem (the robot's pose, with 4 DOF). The initial population size can be chosen randomly or it can be generated by an initialization method. Most of population-based methods select the initial population size in an empirical way basing on several experiments. However, an initialization method based on the information contained in the observation vector has also been implemented here. It will be explained in detail in Section 3.2.6.

For each candidate, its associated fitness function is calculated (line 2 to 5 of Algorithm 1). The fitness function is a key component of the method. It compares real data received in a laser reading with estimated data from a candidate solution. The estimated data have been sorted to configure the observation vector, with 13 vertical scans separated by 5° and 61 horizontal readings separated by 3°. The localization algorithm returns the population member that yields the lower value of the objective function. Since the cost function compares the observation vector of the candidate with the true observation vector, the best candidate will be that one with a lower fitness function value. It means that the observation vector obtained from the best candidate looks like the observation vector obtained from the true pose.

The main loop starts in line 6. If one of the convergence conditions is satisfied, the localization process ends successfully. In other words, the solution set evolves with time to the true solution. The convergence conditions must be chosen in order to decide the best moment to stop. The algorithm must stop when the true pose is reached.

Another loop, which contains the evolutionary search, starts in line 7. It consists on a generation of a new population for the next generation. In a single iteration the algorithm is executed to obtain the next candidates, evolving with time to the correct pose. The basic part of the evolutionary search is composed of three operators: mutation, crossover, and selection. Two additional mechanisms that improve the characteristics of the algorithm have also been developed: thresholding and discarding.

The most important parts of of the evolutionary method introduced in the previous paragraphs are explained in the following sections.

3.2.1 Mutation, crossover, and selection

There are three different operations that are applied to the whole population at each iteration in order to generate the population set for the next generation. First, the population candidates are perturbed to create the mutated population (mutation). After that, the crossover increases the population diversity. Finally, the selection mechanism is responsible of choosing the best candidates to be part of the next generation. This process is explained in this section.

The initial population is perturbed to generate a variation v_i according to the following expression:

$$v_i = x_a^k + F(x_b^k - x_c^k), \tag{3.14}$$

where x_a^k, x_b^k, and x_c^k are parameter vectors chosen randomly from the population at iteration k and are different from running index (this x is not the cartesian coordinate but the element of population, defined by four coordinates). The scale factor, $F \in (0, 1^+)$, is a real and constant factor that controls the amplification of differential variations $(x_b^k - x_c^k)$. It controls the population evolution rate. This factor has an empirical upper limit equal to 1. This value has been assumed because there are no cases where the optimization problem best solution is achieved with $F > 1$. It is possible to solve optimization problems with $F > 1$, but all cases that are successfully solved obtain better performance with $F < 1$. If $F = 1$, it is not possible to distinguish between the following combinations of vectors:

$$v_i^k = x_a^k + F(x_b^k - x_c^k) = x_b^k + F(x_a^k - x_c^k). \tag{3.15}$$

An example of the perturbation process in a 3D space can be seen in Figure 3.3. There is an initial parameter vector x_i^k and the perturbation is done with three

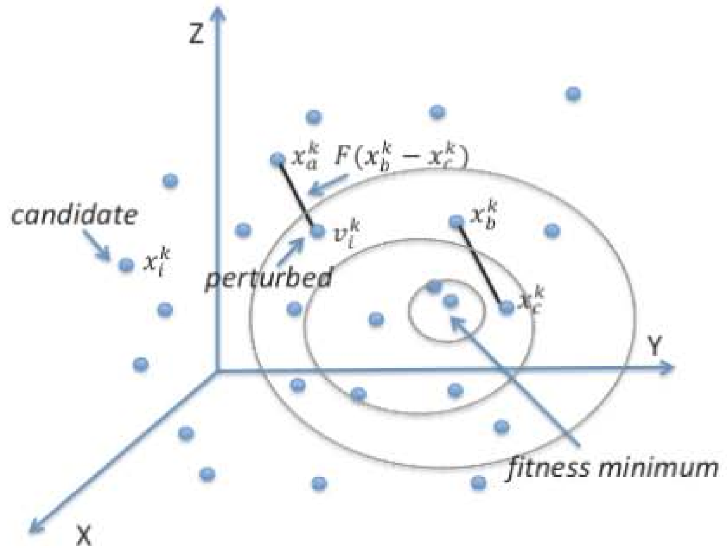

Figure 3.3: New population member generation for a single candidate.

random candidates and the constant factor F, generating the new parameter vector v_i^k. The population set is represented with points and there are circles with different sizes that represent the basin of attraction of the local minimum (depending on the cost function value). The objective is that the candidates evolve to the smaller circle, selecting the best candidates.

The perturbation process is repeated N_P times generating a perturbed population that is composed of the candidates to be the population members in the next generation. However, they are not the final candidates because the population will be modified again by the crossover mechanism.

The vector with index a represents the base vector that is moved by the differential variation. The most common versions of DE use a random vector within the population. It could be changed in order to obtain a different perturbed vector. It is possible to select the *best* population member instead of x_a^k. The current implementation allows both possibilities: random mutation and mutation from the best candidate. The first choice is more robust because it maintains the population diversity and is applied in general situations. However, the mutations from the best candidates are faster and they could be applied in some situations, such as the robot is situated in a place that is very easy to identify, or tracking tasks after a successful localization.

The random selection considers that all vectors have the same probability of being chosen. However, it is possible to pick vectors more than once per generation. It will cause that other vectors are never chosen. This method is known as Roulette Wheel Selection (RWS). The RWS method generates N_P random vectors without restrictions. It can be seen as a roulette wheel with equally sized slots that represent the population members. Each vector has the same chance of being chosen. In many Genetic Algorithms (GA), the slot size is proportional to the solution quality. It means that better solutions have wider slots. A classic roulette is assumed in basic DE.

There is a different approach called Stochastic Universal Sampling (SUS) that provides a more representative population sample because it guarantees that all vectors are selected once. The relation between SUS and RWS can be explained with a roulette game. Instead of playing N_P different times, the SUS method plays just once. The roulette slots are equally sized. The first vector is given by the roulette result. The second one is the slot that is next to the first one. The third one is next to the second one, and so on. In this case, all slots will be consecutively selected.

Different behaviors will be obtained depending on i, a, b, and c. If $b = c$, then the differential variations will be zero and the base vector will not be mutated:

$$v_i^k = x_a^k. \tag{3.16}$$

When b or c is equal to a, it is not possible to distinguish between some recombinations. This coincidence occurs on average once per generation in the RWS method. If the base index a is equal to the target index i, then the diversity is decreased and the crossover can be seen as a mutation of the target vector.

In order to increase the diversity of the new generation, the crossover is introduced. The term recombination is used when two or more vectors are exchanged or merged to create one or more trial vectors. Price *et al.* [84] define the discrete recombination, also known as crossover, as “an operation in which trial vector parameters are copied from randomly selected vectors”. The DE method applies the crossover mechanism to combine current population members (x_i^k) with perturbed vectors (v_i^k). This operation is applied to the whole population to form a new population set composed of trial vectors. These trial vectors are the candidates to be the population members for the next generation.

There are different ways to implement the crossover operation. An example can be observed in Table 3.1. The first row contains the parameter indexes. The crossover operation is applied to vectors 1 and 2. The first method is called one-point crossover. It is based on a randomly generated crossover point (it is equal to three in the example). The parameters are chosen from the first vector until the crossover point is reached. After that, the parameters are taken from the second vector. The exponential crossover has also an starting point, but it is based on different concepts. The parameters are chosen from the first vector until the starting point is reached. The

Table 3.1: Different crossover operations.

Vector Parameters	1	2	3	4	5	6
Vector 1	10	30	5	65	3	13
Vector 2	5	6	12	45	20	32
One-point (Cr point: 3)	10	30	5	45	20	32
Exponential (Cr point: 2)	10	6	12	45	3	13
			$(r_1 < Cr)$	$(r_2 < Cr)$	$(r_3 > Cr)$	
Binomial	10	6	5	65	20	32
	$(r_1 < Cr)$	$(r_2 > Cr)$	$(r_3 < Cr)$	$(r_4 < Cr)$	$(r_5 > Cr)$	$(r_6 > Cr)$

parameter of the second vector is taken for this parameter (it is the second parameter in the example). After that, the parameters are generated by comparing the crossover value (Cr) to a uniformly distributed random number between 0 and 1 that is generated anew for each parameter (r_j). If $r_j < Cr$, the parameter is taken from the second vector. If $r_j > Cr$, this parameter and all remaining parameters are taken from the first vector. Syswerda [85] defines the uniform or binomial crossover as "a process in which independent random trials determine the source for each trial parameter". The crossover is uniform because each parameter has the same probability of being taken from a given vector. This method compares r_j to Cr for each parameter. If $r_j < Cr$, the parameter is taken from the first vector; otherwise, the parameter is taken from the second one. The crossover probability Cr can present different values. The most common choice is $Cr = 0.5$.

The binomial crossover has been implemented in this method. The trial vector is denoted by $u_i^k = (u_{i,1}^k, u_{i,2}^k, \ldots, u_{i,D}^k)^T$ and its components are

$$u_{i,j}^k = \begin{cases} v_{i,j}^k; & \text{if } p_{i,j}^k < Cr, \\ x_{i,j}^k; & \text{otherwise,} \end{cases} \tag{3.17}$$

where $p_{i,j}^k$ is a randomly chosen value from the interval $[0, 1]$ for each parameter j of the population member i at iteration k, and Cr is the crossover probability and constitutes the crossover control variable. The random values $p_{i,j}^k$ are made anew for each trial vector i.

It is necessary to be careful about the selection of the crossover probability. It can also be considered as a mutation rate because the trial vector composition depends on this value. The crossover method also influences the population diversity. In general, a low Cr corresponds to a low population diversity. There are many GA that, on average, choose one mutated parameter per trial vector [86]. In this case, the crossover rate is equal to $1/D$. Zaharie [87] has also studied the crossover probability. She concluded that the best results are obtained with low Cr. Storn and Price [1] found that there are two different intervals with optimal performance ($0 \leq Cr \leq 0.2$ or $0.9 \leq Cr \leq 1$).

The application of a selection mechanism tends to reduce the diversity of a population, whereas mutation increases it. The amplification factor F has an important influence on the population diversity. Zaharie [87] has published and interesting paper that studies F depending on Cr and N_P. She has found a lower limit for F. If F is smaller than the limit, the population can converge even without a selection mechanism. In her experiments, Zaharie computes the population diversity depending on the DE parameters. The algorithm will converge prematurely when the trial population is less diverse than the current population.

She demonstrated that the population variance expected value after mutation and crossover is equal to

$$(2F^2Cr - 2Cr/N_P + Cr^2/N_P + 1)Var(pop), \tag{3.18}$$

where $Var(pop)$ is the population variance before mutation and crossover. The term $2F^2Cr - 2Cr/N_P + Cr^2/N_P + 1$ must be greater than one in order to increase the population diversity. Analyzing Equation 3.18, the critical values will satisfy the following equality:

$$2F^2 - 2/N_P + Cr/N_P = 0, \tag{3.19}$$

Solving this equation, she concluded that the theoretical critical value was $F_{crit} = 0.1341$ for $N_P = 50$ and $Cr = 0.2$. However, she also estimated an empirical critical value equal to 0.3.

The selection mechanism is responsible of choosing the best candidates for the next generation. It can be applied to two different stages in the evolutionary process.

First, it is possible to use it during mutation or crossover [88]. Some individuals have higher probabilities to be mutated because they present best fitness values. This strategy is often used by breeders and botanists that want to improve the offspring. The best candidates have more chances of surviving. However, this approach requires an initial knowledge about the problem because the fitness value limits must be known. This strategy has not been adopted in this work. Each vector has the same chance of being selected for mutation.

Second, it can be applied to choose the next generation members comparing the current generation to the trial vectors. This concept is called survivor selection or replacement.

There are many Evolutionary Algorithms (EA) that apply the selection mechanism to both stages. This option can cause premature convergence to a local optimum. GA typically uses the selection mechanism during the mutation stage, whereas DE and other EA utilize the survivor selection. The survivor selection has been implemented in this work. It is highly dependent on the cost function. It determines how the population set evolve with time to the true solution.

The new member of the population u_i^k is compared to x_i^k to decide whether or not vector u_i^k should become a member of generation $i+1$. If the vector u_i^k yields a better value for the objective fitness function than x_i^k, then it is replaced by u_i^{k+1}; otherwise, the old value x_i^k is retained for the new generation. The population set at iteration $i+1$ is determined by applying the survivor selection to the whole population.

The general ideas of the previous mechanism (mutation, crossover and selection) are well known and can be found in literature [88, 1]. The population set will evolve according to these ideas until the convergence conditions are satisfied. After that, the algorithm has converged to the true solution (robot's true location). Two additional mechanisms (thresholding and discarding) have been implemented in order to improve the characteristics of the evolutionary method. The convergence criteria and the fitness function are extremely important for the evolutionary method. These concepts will be explained in the following sections.

3.2.2 Thresholding mechanism

The selection operator has been modified in order to improve the method performance. A thresholding mechanism that avoids the premature convergence in noisy optimization problems has been implemented.

EA and, in general, population-based methods, have become very popular due to its applicability and implementation simplicity. One of the most important shortcomings of these methods is the premature convergence and the lack of robustness in noisy optimization problems. If DE is compared with other population search-based methods (for example, GA), it shows some weakness. This behavior has been studied by Krink *et al.* [89]. There are two different aspects with a negative influence: the DE method implements a greedy search strategy and the DE mechanisms for generating new potential solutions are less stochastic than other EA.

These disadvantages are significant when the difference between the candidate solution fitness value and the current population element fitness value is smaller than the fitness variance originated by the noise.

The idea of thresholding is to reduce the eagerness of the algorithm by rejecting those new solutions that do not improve the previous hypothesis in a pre-specified magnitude τ. This idea is not new and has already been applied to evolutionary computation problems [90]. The threshold can not be a fixed magnitude, because this unit depends on the noise variance and the fitness distance to the optimal fitness value. The noise variance might be estimated, but it is not easy to estimate the second factor.

First, the fitness value difference between the population member x_i^k and the candidate member u_i^k is calculated. Then, this difference is compared with a predefined threshold value τ in order to determine if the improvement shown by u_i^k is not caused by the noise. If this condition is met, the target vector x_i^k is replaced by u_i^k in the next generation; otherwise, x_i^k is kept into the population. The selection mechanism is modified and the following expression is obtained:

$$x_i^{k+1} = \begin{cases} u_i^k; \text{ if } f(x_i^k) - f(u_i^k) > \tau, \\ x_i^k; \text{ otherwise,} \end{cases} \tag{3.20}$$

where $f(x_i^k)$ is the fitness function value of the current population member, and $f(u_i^k)$ represents the fitness function value of the trial vector.

In conclusion, if the improvement in the fitness function is bigger than the variance (or the standard deviation, depending on the selected units), it can be considered that it is not caused by the noise, and the new member can be introduced in the population.

The algorithm convergence and robustness in noisy optimization problems and the comparison between the current method and the initial method without thresholding has been published in our recent work [91].

In our experiments, we have chosen a value that depends on the sensor noise because the thresholding mechanism tries to avoid the optimization in the noise band. The threshold τ is equal to $S_N f_i^k$, where S_N is the sensor noise (percentage over the distance weighted) and f_i^k is the fitness function value for the ith population member in the kth iteration. This value has been chosen empirically. For example, when the sensor noise is 3% over the distance weighted, a simple and adaptive threshold level of $\tau = 0.03 f_i^k$ has been adopted to reject the offspring solution generated by f_i^k. In spite of its simplicity, it works quite efficiently.

This selection mechanism decreases considerably the eagerness of the DE algorithm and also its speed of convergence. As a consequence of thresholding, the algorithm rejects a high quantity of new solutions and accepts only those solutions which present a clear improvement in the fitness function. Besides, the algorithm has no longer the ability of preserving good solutions if they show a slow evolution towards the optimal value. A compromise could be achieved by always keeping the best offspring solution into the population regardless its improvement size. Finally, the selection mechanism with thresholding is expressed as

$$x_i^{k+1} = \begin{cases} u_i^k; \text{ if } f(x_i^k) - f(u_i^k) > \tau, \\ u_i^k; \text{ if } u_i^k = u_{best}^k \wedge f(u_i^k) < f(x_i^k), \\ x_i^k; \text{ otherwise,} \end{cases} \tag{3.21}$$

where u_{best}^k is the best candidate solution in the offspring population.

The stochastic robustness of the algorithm, its computational cost, and the iterations to converge present worse values, so another mechanism (discarding) must be incorporated to reduce this negative influence.

3.2.3 Discarding mechanism

The use of a thresholding band tends to decrease the convergence speed of the algorithm, particularly at initial stages due to the rejection of the offspring that does not improve the previous hypothesis enough (below the threshold band). A discarding mechanism has been introduced to increase the speed of the algorithm while maintaining the stochastic advantages in terms of robustness of thresholding. This procedure is explained in our published work [91]. The idea is to determine the worst fitness individual of the new population and substitute it by a new solution close to a better one. In order to do that, a percentage of elements to be discarded is chosen, not only one (for example, 5%). In order to avoid concentrating the discarded solutions around the best existing individual, one of the members of the population with its fitness value located in the first half of the fitness ranking is selected randomly. This selected solution plus a relatively small random component is adopted as a new offspring.

The discarding mechanism is important at early stages of the optimization process, where errors are important, and it is not so interesting at final steps, where the population is close to the solution.

In the following, the discarding mechanism incorporated to the localization filter is formally addressed.

In spite of the robustness and high-accuracy of the DE-based localization algorithm, a large number of generations are still needed to converge to a solution due to the high rejections rate introduced by the thresholding operator. For this reason, an acceleration mechanism is proposed to enhance the convergence speed of the proposed method.

The discarding mechanism is based on the idea of replacing a small percentage of the worst elements of the trial population with better ones closer to better ranked solutions. In order to avoid premature convergence, each discarded individual is replaced by a candidate solution randomly selected among the fittest individuals of the current population plus a random perturbation value.

Consider a population P^k of N_P candidates sorted according to its cost value in increasing order:

$$\bar{P}^k = \{x_1^k, ..., x_i^k, ..., x_{N_p}^k\}, \tag{3.22}$$

where $f(x_i^k) \leq f(x_{i+1}^k)$. These are the candidates to form the population at iteration $k+1$ obtained after the selection plus thresholding process.

Let $W^k \subset P^k$ be a subpopulation of $\bar{P}^k$ containing the δ elements (δ represents the percentage of the population that will be discarded) of the population with worst

Algorithm 2 Schematic version of the RELF-3D

```
1: $P^{k=0} \leftarrow$ **initialization**
2: **evaluation of** $P^k$
3: **while** termination criteria $\neq$ true **do**
4:     $Pv^k \leftarrow$ **differential mutation of** $P^k$
5:     $Pu^k \leftarrow$ **binomial crossover of** $Pv^k$
6:     **evaluation of** $Pu^k$
7:     $\bar{P}^{k+1} \leftarrow$ **selection with thresholding of** $(Pu^k \cup P^k)$
8:     $P^{k+1} \leftarrow$ **discarding** $(\bar{P}^{k+1})$
9:     $k \leftarrow k+1$
10: **end while**
```

fitness function values:

$$W^k = \{x^k_{N_p-\delta+1}, ..., x^k_i, ..., x^k_{N_p}\}, \tag{3.23}$$

and let $S^k \subset P^k$ be a subpopulation with the β elements of the population with best fitness function values:

$$S^k = \{x^k_1, ..., x^k_i, ..., x^k_\beta\}, \tag{3.24}$$

with $\beta \gg \delta$.

A δ-size population $\bar{W}^k$ is built with candidate solutions randomly chosen among the elements of S^k. Since diversity must be preserved in order to efficiently explore the search space, a random noise is added to the new candidate solutions as follows:

$$\bar{W}^k = \{\bar{x}^k_w\} = \{x^k_s + \alpha\}, \tag{3.25}$$

where $w = \{1, ..., \delta\}$, α is a Gaussian distributed noise value, $x^k_s \in S^k$, and the index s is a random number chosen within $[1, \beta]$. Finally, the elements in W^k are discarded and replaced in the population by the components of $\bar{W}^k$ when the cost function value of the corresponding element in $\bar{W}^k$ is better.

Two conditions must be met for the discarding mechanism to work properly. First, the population size has to be large enough to assure diversity but not too large to increase the computational cost. Second, the δ value has to be set to a small percentage of the population; otherwise, there is the risk of being trapped in a false optimal value, especially in noisy environments.

To sum up, a schematic version of the evolutionary filter is listed in Algorithm 2, where P^k, Pv^k, and Pu^k are the current, mutated, and trial population at generation k, respectively.

3.2.4 Fitness function

The survivor selection mechanism is responsible of choosing the best candidates to form the population of the next generation. As a reminder, it compares the current population members (x_i^k) to the trial members (u_i^k) and the best candidates are selected. The tool that is used to decide if one solution is better than another one is the fitness or cost function. Each population member has an associated fitness value that represents how good the solution is.

Analyzing its value, it is possible to know if the solution of the global localization problem is admissible. That is the reason why the cost function value is employed by the stopping criterion to finish the evolutionary search.

The cost function compares laser data obtained from the robot true pose to laser data from the pose estimates. The localization algorithm minimizes the error and evolves to the true solution. As an example, the sensor considered in various experiments is a simulated laser that works in 13 vertical scans separated by 5°, and each vertical scan contains 61 horizontal readings separated by 3°. The localization method applies the fitness function aligning the current scan (estimated pose) with the reference scan (real pose). The population evolves to the correct location because the matching error is minimized.

The reference scan is obtained from the robot true pose. If the algorithm is working in a simulated environment, these measurements are obtained by emitting laser beams in the desired directions until the first object in the map is detected. The laser scan is composed of a distances set. It is possible to do it because it is assumed that the map is known. The second scan is obtained from the candidate solutions (estimated poses). The idea is the same used to compute the laser scan from the true pose in a simulated map. Assuming that the map is known, the robot is situated in the estimated pose (population member) and the laser beams are emitted obtaining the measurements. It has to be done from each population member every time the cost function is called.

An example can be observed in Figure 3.4. The simulated indoor environment is shown in the left part of the figure, whereas the laser reading from the true pose is represented in the right part of the figure. The measurements from the candidate poses are taken after situating the robot in different places of the simulated map.

The matching method applied in this work avoids searching for point associations by simply matching points with the same bearing, which is called PSM [66].

According to the MAP optimization problem formulation, the natural choice for the loss function is:

$$f_0(x_t)=\log p(z_t|x_t) + \log p(x_t|x_{t-1}, u_{t-1}) + f_0(x_{t-1}). \tag{3.26}$$

This expression contains the probability distribution of the observations error

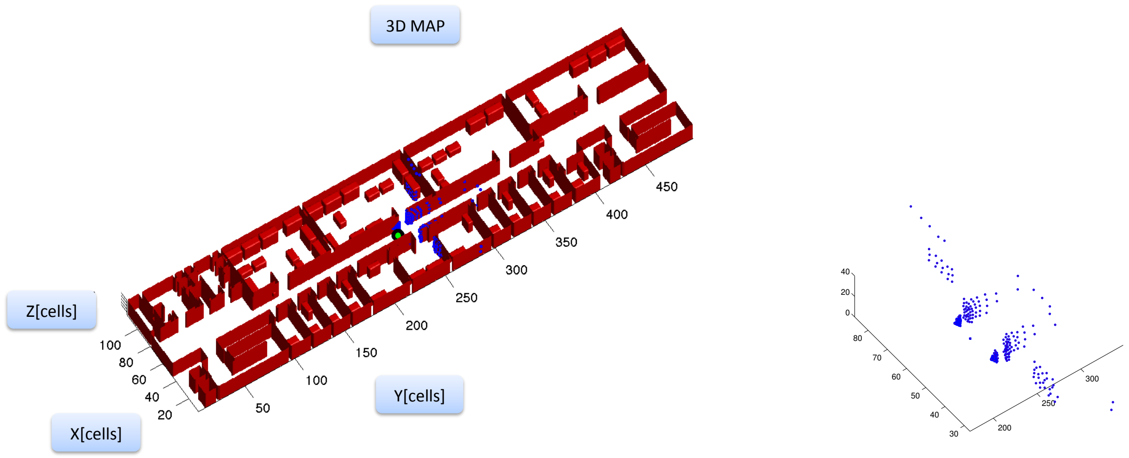

Figure 3.4: Simulated environment and laser scan from the true pose.

$p(z_t|x_t)$ and the probability distribution of the motion error $p(x_t|x_{t-1}, u_{t-1})$. An additional variable is used to incorporate the previous information about the robot's pose $f_0(x_{t-1})$.

The cost function can be rewritten to be consistent with the terms used to explain the evolutionary algorithm in Sections 3.2.1 and 3.2.2:

$$f_0(x_i^t){=}\log p(z^t|x_i^t) + \log p(x_i^t|x_i^{t-1}, o_i^{t-1}) + f_0(x_i^{t-1}), \tag{3.27}$$

where the odometry information is now expressed by o so as not to confuse it with the trial population members (the letter u is used with them). For simplicity reasons, the term t is a superindex. This function returns the cost value of an individual of the population.

If it is assumed that the observation error can be described by a Gaussian probability distribution with zero mean and known variance ($\mathcal{N}(0, \sigma^2)$), the integration of the individual probabilities of the laser beams into a probability density function (assuming conditional independence between the individual measurements) results in:

$$p(z^t|x_i^t){=}\prod_{j=0}^{N_s} p(z_{i,j}^t|x_i^t) = \prod_{j=0}^{N_s} \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{(z_j^t - \hat{z}_{i,j}^t)^2}{2\sigma^2}}, \tag{3.28}$$

where N_s is the number of sensor observations.

If it is also assumed that the motion error is a Gaussian distribution with zero mean and known covariance matrix ($\mathcal{N}(0,\Sigma_m)$), the motion error probability $p(x_i^t|x_i^{t-1},o_i^{t-1})$ can be expressed as:

$$p(x_i^t|x_i^{t-1},o_i^{t-1}) = \frac{1}{\sqrt{|\Sigma_m|(2\pi)^n}} e^{-1/2(x_i^t-\hat{x}^t)\Sigma_m^{-1}(x_i^t-\hat{x}^t)^T}, \tag{3.29}$$

where $\hat{x}^t$ is obtained from the estimated state $\hat{x}^{t-1}$ and the odometry measure o^{t-1}. This expressions has been included in the cost function to obtain a close formula.

There are many possibilities when choosing a suitable objective function to solve the recursive optimization problem. Assuming that the sensor measurements are Gaussian-distributed, one of the most common choices as a basis for the estimate is the L2-norm. In that case, the objective function to be minimized by the evolutionary algorithm for a point of the state space included in the population x_i^t could be given by the following expression:

$$f_{L2}(x_i^t) = \sum_{j=0}^{N_s} \frac{(z_j^t - \hat{z}_{i,j}^t)^2}{2\sigma^2} + \frac{1}{2}(x_i^t - \hat{x}^t)\Sigma_m^{-1}(x_i^t - \hat{x}^t)^T, \tag{3.30}$$

where $z^t = (z_1^t, \ldots, z_{N_s}^t)^T$ is the observation vector given by the 3D laser scanner at instant t, $\hat{z}_i^t = (\hat{z}_{i,1}^t, \ldots, \hat{z}_{i,N_s}^t)^T$ are the expected observations for that measurements if the robot was situated in x_i^t, $\hat{x}^t$ is the estimated true pose (if it exists, in time t), Σ_m is the covariance matrix of the state error, σ^2 is the observation error variance, and N_s is the sensor size or, in other words, the number of laser beams contained in a laser scan. We have considered that the laser error is Gaussian-distributed over the distance weighted. The second term of the expression depends on the estimated pose $\hat{x}^t$, which does not exist at the beginning, and could not be unique, because the robot could start from different poses. It is straightforward that the second term of the right part could be estimated as a function of the distance between the candidate pose and all the viable ones, but it will complicate too much the fitness function evaluation, adding also an statistical component.

In order to accelerate the calculation, we will not include in the fitness function the information given by the distance between the candidate pose and the estimated pose until the algorithm has converged to one single pose (it is considered that the algorithm has converged when all candidate poses are in a sphere of constant radio around the best one). Besides, the information is not stable when including distance information due to abrupt changes in the estimate. Excluding this term, the fitness function before convergence will be given by the following expression:

$$f_{L2}(x_i^t) = \sum_{j=0}^{N_s} \frac{(z_j^t - \hat{z}_{i,j}^t)^2}{2\sigma^2} = \frac{1}{2}\sum_{j=0}^{N_s} \frac{\nu_j^2}{\sigma^2}, \tag{3.31}$$

where $\nu_j = (z_j^k - \hat{z}_{i,j}^k)$ represents the discrepancy between the observed and predicted sensorial data.

Following are some factors in global localization that make this fitness function difficult to manage:

- The range and accuracy of the sensor and the number of sensors limit the possibility of distinguishing between different poses, leading the fitness function to a high number of global maxima.
- The geometrical similarities in the environment due to the repetition of the space distribution originates the presence of a high number of possible robot's pose solutions to the mean square loss function.

The loss function defined in this way provides us with an optimization mechanism which obtains an unstable parameter solution and, by extension, an unstable filter. The evolutionary filter can move from one local minimum to another, originating abrupt changes in the pose estimate. This problem comes from the fact that we are not using all the information we know about the system in the loss function. In fact, we only use the observation model to predict the sensor measurements at each position, and these predicted measurements together with the actual measurements are introduced in the loss function to evaluate the robot's pose estimate. The instability originated by the existence of multiple solutions to the loss function cannot be solved by considering only the available sensor observations.

In practice, the assumption of Gaussian observation noise is arguable. On the one hand, the presence of non-modeled obstacles, both static and mobile, lets us notice that the Gaussian probability distribution can be convenient, but in practice the distribution tail is too optimistic. Besides, the least squares method is not completely satisfactory when the noise model is not exactly known or the model is contaminated with other probability distributions.

The absolute error (L1-norm) may be an appropriate measure in some situations. For example, it presents a better performance with high errors originated in outliers or contaminated measures. If the L1-norm is not derivable, then it is necessary to apply linear programming methods to obtain a solution to the optimization problem (it does not happen in this method). Based on MC studies, the use of the L1-norm has been recommended when the errors follow one of these distributions: Laplace, Cauchy, mixture of normal and uniform, and contaminated normal.

If we assume that the observation error can be described by a Laplace error distribution, this distribution has the following form

$$f(x|\mu,\lambda) = \frac{1}{2\lambda}\exp(-\frac{|x-\mu|}{\lambda}) \tag{3.32}$$

with mean μ and variance $2\lambda^2$, usually referred as location and scale parameters, and typically denoted as $L(\mu, \lambda)$.

If we apply this probability distribution to model the observation noise, we obtain an L1-norm loss function to optimize:

$$f_{L1}(x_i^t)=\sum_{j=0}^{N_s} \frac{|z_j^t - \hat{z}_{i,j}^t|}{\lambda_z} = \sum_{j=0}^{N_s} \frac{|\nu_j|}{\lambda_z}, \tag{3.33}$$

where $\nu_j = (z_j^t - \hat{z}_{i,j}^t)$ represents the discrepancy between the observed and predicted sensorial data.

Obviously, the use of the L1-norm is independent of the model if we consider the loss function simply as a loss function to optimize, independently of the noise model. Due to the optimization method we use is not derivative-based, the L1-norm optimization problem can be solved easily and the computational cost is not substantially different from the cost of solving other loss functions.

A comparison between the L1-norm and the L2-norm has been published in [92].

3.2.5 Convergence conditions

There are some situations where it is easy to know if the optimization process has converged to the optimal solution. For example, if the optimization problem consists of satisfying several constraints, the algorithm converges when all constraints are satisfied. Nevertheless, it is not easy to define optimal convergence criteria in multi-modal or multi-objective optimization problems. An optimization problem is multi-objective when there are several objective functions. A multi-modal problem can be defined as a problem with multiple good solutions, i.e., optimization problems with multiple local minima. Besides, it is not easy to quantify the optimal value for the fitness functions. The termination criteria of the RELF-3D are described in this section.

Many different parameters can be included in the convergence conditions of the EA depending on the optimization problem characteristics. A brief review of the most important ones is given below:

- Fitness function value: the fitness function optimal value can be know or can be estimated in some cases. In these cases, this value represents how good the solution is. However, there are many other cases where this value cannot be used because it is not possible to estimate an optimal value.

  If the optimal value is known, the termination criterion will depend on the distance between the best member fitness value and the optimal value (d_o). Different distances can be fixed resulting in different behaviors.

- Number of iterations: a different variable that can be used as a termination criterion is the number of iterations or generations. This factor can be applied when the fitness function optimal value is unknown. It is possible to choose an upper-limit of iterations or a number of iterations without changes in the cost value of the best population member (N). It is not an advisable criterion when the optimal value is known because the algorithm may be stopped before the minimum is reached.

  The variable N has to be fixed empirically. This factor depends on the convergence speed of the optimization method. For example, DE has lower convergence speed than other EA, so N has to be larger.

- Population characteristics: it is also possible to choose characteristics associated to population members as convergence conditions. A typical example is the difference between the cost function value of the best element and the cost function value of the worst candidate (d_{bw}). This variable can cause premature convergence in some situations. For example, if the limit is set to 100 and the distance to the optimal value is also around 100, the convergence condition can be satisfied with high values in the fitness function of the best candidate. The convergence is premature because the algorithm can converge to better solutions. If this factor is used together with the distance to the optimal value, it is advisable to choose d_{bw} much lower than d_o.

- Time limit: the amount of time can be chosen as a convergence condition. In this case, the termination criteria do not depend on the fitness function values or the number of iterations. This factor is closely related to the computational cost. If the optimization process is heavy from a computational point of view, it can be a good idea to limit the time.

- Human monitoring: the optimization process can be stopped manually depending on several factors: best fitness value, number of iterations, time limit, and so on. However, it is not a criterion that can be utilized in our field.

It would be interesting for us to find an expected value for the cost function. In order to do this, we have noticed that there are two effects that influence the cost function: the measurement noise and the estimation error. If we obtain a perfect estimation, the second effect can be eliminated, but an error introduced by the noise will always exist. Hence, it is possible to estimate the expected value of the objective function when it is close to the true value $E(f_0)$. We have to remark that it is the fitness function value in the real position.

Two different fitness functions were described in Section 3.2.4. The first one is given by the L2-norm:

$$f_{L2}(x_i^t) = \sum_{j=0}^{N_s} \frac{(z_j^t - \hat{z}_{i,j}^t)^2}{2\sigma^2} = \frac{1}{2}\sum_{j=0}^{N_s} \frac{\nu_j^2}{\sigma^2} \tag{3.34}$$

If we observe the term $\sum_{j=0}^{N_s} \nu_j^2/\sigma^2$ of the last expression, the components ν_j^2/σ^2 are random variables with a standard Normal distribution $N(0,1)$, and the sum follows a Chi-Square probability distribution with N_s DOF. This probability distribution, which is well known and tabulated, has an average of N_s and a variance of $N_s/2$. Therefore, the expected value of the objective function we are trying to minimize is

$$E[f_{L2}] = \int_{-\infty}^{+\infty} f_{L2}(\nu)p(\nu)d\nu = N_s/2 \tag{3.35}$$

This expression tells us that even if the pose that we are evaluating was the correct pose of the robot, and due to measurement errors that occur in the sensor during the environment perception, the expected value of the objective function is $N_s/2$.

The determination of a stopping condition for the algorithm brings us back to what was discussed above, because once we know the best expected value of the objective function, it is possible to establish a stopping condition based on some of the statistical parameters associated with this objective function.

As we have concluded that our cost function can be approximated by a Chi-Square with N_s DOF, it is straightforward to associate the objective function value with a given probability. In other words, an objective function value f_{1-p} with probability $1-p$ means that the optimum value has a probability $1-p$ of being below f_{1-p}. These values are often found tabulated in statistical literature for some typical quantiles and a given number of DOF.

If the fitness function is the L1-norm, the function to minimize takes the following form:

$$f_{L1}(x_i^t) = \sum_{j=0}^{N_s} \frac{|z_j^t - \hat{z}_{i,j}^t|}{\lambda_z} = \sum_{j=0}^{N_s} \frac{|\nu_j|}{\lambda_z}. \tag{3.36}$$

As in the L2-norm case, it is required to calculate the expected value $E[f_{L1}]$ when the pose under evaluation is the true one and the noise is a Normal distribution with zero mean and standard error deviation λ_z. The expected minimum fitness value can be easily calculated and will be

$$E[f_{L1}] = \int_{-\infty}^{+\infty} f_{L1}(\nu)p(\nu)d\nu = \int_{-\infty}^{+\infty} \sum_{j=0}^{N_s} \frac{|\nu_j|}{\lambda_z} p(\nu_j)d\nu_j$$

$$
\begin{aligned}
=&\sum_{j=0}^{N_s}\int_{-\infty}^{+\infty}\frac{|\nu_j|}{\lambda_z}\frac{1}{\sqrt{2\pi\lambda_z^2}}e^{-\frac{\nu_j^2}{2\lambda_z^2}}=\frac{2}{\lambda_z\sqrt{2\pi\lambda_z^2}}\sum_{j=0}^{N_s}\int_0^{\infty}|\nu_j|e^{-\frac{\nu_j^2}{2\lambda_z^2}}\\
=&\frac{2}{\lambda_z\sqrt{2\pi\lambda_z^2}}\sum_{j=0}^{N_s}\left[-\lambda_z^2e^{-\frac{\nu_j^2}{2\lambda_z^2}}\right]_0^{\infty}=\frac{2}{\lambda_z^2\sqrt{2\pi}}\sum_{j=0}^{N_s}\left[0-(-\lambda_z^2)\right]\\
=&\frac{2N_s}{\sqrt{2\pi}}.
\end{aligned}
\tag{3.37}
$$

Besides, we have chosen other simple criteria that have been used experimentally in certain situations. This criteria do not ensure convergence but may lead to good results in less time. If one of the following convergence conditions is satisfied, the localization process finishes:

- Number of iterations without changes in the fitness function value of the best estimation is bigger than a constant.
- Number of iterations without changes in the fitness function value of the worst estimation is bigger than a constant.
- Number of iterations without changes in the difference between the fitness function value of the best estimation and the fitness function value of the worst estimation is bigger than a constant.

3.2.6 Initial population size determination

If we analyze the population-based global localization algorithms from a practical point of view, one of the most important limitations consists of the estimation of an optimum size of the initial population [93]. MC methods require a precise adjustment of the number of particles, optimization-based methods also need an estimation of the initial population (if they are population-based), and a certain number of hypotheses must be generated in multi-hypotheses Kalman filters. The initial population size is very often adjusted empirically for a given environment and it requires a large number of trials and experiments.

A solution to the initialization problem in a population-based evolutionary global localization filter is developed in this thesis. It automatically generates a number of elements for the initial population based on the environment size and the amount of information contained in the first observation of the mobile robot. The efficiency of the proposed method and its capabilities have been tested in a simulated indoor environment.

Related Work

Different families of algorithms focus on the global localization problem: Bayesian-based methods (grid-based probabilistic filters and MC localization methods can be included here), optimization-based methods (DE and PSO filters), and hybrid methods (multi-hypotheses Kalman filters). A large amount of examples can be found in the literature [40, 94, 41, 88, 39, 58]. It is very common to use particle-based methods to solve this problem because they are robust and their applicability has widely been demonstrated. These approaches use a number of elements that are sampled over the space as candidates to be the true solution. Most of them use a fixed number of elements and the initial number is estimated empirically. These two characteristics, which have rarely been studied by groups working on this field, make these algorithms inefficient from a computational point of view. Marchetti *et al.* [95] provide a systematic analysis of particle-based localization methods, and Kümmerle *et al.* [51] apply a particle filter to estimate the full 6D state of the robot.

If we talk about an adaptive number of elements, Fox [96] changes the size of the particle filter on-the-fly depending on the error and Koller and Fratkina [97] adjust the number of samples according to the likelihood of observations. Grisetti *et al.* [98] improve the performance of their method by re-sampling and reducing the number of particles.

Finally, there are not many groups working on information theory and its application to mobile robotics. Bourgault *et al.* [99] maximize the map information by simultaneously maximizing the expected Shannon information gain (Mutual Information) on the occupancy grid map and minimizing the uncertainties of the vehicle's pose and map features in the SLAM process. Grocholsky [100] has studied information-theoretic models for coordination and cooperation. Rocha *et al.* [101] address the mapping problem through a probabilistic approach based on information theory.

Initialization problem

The initialization problem consists of the development of a method capable of obtaining an appropriate number of elements for each situation. This number must satisfy two conditions: it must be large enough to ensure that the global localization algorithm converges to the true pose, and it must be close to the minimum value that ensures a robust convergence (because the computational cost increases with the size). It is easy to check the first condition empirically, but it is more difficult to deal with the second one.

However, it is too early to explain it completely, and we will progressively introduce the reader to this subject.

This problem has been solved empirically in our previous experiments. The algorithm has been tested in a given environment in order to obtain the appropriate population size to be used in the initialization phase. The robot is situated in different positions and the population size is changed to obtain a value that ensures convergence to the true solution.

There are some factors that affect the initial population size required to localize the robot successfully:

- **The environment size:** it is intuitive that the environment size is a crucial factor because the number of elements required to localize the robot successfully will be larger for larger environments. After the experiments, it can be ensured that this is the factor that most influences the calculation of a suitable population size.

- **The environment symmetries:** symmetries exist when there are different places within the environment where the robot receives similar observation vectors. For instance, office buildings are highly symmetrical because they contain many offices of the same size and appearance. It is straightforward that the population size required to localize the robot increases with the number of symmetrical places, because these similar places require a minimum effective set of poses to manage them properly. According to the experiments carried out, this number is in the range of $20 - 25$ candidates for each possible pose. It is important to remark that this number can vary from one place to another in a single map. For example, if there are six similar offices, the number required to localize the robot in one of these rooms will be six times higher than the number required when the robot is situated in the corridor (assuming that there are not symmetrical places in the corridor). Therefore, it can be concluded that the initial population requirements increase with the number of symmetries.

- **The sensors information:** the number of symmetries of the environment depends on the information given by the sensors. If the robot does not receive enough information, the number of symmetries could increase. It is strongly related to the sensors resolution. For example, imagine an empty office with four different corners and a door. If the robot is looking at one corner, there are four possible locations, while if it is facing the door and it is open there is only one option. The robot's position and orientation within the environment, the sensors range, and the type of sensor have an important influence on the population size requirements.

- **The size of the basin of attraction of the global minimum:** if the basin of attraction of the global minimum is small, it will be more difficult to achieve the goal, and the population size must be larger. There are two different characteristics with an important influence on this aspect. The first one is the number

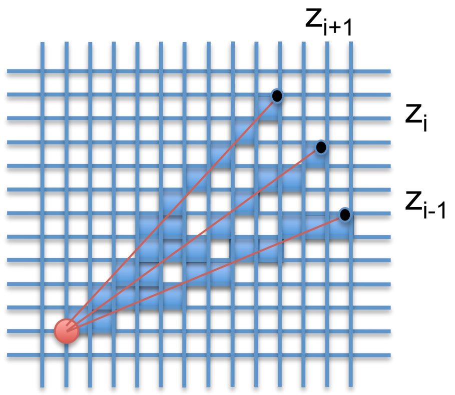

Figure 3.5: Three consecutive laser beams over a grid map.

of discontinuities (for example, the discontinuities originated by columns), because they originate abrupt changes in the cost function and also in the local minima basin. The second one is the size of the observed area. If the robot is situated in a small room, it will be more difficult to localize it because the basin of attraction of the global minimum will be generally smaller. It will be easier to find a big basin of attraction in a larger area.

Sensors information

The first question to be solved is how to compute the information that is received by the mobile robot. This information is contained in the observation vector. This question is one of the keys of the method, and it should be taken into account that the global localization algorithm is grid-based and the information is obtained by a laser range finder.

The mobile robot is initially situated in an unknown place of the environment and it receives the observation vector, which is composed of laser readings. Three consecutive laser beams (z_{i-1}, z_i, z_{i+1}) are shown in Figure 3.5.

The laser beam projection over the grid map contains information about the occupation probability of the cells that are crossed by it. This information is traditionally used in mapping methods to estimate the occupancy probability of each cell of the map.

The designed algorithm focuses on the information contained in each measurement. In this sense, it is considered that each crossed cell has a piece of information and the most important ones are the measurements with a low entropy. A low entropy means that there are significant evidences about the occupation of the cell, because the probability of being occupied or not is high. Figure 3.5 shows that the cells whose center is close to the laser beam (crossed by the laser beam) are filled. These cells have a very high or a very low occupancy probability and the information contained in the laser reading is referred to these cells. The number of cells crossed by a laser beam is used to develop a first idea about the quantity of information contained in this measurement. The information contained in the observation vector $z_{i,t}$ can be expressed as

$$I(z_{i,t}) = \sum_{j=1}^{N_z} I(m_j) \simeq [\frac{z_{i,t}}{d}]\frac{1}{N_x N_y}, \tag{3.38}$$

where I is the information, m_j represents the jth cell, N_z is the number of cells crossed by the laser beam, d is the cell size used in the map, and N_x and N_y are the map dimensions in cells. It is straightforward that the information contained in a given measurement is linked to the range. The larger the measurement, the larger the information about the environment contained in it. The total amount of information contained in a whole scan can be expressed as

$$\begin{aligned} I(z_t) &= \sum_{i=1}^{N_s} I(z_{i,t}) \\ &\simeq \sum_{i=1}^{N_s} [\frac{z_{i,t}}{d}]\frac{1}{N_x N_y}. \end{aligned} \tag{3.39}$$

Expression 3.39 shows that the information contained in a scan, which depends on the sensor nature, is equal to the area covered by the sensor scan divided by the total area of the map (the information has been normalized). N_s represents the number of laser beams of the observation vector. For example, the area covered by an ultrasonic sensor is larger than the area covered by a laser beam (but the accuracy is higher in the second case). It is necessary to introduce more factors in Expression 3.39 to obtain a better estimate of the laser scan information. The first one is the overlapping between sensors (two consecutive laser beams). Comparing two consecutive laser beams, there are some cells read by both scans. These overlapped cells do not give more information, and they should be introduced only once in the estimation formula. Such objective can be reached through the estimation of the overlapping between scans.

Sensor overlapping

The overlapping between two consecutive laser beams (i and $i+1$, separated by $\alpha°$) can be seen in Figure 3.6. The laser readings contain information about those cells whose center distance to the laser beam is smaller than $d/2$. The overlapped area is defined by the polyhedron $O - p_2\prime - p_1 - p_2$. If α is small enough, the area will

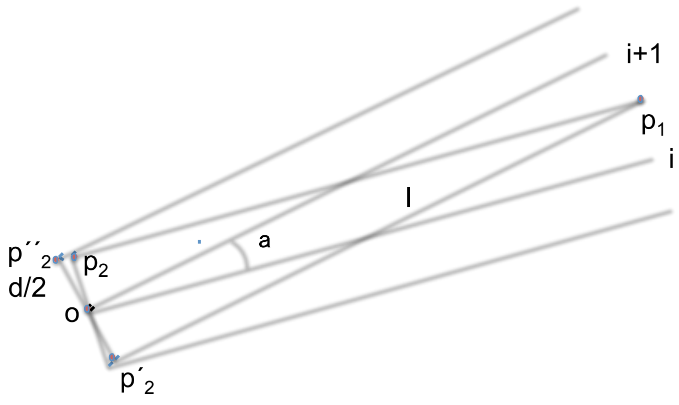

Figure 3.6: Overlapping between two consecutive laser beams.

be defined by the triangle $p_2\prime\prime - p_1 - p_2\prime$. The distance between $p_2\prime$ and $p_2\prime\prime$ is d and the distance from $p_2\prime$ to p_1 is l. The overlapped area is basically defined by d and l.

The first factor (d) depends on the beam width and it will be assumed that it is equal to the cell size of the map.

The second factor (l) can be obtained as $l = d/\tan\alpha$ and some values can be calculated for different angular resolutions and a fixed cell size equal to $d = 12$ cm:

- $\alpha = 0.5° \Rightarrow l = 13.75$ m.
- $\alpha = 1° \Rightarrow l = 7.06$ m.
- $\alpha = 3° \Rightarrow l = 2.03$ m.

It is clear that the overlapping will be more significant with a higher angular resolution. It is because there are more cells that are crossed by both laser beams and the angular distance between them is smaller.

The maximum overlapped area between two consecutive laser readings is equal to $A_m = d \cdot l/2$. But different levels of overlapping are found depending on the measured distances $z_{i,t}$ and $z_{i+1,t}$. In the first case, when both values are over l, the area is the maximum one (A_m). In the second case, when at least one of the measurements is smaller than l, this maximum value is not reached.

The effective area ($A_{eff,i+1}$) covered by the laser beam depends on the overlapped area and is equal to

$$A_{eff,i+1} = \begin{cases} z_{i+1}d - [\frac{d \cdot l}{2}]; & \text{if } z_i > l \text{ and } z_{i+1} > l, \\ z_{i+1}d - [\frac{d \cdot l}{2} - \frac{d(l-z_m)^2}{2l}]; & \text{otherwise,} \end{cases} \tag{3.40}$$

where z_m is the minimum of z_i and z_{i+1} and the terms in brackets are the overlapped areas. When one measurement is smaller than l, the overlapped area is not maximum and a portion of the previously defined triangle must be extracted. This area is defined by the triangle of sides $l' = l - z_m$ and d'. The quotient between d' and l' is constant: $d'/l' = d/l$, thus the area is equal to $d' \cdot l'/2 = d \cdot l'^2/2l$.

Initial population formula

An estimate of the information contained in the sensor measurements at a given time is obtained in the previous paragraphs. This value is a measure of how complete the information about the environment is. It is necessary to obtain an expression that calculate the population size that is introduced in the population-based global localization algorithm. An empirical method based on the previous concepts has been developed in this work. It depends on the effective area covered by the laser scan and the total area of the environment. Therefore, this number will be proportional to the next expression:

$$N_p = \frac{A_{map}}{A_{eff}} = \frac{d_x d_y}{\sum_{i=1}^{N_s} A_{eff,i}}. \tag{3.41}$$

The effectiveness and robustness of this method will be demonstrated with experimental results. A 2D laser scan of 61 laser measurements separated by 3° is used as sensor and the cell size is equal to 12.1 cm/side. It originates an overlapping of 2.28 m affecting 19 cells. When the observed area is larger, the estimated number will be smaller, and viceversa. For instance, when the robot is situated in a corner, most of the measured cells are overlapped and the effective area will be small. The number of potential perceptive equivalences is high (all corners are similar) and the total number of equivalent poses rises very fast.

There are other factors influencing the initial estimation of N_p. Some of them, such as symmetries, cannot be easily computed, but the effect of the discontinuities and the sensor noise can be introduced in the formula.

Discontinuities effect

The previously explained method works efficiently in environments without strong discontinuities, but it will not be robust enough with a large number of strong discontinuities. This will be explained using Figure 3.7. The robot is located at point 3 facing the wall and there is a column which does not let it have a complete view of the wall. An unidimensional localization is considered for simplicity. There are two strong discontinuities when the robot receives the information about the column and the wall. The cost function values observed in the lower part of the figure are obtained checking different hypotheses from different positions. There is a local minimum in the correct position, but it is situated between two local maxima. This kind

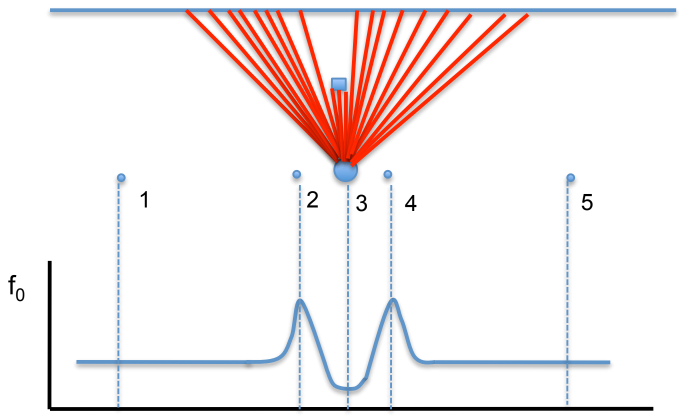

Figure 3.7: Discontinuity effect on the fitness function minimum.

of abrupt local minimum requires a high-density sampling to reach the basin.

Different and strong discontinuities can be found in higher dimensions (x, y, θ), thus it is necessary to detect them to increase the population size.

A way of dealing with this problem is to check the number of strong discontinuities in the initial scan and use this number as a factor to increase N_p. The factor introduced is equal to $k_d = (n_d - 1)$, where n_d is the number of strong discontinuities. It has been assumed that a strong discontinuity appears when the difference between two consecutive measurements is over the 40% of the measurement and more than 30 cells. The number of strong discontinuities of the laser scan is calculated and the correction value is introduced in the initial population estimation expression:

$$N_p = k_d \frac{A_{map}}{A_{eff}}. \tag{3.42}$$

Noise effect

Another effect to be considered is the observation noise level, because it increases the differences between measurements and the basin of attraction is also affected. This effect is similar to the discontinuities one, because the population requirements increase with the noise. A multiplicative factor k_n is also introduced in the estimation formula. This factor is equal to $k_n = 1 + \sqrt{\sigma}$, where σ is the average standard deviation of the measurement noise, which has zero mean and a standard deviation equal to the σ % over the laser distance. The updated formula will be

$$N_p = k_n k_d \frac{A_{map}}{A_{eff}}. \tag{3.43}$$

The effectiveness of the empirical formula used to determine the initial population will be evaluated in different simulated environments under noise levels which are considerably worse than the noise levels of commercial laser devices.

3.3 The Localization Method

A global localization method based on evolutionary concepts (RELF-3D) that solves the problem in a robust and efficient way using 3D sensor data has been developed. The evolutionary filter, which is the engine of this method, has been explained in the previous section. The whole localization module is described in this section. A simulation of the localization method implemented in the robot can be seen in Algorithm 3.

Algorithm 3 GLOBAL LOCALIZATION METHOD

```
 1: PARAMETERS INTRODUCTION
 2: $real\_dist\_3d \leftarrow dist\_est\_3d(...)$                              ▷ Laser simulation
 3: $pop \leftarrow start\_pop(...)$
 4: $dir\_desp \leftarrow NULL$
 5: while $dir\_desp \neq end$ do
 6:     $[bestmem, error, pop] \leftarrow$ RELF-3D(...)
 7:     $move\_robot \leftarrow VALUE$                                      ▷ robot's displacement
 8:     $bestmem \leftarrow bestmem + move\_robot$
 9:     $real\_position \leftarrow real\_position + move\_robot(1 + error\_pos)$
10:     $real\_dist\_3d \leftarrow dist\_est\_3d(...)$                           ▷ Laser simul.
11:     for $i = 1 : N_P$ do
12:         $pop(i) \leftarrow pop(i) + move\_robot(1 + error\_pos \times random(1))$
13:     end for
14: end while
```

The robot is located at a place in the environment. This location is represented by three spatial coordinates and the orientation in the horizontal plane. It is important to remark that our method works in 3D environments with a maximum of six DOF (x, y, z, $roll$, $pitch$, and yaw), but there are cases where we can assume the hypothesis that the localization could be solved in less dimensions. For instance, when the environment is plane and the robot has a fixed height, we can consider the z coordinate as a constant value. Our robot works in a 3D map with four DOF.

We do not have any information about the location of the robot, thus the place can be considered randomly. The mobile robot receives the information in a 3D laser reading. Our method is now working with simulated data, and the reading is done in line 2 of the algorithm from the true location of the robot. The 3D laser scan is generated by obtaining the laser beam projections from the true location within the simulated map.

After that, an initial population is generated, covering the map randomly (line 3). That population contains N_P possible candidates. The population size can be an empirically adjusted fixed number or it can be generated by the initialization tool described in the previous section.

When the laser reading has been done, the main loop starts (line 5), working in the following way: the robot tries to locate itself on the map (we have called it *step*) and, in order to do that, the evolutionary algorithm (line 6) is executed, returning the estimated solution, the error, and the final population. The reader can observe that the evolutionary algorithm described in the previous section is called in line 6 by the global localization module.

The concept *step* will be used several times, and we will explain what it means in our context. A *step* can be regarded as the moment in time at which the robot uses a single 3D laser scan to locate. When it receives another scan and motion information, a new *step* starts. In other words, we say the robot is located in a single *step* when the robot uses information from a single 3D laser scan, without considering movement information. If the robot moves to another site and takes another scan, it uses more information to locate (the second laser scan and the movement information). In that case we say it is in the second *step*.

Once the localization process in a *step* has finished, the robot starts moving. The robot location and the best estimate are moved according to that displacement (with an added error, lines 8 and 9) to integrate the movement information. The sensor makes a new laser reading (line 10) and receives odometry data. Then the possible locations are displaced according to the odometry (with an error, too, lines 11-13) and the localization process in the second *step* starts, but now with the results of the first one in the initial population. It is obvious that the localization process is easier in this case, because there is more information. When the second *step* finishes, the third *step* starts, and so on.

The algorithm is running continuously while the robot moves within the environment, thus the robot's pose is autonomously updated. We have simulated that the robot has completed its task introducing an *end* in line 5. According to Algorithm 3, the loop finishes when an *end* is introduced in the movement variable, which means the robot is turned off or the movement is over and the robot is successfully localized.

It is necessary to define several parameters before the execution. These parameters can be fixed offline or they can be manually introduced before execution (line 1). They can be a constant number or adaptive values that depend on different factors. The most important ones, with a critical influence on the behavior of the algorithm, are listed below:

- Population Size.

  It is also necessary to introduce the initial number of elements. It is critical and depends on the environment size. When the robot is localized, the needed number is smaller, and we can reduce it with a positive influence in the computational cost.

Another problem is how to determine the population number for a given environment. On one hand, the number depends on the map size. On the other hand, there is another important element: the symmetries. The symmetries originated by the robot's position and its perception capabilities can multiply the potential number of equivalent maxima. This number can vary from one in the most favorable case to more than 50, as we have observed in our experiments. The required population number can change dramatically because a certain number of population elements are required to maintain an hypothesis until new data are perceived.

It is commonly known that non-linear population-based global optimization algorithms can easily and prematurely converge to point estimates that are not globally optimal. It can be originated by a lack of population diversity, the slowness of the search algorithm, or the existence of a local minima in a multimodal objective function. The first two problems can be addressed by increasing the population size, but the third one is more difficult. First, the function can be multi-modal, so the algorithm will not converge until the multi-modality disappears. Second, if the noise level rises, a fictitious multi-modality induced by the noise level can appear.

As a reminder, this parameter can be generated by an initialization method (Section 3.2.6).

- Mutation Amplification Factor (F) and Crossover (Cr).

  F is a real and constant factor that controls the amplification of differential variations in the perturbation of the parameter vector (mutation). In this document, the experiments are done with $F = 0.85$. The value of Cr controls the crossover and is fixed to 0.75 experimentally. Both variables were explained in Section 3.2.1.

  We have also adopted an adaptive adjustment of the perturbation amplification factor F. This mechanism tries to maintain a high amplification factor while the population has not converged to the promising areas (a wide scope search is required) and to limit the algorithm search scope when the population set is distributed in the most feasible areas.

- Upper Limit of Iterations.

  There is also an upper limit for the number of iterations per *step*, which is an added feature designed to track the robot faster when it is not necessary to localize it in a single one. It is faster because we use more information (several laser readings and odometry information), and it can be changed to obtain different results. For example, 15 iterations per *step* makes the algorithm go faster, but we need more *steps* to obtain a good localization. Nevertheless, 500 iterations make the algorithm go slowly, but we can localize the robot in a single *step*.

- Tracking.

  Once the localization process has finished in a satisfactory way, the robot knows exactly its position. Our problem is now converted into the tracking problem. In other words, the robot will need to be continuously localized, updating its pose after small changes of position.

  In order to do that online, the population number is reduced drastically when the fitness function of the best estimate is under a threshold, which means that the robot is localized. We have chosen 10 as the population size after that. This reduction implies an impressive advance in the computational cost. In each motion cycle the robot can now update its pose accurately and quickly.

Chapter 4

3D Mapping: DE-based Scan Matching and Feature-based Loop Detection

An autonomous robot must obtain information about its surroundings to accomplish multiple tasks that are greatly improved when this information is efficiently incorporated into a map. Some examples are navigation, manipulation, localization, and so on. This mapping problem has been an important research area in mobile robotics during the last decades. It does not have a unique solution and can be divided into multiple sub-problems.

The goal is that an autonomous robot can build a map and localize itself in it. The environment is composed of physical objects and the robot needs to obtain a truly representation of the world perceived by its sensors.

Different aspects of the mapping problem have been described in Chapter 2. This work is focused on various problems that are connected to this subject. Summarizing, the main problem consists of the generation of consistent and robust maps considering the available information. This mapping process has been divided here into several stages that represent the tasks that should be solved in order to generate consistent maps. These tasks are the following ones:

1. Robot's pose estimation: the robot's pose (position and orientation) with respect to the last one must be computed. The movement information has to be updated.

   The purpose of our autonomous mobile robot is to work in indoor environments where it is not possible to use external signals or sources such as the GPS. There are two options to estimate the robot's pose. The first one is to detect known features using the sensor information. It implies *a priori* knowledge about the world. The second idea, which has been adopted here, is the incremental integration of the motion information (wheel encoders). The encoders information is transformed into wheel displacements. After that, an odometry model is used to compute the robot's pose from the wheel displacements. This model will be explained in detail later in this document. It is well known that the pose estimation using only odometry information is not accurate enough because of the accumulated error. Hence, these errors should be corrected by other methods.

2. Pose correction via registration: the accumulated error when considering only proprioceptive sensors (robot's pose estimation defined in the first point) has to be corrected using the exteroceptive sensor information. It is crucial to obtain a good estimate of the metric relation between different scans (our robot works using a 3D laser range finder), which is often called registration or scan matching. The pose is corrected by matching the last acquired scan with the previous one.

3. Loop detection and loop closure: the model obtained is not consistent after registration because the accumulated error due to local small errors can be still

very important. It is basic to detect when the robot is navigating through a previously visited place, to check the global error and to minimize it in order to give consistency to the global map. The detection task is usually referred to as loop detection, and the global error minimization is called loop closure.

This chapter is focused on two aspects of the mapping process: registration and loop detection.

We have implemented a scan matching algorithm for 3D environments. It is based on the DE algorithm (particle-based evolutionary algorithm that evolves in time to the solution that yields the cost function lower value). If the cost function is properly chosen, it is possible to solve the scan matching problem using this method. The high accuracy and computational efficiency of the proposed method have been demonstrated with experimental results.

The loop detection problem is also addressed in this thesis. We have developed a loop detection method that compares features extracted from two different scans to obtain a loop indicator. This approach allows the introduction of very different characteristics in the descriptor. First, the *surface* features include the geometric forms of the scan (lines, planes, and spheres). Second, the *numerical* features describe several numerical properties: volume, average range, curvature, and so on. The algorithm has been tested with real data to demonstrate that it is an efficient tool. All true loops are correctly detected without false detections. Besides, it is a versatile method that admits different parameters and settings.

This chapter is organized as follows. A review of the mapping process is given in Section 4.1. In Section 4.2, the scan matching method is explained. Finally, the loop detection algorithm is presented in Section 4.3.

4.1 Mapping Method

The mapping task for mobile robots is an open problem with different solutions and descriptions depending on many factors: available sensors, techniques applied, type of environment, etc. There is neither a standard way to accomplish it nor a unique description. In this section, this problem is described according to the techniques used here. A flow chart of the mapping process presented in the following paragraphs can be observed in Figure 4.1. It will be explained below, but some aspects related to this task are described before.

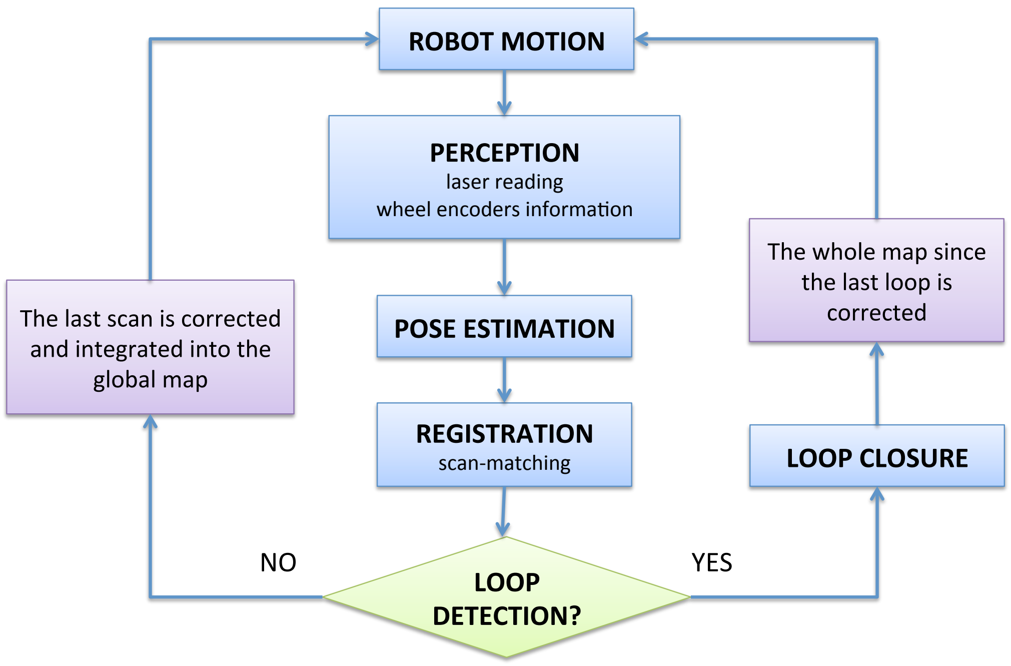
ROBOT MOTION
PERCEPTION
laser reading
wheel encoders information
POSE ESTIMATION
REGISTRATION
scan-matching
LOOP DETECTION?
NO
YES
The last scan is corrected and integrated into the global map
LOOP CLOSURE
The whole map since the last loop is corrected

Figure 4.1: General mapping process. Laser measurements and odometry information are computed after the robot motion. The robot's pose is estimated considering motion information. After that, it is corrected via registration. Finally, the loop detection algorithm gives consistency to the map by correcting the accumulated error when a pre-visited place is detected.

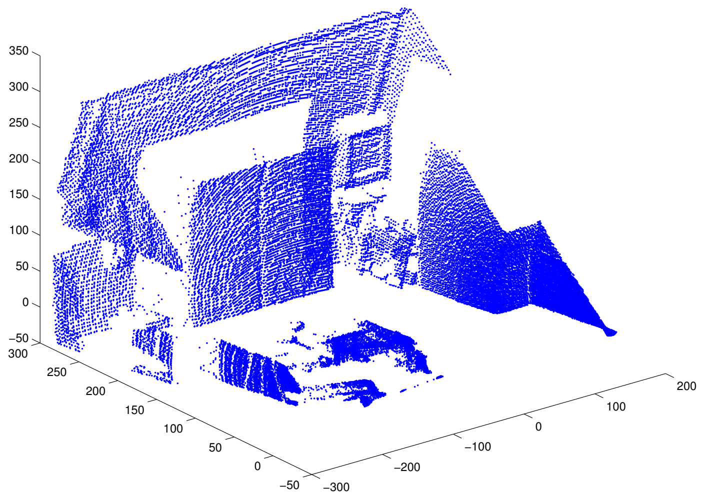

Figure 4.2: 3D laser reading obtained by MANFRED-2, which is an experimental platform fully developed by the Robotics Lab of the Carlos III University of Madrid, Spain. It has been recorded inside a lab of the university campus. It is composed of approximately 45000 points. All units in cm.

The first aspect to consider is the map dimensionality, which depends on the available sensors. We have assumed that the mobile robot works in a 3D world. The exteroceptive information is measured by a 3D laser scan (an example can be observed in Figure 4.2) and the motion information is computed by wheel encoders. The DOF in this type of environment is six in the most general case. Three DOF represent the position, which will be given in cartesian coordinates, and the other three compute the orientation.

Therefore, the robot's pose is defined by the following coordinates: position ($Pt = (x, y, z)$) and orientation ($O = (\varphi, \theta, \psi)$) (roll, pitch, and yaw angles), resulting in the following variable:

$$P = (x, y, z, \varphi, \theta, \psi), \tag{4.1}$$

where P is the robot's pose.

Each movement in the described space is composed of a translation plus a rotation. The translation is given by the cartesian coordinates $Tra = (\delta x, \delta y, \delta z)$. The orthogonal matrix that describes a clockwise/left-handed rotation with Euler angles

φ, θ, ψ, and x-y-z convention, is given by

$$Rot = \begin{pmatrix} c\theta c\psi & -c\varphi s\psi + s\varphi s\theta c\psi & s\varphi s\psi + c\varphi s\theta c\psi \\ c\theta s\psi & c\varphi c\psi + s\varphi s\theta s\psi & -s\varphi c\psi + c\varphi s\theta s\psi \\ -s\theta & c\theta s\varphi & c\varphi c\theta \end{pmatrix}. \qquad (4.2)$$

The notation has been simplified ($c\varphi$ means $cos(\varphi)$).

Each movement from an initial position to a final one can be calculated computing a translation plus a rotation:

$$Pt_{final} = Rot \times Pt_{initial} + Tra. \qquad (4.3)$$

Working directly with 3D data provided by the laser scan is not efficient from a computational point of view. A 3D laser reading can be composed of more than 45000 points when using a maximum resolution. The scan size is reduced in a preprocessing phase in order to increase the computational efficiency, but without losing the capability of obtaining good matching. The following steps have been implemented: resolution decrease, median filtering, range correction (elimination of outliers), distance filtering (points located close together are joined into one point), and segmentation (elimination of those elements that are far away from the rest). The computational cost is decreased, approximately, by 50 times by introducing data reduction.

The reader is now prepared to understand the mapping method described in Figure 4.1. Several stages must be followed to generate a consistent model of the robot's surroundings.

It has been considered that the *Robot Motion* stage is the starting point of the diagram, which means that the mapping process starts at this point. The reason of this choice is that the sensor reading is done after robot motion [5] . In other words, the robot is moving around the environment and it stops. After that, its sensors receive environment information from this location. This information is incorporated into the model using the mapping method, thus the global map is updated with newly obtained measurements. The robot can continue exploring the environment when this information has been efficiently incorporated into the model and a consistent map has been generated and/or updated.

First, the robot's pose changes with respect to the last one must be computed and the laser reading is received in the *Perception* stage. The mobile robot is equipped with encoders that measure wheel displacements. The environment information is contained in a 3D laser reading in the same way described before.

The *Pose Estimation* stage consists of converting the measured wheel speed into a vehicle displacement using the kinematic model of the robot. The robot's pose is

[5]There is no *a priori* information about the environment, so there is no motion to compute during the first iteration or execution. The robot directly obtains the sensor measurements.

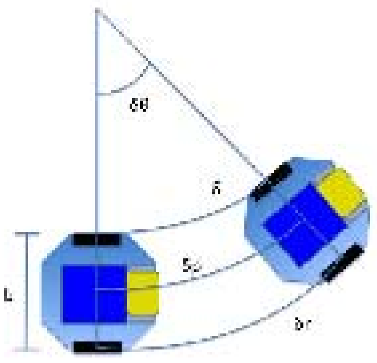

Figure 4.3: Odometry model of a mobile robot when moving between two different locations. This model is used to generate the robot trajectory using information given by wheel encoders.

iteratively updated given the odometry information. It can be continuously estimated, but it is not accurate enough so as to obtain a robust model of the environment, and that is why other tools are necessary for mapping.

Since the encoders measure wheel displacements, only three DOF representing plane position and horizontal orientation can be estimated considering the odometry information. The following explanation of the pose estimation method is based on our experimental platform and assumes that there are three DOF for simplicity. However, our mapping tools have been designed to work in six DOF as maximum and different examples with sensors that receive 6D information are shown in the experimental results.

Figure 4.3 shows the robot displacement between two different locations given by their cartesian coordinates and orientation: $p_0(x_0, y_0, \theta_0) \rightarrow p_1(x_1, y_1, \theta_1)$.

The wheel displacement information is given by sensors, and it is equal to δl for the left wheel and δr for the right one. The pose p_1 is computed using the following equations:

$$\delta p = \frac{\delta l + \delta r}{b}, \qquad \delta\theta = \frac{\delta l - \delta r}{b}, \tag{4.4}$$

$$x_1 = x_0 + \delta p \cos(\theta_0 + \delta\theta/2), \tag{4.5}$$

$$y_1 = y_0 + \delta p \sin(\theta_0 + \delta\theta/2), \tag{4.6}$$

$$\theta_1 = \theta_0 + \delta\theta/2, \tag{4.7}$$

where δp and $\delta\theta$ are the position and angular variations and b is the robot's base width.

This information is then fed up to the mapping module for pose correction according to the exteroceptive information. This is done by registering the newly acquired laser reading. The current scan pose is corrected until the best overlap with the reference scan is achieved. The most successful scan matching methods have been described in Section 2.3. A DE-based scan matching method has been implemented in this work. A complete explanation of our method is given in Section 4.2. Besides, the ICP-based scan matching method developed by Triebel *et al.* [27] has also been implemented for comparison.

The models generated after registration are not consistent enough to conclude that the mapping task has been successfully achieved. This consistency will be given by detecting the known places and eliminating the accumulated error when it exists (loop closure). If the pose estimate has an small orientation error at the beginning of the robot's path, the accumulated error after some time can be very important. In order to avoid that, several methods have been proposed. For example, two different possibilities have been proposed by Jensen *et al.* [102]. The first one is based on scan alignment and global registration, and the second one is a feature-based approach using an EKF and the Symmetries and Perturbation Model [103].

Our work focuses on loop detection. Different features that are extracted from a 3D laser scan are the variables used to calculate the similarities between two places. We have developed a feature-based loop detector based on the works by Magnusson *et al.* [104] and Granström *et al.* [105]. Our method is completely detailed in Section 3.3.

The whole map since the last loop is corrected when the loop is detected. After that, the robot continues exploring the environment with the corrected and updated map.

If there is no loop, the current scan coordinates obtained after registration and their associated scan are introduced in the global map, thus the model is also updated but no loop is closed. The robot continues its exploration and new measurements will be obtained.

4.2 Registration: Scan Matching

Since the pose estimation considering only motion information is not accurate enough, it is necessary to develop additional mechanisms to obtain better maps. The first idea, that includes information obtained by perceptive sensors, is called registration or scan matching. One of the crucial parts of any SLAM algorithm is the registration method. The laser scan matching consists of the current scan pose correction until the best overlap with the reference scan or model is achieved.

There are different classifications of scan matching methods. This was detailed in Chapter 2, but is also summarized here. The scan matching methods found in the literature can be developed to work with 2D or 3D data. The last case is addressed here. The main difference between both of them is the computational cost. There are also local [18] and global [62] methods. The local methods match single scans. Their disadvantage is that the final map is inconsistent because the accumulated error can be important. However, their advantage is that this inconsistency can be minimized by relaxation mechanisms. The global methods consider the current scan and the global model. Their associated shortcoming is that one single mistake is fatal because a wrong measurement is included in the model, not being possible to correct this error by additional mechanisms. It is also possible to distinguish between feature-based (feature extraction before the scan matching), point-based, or mixed (correspondence between points and features) approaches. Some examples of point-based scan matching are ICP [24], IMRP [18], IDC [18], PSM [66], and so on.

A DE-based scan matching algorithm has been implemented in order to solve the referred problem. It consists of a particle-based evolutionary algorithm that evolves in time to the pose that yields the cost function lower value. The fitness function calculates the correspondences between model and data and the evolutionary filter minimizes the distances between corresponding points.

The DE algorithm engine has also been used in our recent work to solve the global localization problem, which can be defined as the search of the robot's coordinates in a known environment with no *a priori* information about its location. A complete explanation can be found in Chapter 3. This scan matching technique is similar to the localization filter.

Two different ideas have been applied to improve the algorithm performance:

- Correspondences modification: the scan matching algorithms are based on the minimization of cost functions that calculate correspondences between points. A modified scan matching algorithm has been implemented, which finds correspondences between points of the same type. This means that the cost function considers corresponding points that share any specific characteristic.

- Acceleration with k-d trees: a k-d tree [106] (k-dimensional tree) is "a space-partitioning data structure for organizing points in a k-dimensional space". The scan matching algorithm is accelerated using k-d trees when looking for correspondences between points. The computational cost is greatly improved when this method is incorporated into the search mechanism.

The proposed method is presented in Section 4.2.4, but it is necessary to explain some concepts before. First, a preprocessing step that improves the computational efficiency and makes it possible to work with huge 3D scans is detailed in Section 4.2.1. After that, the acceleration method based on k-d trees is presented in Section 4.2.2. Finally, the fitness function that incorporates the improvement mechanisms, which is the key element of our method, is introduced in Section 4.2.3.

4.2.1 Preprocessing: data reduction

Working directly with the 3D data provided by the laser scan is not efficient from a computational point of view. A 3D laser reading can be composed of more than 45000 points when using maximum resolution. It is impossible to obtain an acceptable computational time with the original size of the scan. Therefore, it is necessary to reduce the scan size to an optimal value by implementing a preprocessing phase. Besides, if the data reduction is done in an appropriate way, there is no diminution in the scan matching quality.

However, it is interesting to have such a resolution for modeling purposes. The 3D scan size will be reduced in order to obtain a better computational cost, but without losing the capability of obtaining a good matching between scans.

The following steps are applied obtaining a better size from a computational point of view with no reduction in the scan matching quality:

- Size reduction: the initial 3D point cloud can be too dense (too much information). Imagine that the resolution is $0.25°$ in the horizontal plane (pan) and $1°$ in the vertical plane (tilt) for the MANFRED-2 laser scan. The whole laser scan contains approximately 45000 points. The laser scan initial resolution is reduced by a factor R. A typical value for R is 0.1, which means that the 90% of the initial data are discarded, considering only the 10% of the points.

- Median filter: it is a nonlinear digital filtering technique that is often used to reduce noise. It does not change the data size, but it removes the outliers. This tool is frequently used in digital image processing because it can preserve edges while the noise is removed.

- Distance filter: multiple data points located close together are joined into one point. If it is used with all data, it is too heavy from a computational point of view. But it can be applied after the size reduction stage. First, the euclidean filter is applied to slices depending on the tilt planes. The result is a distance filter not only in slices but in the whole scan.

- Range Correction: it is necessary to eliminate the readings that represent the maximum range of the sensor. They present extreme values and are not reliable. For example, the maximum range that is accepted when using the MANFRED-2 laser scanner is about 15 m.
- Segmentation: there are points in the laser scan that do not represent any reliable feature. It can be interesting to make a segmentation in order to eliminate those elements that are far away from the rest. For example, only segments of at least three points located close together could be considered for scan matching.

The benefits of the preprocessing step are shown in the following example. The scan matching method is applied to two different scans composed of 16744 points. The computational time using the original method without data reduction is 53.002 s, while the same time is equal to 1.086 s when introducing data reduction (the time needed to reduce the data is also included). Both scans were reduced to 1100 points.

Similar results are obtained using different scans. The computational cost is decreased by 50 times (approximately).

4.2.2 Acceleration using k-d trees

Working in a 6D space is a heavy task from a computational point of view, thus it is very common to accelerate scan matching algorithms using different methods. Most matching methods require to search corresponding points between scans. This search is the bottleneck from a computational point of view. The most widely used acceleration approaches modify the correspondence search using trees.

A tree is "a data structure that emulates a hierarchical tree structure with a set of linked nodes". The scan matching method developed in this work has been modified including a type of tree called k-d tree. This mechanism accelerates the scan matching method in an efficient way.

The k-d tree is defined as "a binary tree in which every node represents a k-dimensional point". The nodes can be classified as leaf or non-leaf. An hyperplane is generated from every non-leaf node dividing the space into two parts (subspaces). In other words, two branches that represent subspaces have its origin in a non-leaf node. The left subtree is composed of the points located to the left of the hyperplane. The tree is generated as follows: every level (node) in the tree is associated with a cartesian coordinate. An hyperplane that is perpendicular to that dimension's axis is used to generate the subtrees. For example, if the node with coordinates (x_n, y_n, z_n)

is associated with the x-axis, the left subtree will contain the points with x lower than x_n. In this case, the hyperplane is generated using x_n.

An illustrative example of a 2D k-d tree is shown in Figure 4.4. It can be observed that the tree is built at different levels. At the beginning, the median value according to the x dimension is chosen as starting node $(7, 2)$. The space is divided into two subspaces. The left branch is composed of points with x lower than 7 and the rest of the set will be in the right branch. After that, the median value according the y dimension is calculated for both subspaces resulting in points $(5, 4)$ and $(9, 6)$. Finally, the median is calculated again for those subspaces composed of more than one point.

This is an iterative process that is repeated until the tree has been completely built, which means that all branches end in a single point. If the number of dimensions is higher than two the tree is built in the same way, but considering all dimensions when dividing the space into hyperplanes. An example of the 3D k-d tree splitting process is shown in Figure 4.5.

Although there are many different ways to construct k-d trees because the planes can be split by different methods, the canonical algorithm has the following constraints:

- The subspaces are created by selecting the median of the points that belong to

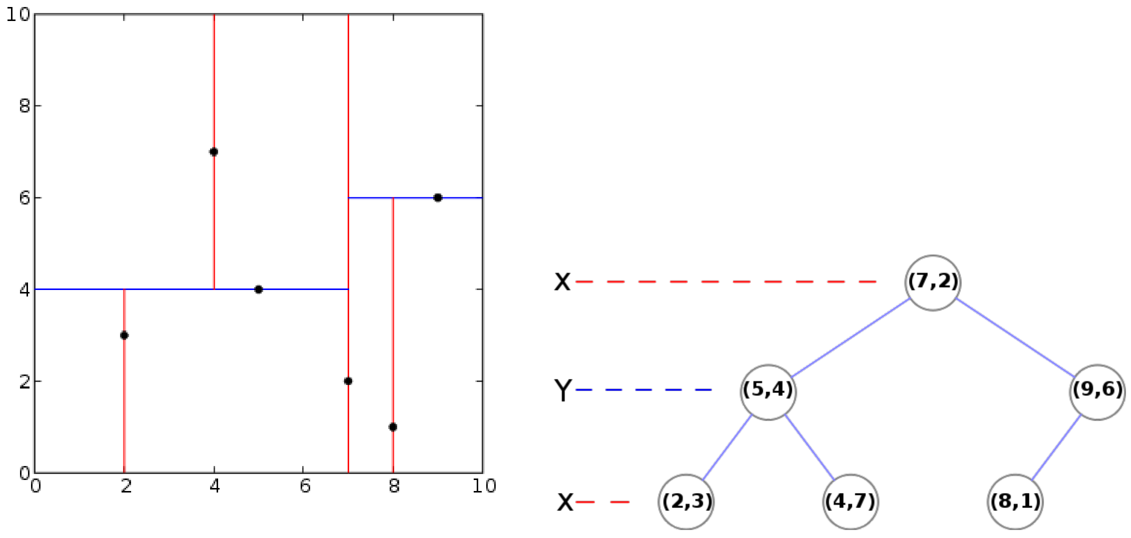

Figure 4.4: 2D k-d tree. Left: six points in a 2D space. Red and blue lines define subspaces. Right: k-d tree. Source: `http://en.wikipedia.org/wiki/K-d_tree`.

the corresponding subtree. The median is calculated according the axis that is used to create the splitting plane.

- Each level of the tree corresponds to one dimension and the axes are chosen cyclically to select the splitting planes. For example, in a 3D k-d tree, the first subspaces will be generated depending on the x coordinate of the median point, the next level will depend on the y coordinates, and the next one will consider the z coordinates. After that, the x coordinates are used again.

The k-d tree created in this way is usually balanced, which means that the distance from the leaf nodes to the root node is approximately the same. However, there are some applications where the balanced trees are not the best choice.

When the associated coordinate in the splitting axis is equal to the median value, the point is usually placed in the right subtree. It is also possible to define an additional function that takes into account different dimensions. Besides, these points can be included in both branches of the tree.

The time complexity when building a k-d tree from n points is $O(n \log^2 n)$ if an algorithm of time complexity $O(n \log n)$ is used to sort the data and compute the median. The time complexity is reduced to $O(n \log n)$ when the linear median-finding algorithm proposed by Cormen *et al.* [107] is applied.

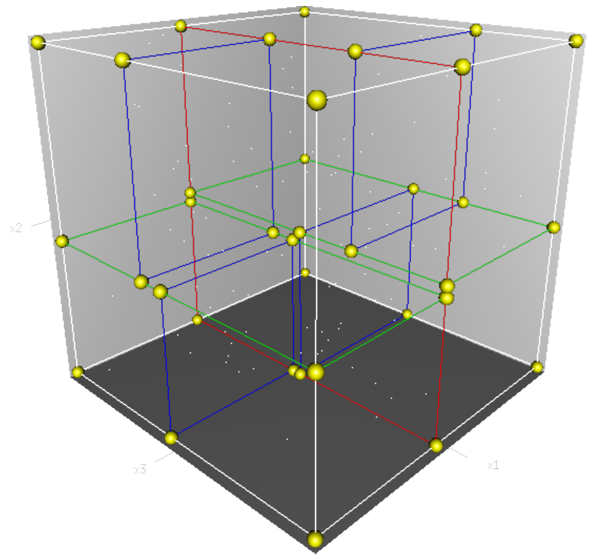

Figure 4.5: 3D k-d tree building process. First, the original cell (white) is divided into two subspaces (red plane). After that, each subcell is split (green) into two subcells. Finally, each of those four is split (blue) into two subcells. Source: `http://en.wikipedia.org/wiki/K-d_tree`.

The scan matching methods need to find the corresponding points between scans in order to compute the cost function to minimize. This process, which is equivalent to a Nearest Neighbour (NN) search, is accelerated by the k-d trees.

The purpose of the NN algorithm is to find the point in the k-d tree which is closer to a given input point. The search efficiency of the NN algorithm is improved by eliminating large portions of the search space. This method is described in the following lines:

1. The NN starts in the root node. The input point and the root node are compared. The input point is situated in one branch of the tree depending on the split dimension. After that, the next split dimension is considered to choose the next subtree (the search algorithm moves down the tree). This process is equivalent to a new point insertion.

2. The first point is executed until a leaf node is reached. After that, the leaf node is saved as the "current best".

3. The algorithm moves up the tree executing the following steps at each node:

    (a) The distance between the current node and the input point (d_{node}) is compared to the distance between the current best and the input point (d_{best}). If $d_{node} < d_{best}$, the current node becomes the current best.

    (b) The other side of the splitting plane is checked to detect if there are points in the other subtree that could be closer to the input point than the current best. This is done by a simple comparison between the distance from the input point splitting coordinate to the current node splitting coordinate and the distance from the input point to the current best.

        i. If the subtree can contain closer points, the algorithm moves down the tree and the first step is executed again.

        ii. If the subtree have not closer points, the algorithm moves up to the next node and the entire branch is eliminated.

    (c) The search is completed when this process is finished for the root node.

The NN search is highly accelerated by this method. The time complexity of the original NN search without k-d trees is $O(n)$, while it is $O(\log n)$ when using this tool. The worst case for a k-d tree containing n nodes according to the work by Lee and Wong [108] is given by the following equation:

$$t_{worst} = O(kn^{1-\frac{1}{k}}). \tag{4.8}$$

The "curse of dimensionality" [109] refers to the fact that "some problems become intractable as the number of the variables increases". It appears in high dimensions. The search algorithm needs to visit many more branches than in lower dimensional spaces. This fact depends on the number of points. When the number of points is only slightly higher than the number of dimensions, the accelerated algorithm is only slightly better than the original method. In general, the NN search is efficiently accelerated by k-d trees when the following condition is satisfied:

$$n >> 2^k. \tag{4.9}$$

Otherwise, most of the points in the tree are visited and the efficiency is not improved. There are approximate NN methods that can be used in these cases.

Besides, it is very simple to modify this algorithm in order to obtain the solution of the K-Nearest Neighbours (KNN) problem, which consists of calculating the nearest neighbours given an input point. The solution of this problem is computed by keeping the K current bests instead of just one. This mechanism will be used by our scan matching and loop detection methods. Although K was chosen before to represent the dimensionality, it is now used to define the number of neighbours. The same name has been kept because it is the standard choice in the literature.

It is also possible to obtain approximate solutions to make the algorithm run faster. For example, the number of points to be examined or the execution time can be limited. The NN search can be stopped when the distance to the input point is equal to zero. The approximate NN methods are useful in real-time applications such as robotics because the computational cost is highly improved.

More information about k-d trees can be found in the work of Berg *et al.* [110].

4.2.3 Fitness function

The fitness function is the key component of the scan matching algorithm. It will be explained in this section.

The starting point of the cost function is the well known ICP algorithm cost expression proposed by Surmann *et al.* [111]. This expression has been chosen because the cited authors define this problem with simplicity and they obtain a reliable solution. The cost function compares the previously acquired scan (model) to the last acquired one (data). Therefore, the inputs of the fitness function are two different sets of 3D points:

- Model: $M = \{m_1, ..., m_{N_m}\}$.
- Data: $D = \{d_1, ..., d_{N_d}\}$.

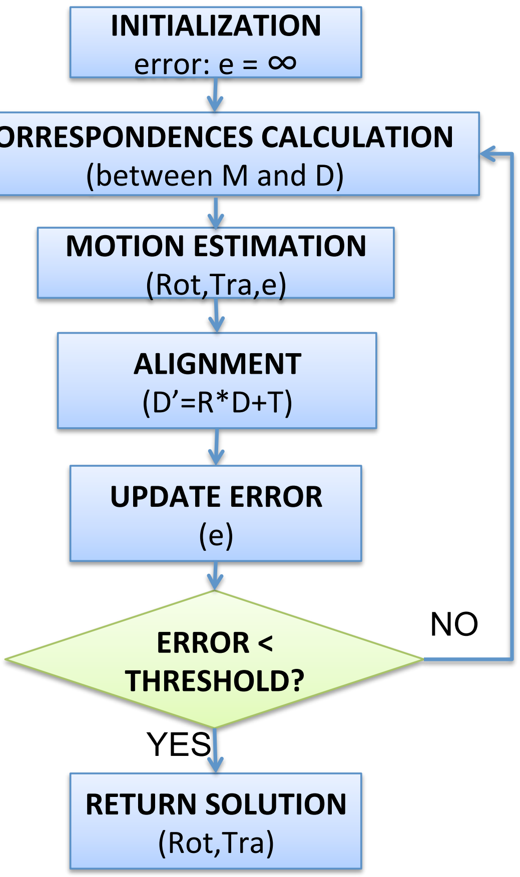

Figure 4.6: ICP flow chart.

The ICP method “iteratively revises the transformation (**Rot**,**Tra**) needed to minimize the distance between the points of two raw scans” (M and D). The cost function to minimize is the following one:

$$e(\mathbf{Rot}, \mathbf{Tra}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \|\mathbf{m}_i - (\mathbf{Rot} \cdot \mathbf{d}_j + \mathbf{Tra})\|^2, \tag{4.10}$$

where $w_{i,j}$ is equal to 1 if i nd j are corresponding points. It means that they describe the same point in space. If they are not corresponding points, $w_{i,j}$ is 0.

A flow chart of the original ICP method [24] can be seen in Figure 4.6. A brief explanation is given below. The optimization is achieved according to the flow chart.

First, the cost function (e) is initialized to infinite. The corresponding point within the data is calculated for each point of the model in a second step. This search is done according to the concepts explained in Section 4.2.2.

After that, the motion is computed. The rotation matrix and the translation vector that minimize the cost function must be estimated. Any optimization method (steepest descent, conjugate gradient...) can be used to find the rotation and translation. A more efficient method that is called "dual number quaternion" has been proposed by Zhang [63]. A basic version of the ICP algorithm that uses the Newton-Raphson method has been implemented in order to test and compare our method. This version has been developed by Per Bergström and it is available online [6].

Each point of the data set is displaced in the next stage according to the following equation:

$$\mathbf{d'}_j = \mathbf{Rot} * \mathbf{d}_j + \mathbf{Tra} \tag{4.11}$$

Finally, the data set is updated according to the new location ($D \leftarrow D'$) and the error is calculated again using the displaced data. The new error is compared to a threshold, resulting in two options:

- $e > threshold$: the results are not good enough and the ICP algorithm must continue its execution using the displaced data. The correspondences calculation step is executed again with the displaced data.
- $e \leq threshold$: the algorithm finishes and the results are returned.

This is an iterative process that is executed until the error is below a pre-specified threshold. An upper limit of iterations can also be utilized. The ICP version implemented by Bergström stops the algorithm after 40 iterations.

However, the scan matching method proposed in this document is not ICP-based but DE-based.

The registration has been considered as a local process, thus the model set is the last scan that has been successfully incorporated into the map.

The cost function given by Equation 4.10 has been improved by several mechanisms in order to obtain an effective scan matching method.

The corresponding points computation (NN search) is accelerated using k-d trees. The objective is to find the corresponding point within the data (represented by a k-d tree) for each point of the model.

The correspondences are introduced as weights ($w_{i,j}$) in the error function. These weights have a very important influence on the scan matching process. The method proposed here exploits this idea by modifying the weight selection criterion to improve its capabilities.

If the number of correspondences is equal to C and the distance between two corresponding points is denoted by $d(\mathbf{m}_{i_c}, \mathbf{d}_{j_c})$, the cost function 4.10 can be rewritten as

[6] `http://www.mathworks.com/matlabcentral/fileexchange/12627-iterative-closest-point-method`.

$$e(\mathbf{Rot}, \mathbf{Tra}) = \sum_{c=1}^{C} d(\mathbf{m}_{i_c}, \mathbf{d}_{j_c})^2. \tag{4.12}$$

The next modification exploits the idea introduced by Triebel *et al.* [27], which separates the scan in different types of points, only matching points that belong to the same class. The set of points is divided into walls, horizontal planes, and obstacles:

$$e(\mathbf{Rot}, \mathbf{Tra}) = \underbrace{\sum_{c=1}^{C_1} d(\mathbf{m}_{w,i_c}, \mathbf{d}_{w,j_c})^2}_{walls} + \underbrace{\sum_{c=1}^{C_2} d(\mathbf{m}_{p,i_c}, \mathbf{d}_{p,j_c})^2}_{planes} + \underbrace{\sum_{c=1}^{C_3} d(\mathbf{m}_{o,i_c}, \mathbf{d}_{o,j_c})^2}_{obstacles}, \tag{4.13}$$

where subindex w is used to express that a point belongs to a wall, p represents horizontal planes (unobstructed floor), and obstacles are denoted by o.

The type of a given point is easily computed following the next steps:

1. KNN search: the nearest neighbours of the point within the scan are calculated. This search is also accelerated by using k-d trees. The number of neighbours has been empirically fixed to 10.

2. Plane estimation: the plane that best fit the previously obtained neighbours is calculated. A plane P_i is defined by a point $\vec{p_i}$ and the direction cosines of the normal:

$$P_i = \{\vec{p_i}, \cos\alpha, \cos\beta, \cos\gamma\}. \tag{4.14}$$

   The different types are calculated according to the cosines:

   (a) $\gamma > 80° \rightarrow$ Wall.

   (b) $\gamma < 10° \rightarrow$ Horizontal plane.

   (c) otherwise $\rightarrow$ Obstacle.

The error function only considers points that belong to the same types. Different types can be chosen depending on the environment characteristics. This segmentation is suitable for structured indoor environments which are mostly composed of walls, horizontal planes, and obstacles.

The next idea consists of choosing adequate weights for each type. In the previous expressions, the weight factor was assigned to 1 when corresponding points were found. However, it can be interesting to give higher weights to those points that represent more important characteristics.

Analyzing point types, the most important one in an indoor environment is the vertical cell. The obstacles are characteristics that uniquely define the environment, but their importance depends on the number of obstacles and their size. In general, there are more vertical points than obstacles points in a frame. It is not always possible to distinguish between places taking into account only traversable zones.

If Equation 4.13 is modified by introducing $w_{i,j} = 2$ for walls and $w_{i,j} = 0.5$ for the other types, the error function is now equal to

$$e(\mathbf{Rot}, \mathbf{Tra}) = \underbrace{\sum_{c=1}^{C_1} 2d(\mathbf{m}_{w,i_c}, \mathbf{d}_{w,j_c})^2}_{walls} + \underbrace{\sum_{c=1}^{C_2} 0.5d(\mathbf{m}_{p,i_c}, \mathbf{d}_{p,j_c})^2}_{planes} + \underbrace{\sum_{c=1}^{C_3} 0.5d(\mathbf{m}_{o,i_c}, \mathbf{d}_{o,j_c})^2}_{obstacles} . \tag{4.15}$$

An improvement in the distance between corresponding points that belong to walls will have more influence on the error function. It is important to remark that these values depend on the environment. It will be crucial to choose adequate weights in order to obtain good results.

The cost function represented by Equation 4.15 is the engine of the DE-based scan matching method. This method is explained in the following section.

4.2.4 DE-based scan matching (6 DOF)

A particle-based evolutionary algorithm evolves with time to the solution that yields the cost function lower value. If the cost function is properly chosen, it is possible to solve the scan matching problem. The cost function basic idea is to compute the distances between corresponding points when they belong to the same type of surface. An error is computed by the cost function that is given by Equation 4.15.

This scan matching technique is similar to the localization filter detailed in Chapter 3. If the reader is familiarized with this technique, this section can be skipped. The contents of these paragraphs can be a bit repetitive. A brief explanation is given here, so the reader can understand the full contents of this chapter without having read the previous one. The differences between both methods are highlighted.

Given two sets of 3D points (model and data), the DE-based algorithm calculate the rotation and translation that is necessary to apply to the data set in order to maximize the matching between them.

One difference with respect to the ICP-based version is that the solution is now represented by the robot's pose, considering that the robot is situated in the origin of the data scan. It means that this method calculates the robot's displacement in order to find the best matching between data and model. After that, the registration is done by moving the data set according to the estimated solution.

Algorithm 4 DE-based Scan Matching

```
1: for $i = 1 : N_P$ do
2:     $pop_i^1 \leftarrow init\_pop(data\_initial\_pose)$          ▷ First population generation
3:     $e^0(i) \leftarrow cost(model, data, pop_i^1)$          ▷ Cost function calculation
4: end for
5: for $k = 1 : max$ do
6:     for $i = 1 : N_P$ do
7:         $v_i^k = pop_a^k + F(pop_b^k - pop_c^k)$          ▷ Mutation
8:         for $j = 1 : D$ do
9:             $u_{i,j}^k = v_{i,j}^k, \forall p_{i,j}^k < Cr$          ▷ Crossover
10:            $u_{i,j}^k = pop_{i,j}^k, \forall p_{i,j}^k \geq Cr$
11:        end for
12:        $e^k(i) \leftarrow cost(model, data, pop_i^k)$          ▷ New cost function calculation
13:        if $e^k(i) < e^{k-1}(i) * \tau$ then          ▷ Selection with Thresholding
14:            $pop_i^{k+1} = u_{i,j}^k$
15:        else
16:            $pop_i^{k+1} = pop_i^k$
17:        end if
18:    end for
19:    $pop^k = disc(pop^k)$          ▷ Discarding
20:    $ind\_best \leftarrow min(e^k)$
21:    $bestmem \leftarrow pop^k(ind\_best)$
22:    if $convergence = true$ then          ▷ Execution stops after convergence
23:        $exit(bestmem)$
24:    end if
25: end for          ▷ Return best estimation
```

The stochastic search of the matching pose is done using the DE method for global optimization problems over continuous spaces. The reader can see Algorithm 4 to understand the main concepts. Besides, a complete explanation of the DE algorithm can be found in Chapter 3 or in our previous work [45, 81].

The search starts with a population of N_P candidates (the number of elements is represented by N_P). Each candidate is a possible solution. The robot's pose has 6 DOF because the robot is situated in a 3D world:

$$pop_i^k = (x_i^k, y_i^k, z_i^k, \varphi_i^k, \theta_i^k, \psi_i^k), \tag{4.16}$$

where pop_i^k represents element i at iteration k. The position is given by the cartesian coordinates and the orientation is defined by the Euler angles.

The initial population will be chosen randomly but situated in a sphere close to the pose estimate given by the odometry information.

The DE-based scan matching is simpler than the DE-based global localization because there is an initial estimation of the robot's pose. In global localization, the initial population was randomly generated covering all the map because there was no *a priori* information about the robot's location. The problem here consists of correcting the robot's pose obtained by the odometry sensors, thus an smaller population is generated around the original pose. Besides, the number of iterations to convergence is also lower because the population is located close to the solution.

For each candidate, its associated cost function is calculated (line 3 of Algorithm 4). The cost function defined by Equation 4.15 computes correspondences and data types and assigns different weights to different types. The nearest neighbour search has been efficiently accelerated using k-d trees.

The main loop starts in line 5 and it is repeated until the maximum number of iterations is reached or one of the convergence conditions is satisfied. Another loop which contains the evolutionary search starts in line 6. It consists of the generation of a new population for the next generation. In a single iteration the algorithm is executed to obtain the next candidates, evolving in time to the correct pose.

The initial population is perturbed to generate a variation v_i for each population member:

$$v_i^k = pop_a^k + F(pop_b^k - pop_c^k), \tag{4.17}$$

where pop_a^k, pop_b^k, and pop_c^k are three randomly chosen elements at iteration k and a, b, and c are different from running index i. F is a real and constant factor which controls the amplification of the differential variations $(pop_b^k - pop_c^k)$. The variables of this expression can be changed to obtain a different perturbed vector. For example, instead of pop_a^k, it is possible to select the best element of the population.

In order to increase the diversity of the new generation, the crossover is introduced. The trial vector is denoted by $u_i^k = (u_{i,1}^k, u_{i,2}^k, \ldots, u_{i,D}^k)^T$ and its parameters depend on the crossover probability:

$$u_{i,j}^k = \begin{cases} v_{i,j}^k; & \text{if } p_{i,j}^k < Cr, \\ pop_{i,j}^k; & \text{otherwise,} \end{cases} \tag{4.18}$$

where $p_{i,j}^k$ is a randomly chosen value from the interval $[0, 1]$ for each parameter j of the population member i at iteration k, and Cr is the crossover probability that constitutes the crossover control variable. D, which represents the number of chromosomes (number of components of the population element), is equal to six because the robot's pose has six DOF. The random values $p_{i,j}^k$ are made anew for each trial vector i.

The new candidate u_i^k is compared to pop_i^k to decide which element should become a member of generation $k + 1$. If u_i^k yields a better value for the objective fitness function than pop_i^k, then it is replaced by u_i^k; otherwise, the current member pop_i^k is retained for the next generation.

The selection mechanism has been modified by adding a thresholding band that avoids the premature convergence in noisy optimization problems. The idea of the thresholding is to reduce the eagerness of the algorithm by rejecting those new solutions that do not improve the previous hypothesis in a pre-specified magnitude τ. The threshold cannot be a fixed magnitude, because this unit depends on the noise variance and the fitness distance to the optimal fitness value. The noise variance might be estimated, but it is not easy to estimate the second factor.

The previous process (mutation, crossover, and selection with thresholding) is applied to the whole population obtaining the next generation population $(k + 1)$.

The convergence speed is decreased by the thresholding band. This aspect is particularly important at initial stages. A discarding mechanism has been introduced to increase the algorithm speed while maintaining the thresholding advantages. The idea is to determine the worst fitness individuals of the new population and substitute them by new solutions that are situated close to better ones. In order to do so, a percentage of elements to be discarded is chosen (5%). To avoid concentrating the discarded solutions around the best existing individual, one of the members of the population with its fitness value located in the first half of the fitness ranking is randomly selected. This selected solution plus a relatively small random component is adopted as a new offspring.

Finally, the algorithm returns the best member of the population. This member is the robot's pose that minimizes the difference between model and data (scan matching solution). The data set is moved and rotated according to this solution, the registration is successfully accomplished, and the updated data are incorporated into the model.

4.2.5 Simplified version: plane scan matching

Although the previously explained method is robust and its efficiency in 3D environments with six DOF will be widely demonstrated with experimental results, a different and simplified version has also been tested. A short description of this tool is given in this section. This version can be very useful in some cases and it has some associated advantages. The available data are 3D, but it is possible to consider 2D frames for scan matching.

The use of 3D data in mapping has an important drawback which is the computational cost. Nevertheless, some simplifications can be assumed when the robot movement does not cover all DOF and some components of the pose vector remain constant. Imagine a mobile robot equipped with a 3D laser scanner fixed in its base. The relative distance between scanner and robot is constant. This robot is situated

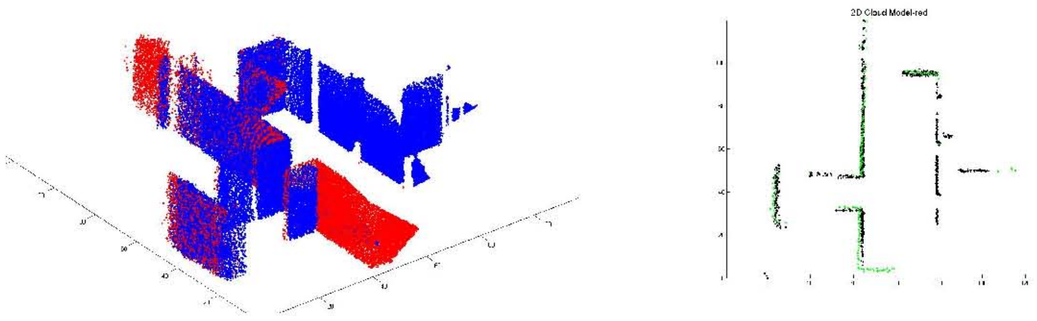

Figure 4.7: Scan matching example with 2D data. Left: 3D view. Right: horizontal projection. Seven different scans are joined into the model. Elapsed time: 1.05 s. Laser noise: 1%.

in an indoor environment which is mostly composed of flat surfaces. In this case, the movement can be restricted to 3 DOF that represent position (x, y) and orientation (yaw) in a plane, thus the scan matching can be simplified to these coordinates.

A modified version that works in the horizontal plane has been implemented in order to test it. A 3D data set is obtained by the laser scan, but only one plane is considered when matching (z, roll, and pitch are constant).

One advantage of this simplified version is that the computational cost is drastically reduced. Therefore, it is possible to use more information. For example, the whole model or a part of it can be included instead of a single scan. It means that the new scan can be matched with the whole model or the last acquired scans. The process is converted into global scan matching, with its advantages and disadvantages, and the computational cost is not prohibitive. Since a single mistake produces an irrecoverable error in global registration, only the local version has been implemented in this work.

The time requirements (in seconds) are in the interval $[1, 2]$ with 3D data and $[0.2, 0.5]$ with 2D measurements. If the process is not local the computational resources needed grow very fast, making the first option unfeasible from a computational point of view.

The following example illustrates the simplification adopted in this section. The ICP-based scan matching algorithm has been applied in a simulated indoor environment. In Figure 4.7, the results with 2D scans are presented. Figure 4.8 shows the results with 3D frames. Seven different scans has been registered in each figure. There is not a big visual difference between the 3D views. However, it can be appreciated that the horizontal projection of the 2D option has a lower error, which seems to be an additional advantage.

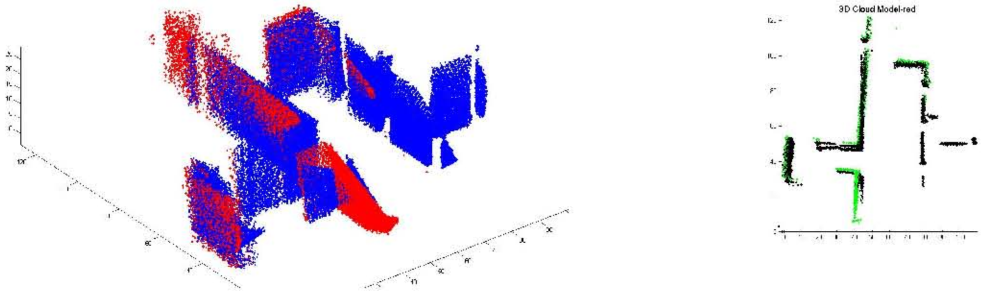

Figure 4.8: Scan matching example with 3D data. Left: 3D view. Right: horizontal projection. Seven different scans are joined into the model. Elapsed time: 9.76 s. Laser noise: 1%.

The simulated map is an indoor environment which is mostly composed of offices and corridors. Its principal components are parallel or perpendicular. The results when using 3D data are referred to six dimensions, thus the relation between model and data is described by six components: cartesian coordinates (x, y, and z) and orientation angles (roll, pitch, and yaw). The results when using 2D data are described by three components: x, y, and horizontal orientation. It has to be remarked that the horizontal orientation (yaw) is the parameter that actually changes in this type of environment, not the other orientation parameters (roll and pitch).

It is straightforward that a small error in roll and pitch angles can produce an appreciable error in the map because they do not change at all. In this type of environment, the plane version considers the angle that actually changes. That is the reason why the best results are obtained with 2D data. As a conclusion, it can be said that the plane option is suitable for this environment. However, it cannot be generalized because it has been applied to a very specific situation.

Finally, it has to be remarked that the plane mapping has been widely studied and it is not the purpose of this work. The 3D modeling, with its characteristics and applications, has been studied here, and we focus on the aspects related to this problem.

4.2.6 Scan matching associated difficulties

It is very difficult to obtain satisfactory results with scan matching techniques in some cases. It occurs when there is not information enough to match both frames or this information causes a wrong matching. This difficulties are described here to give a complete explanation of the method and its problems.

First, strong changes in orientation (up to 90° has been tested) make it difficult to apply scan matching techniques because both scans (model and data) have not similarities enough and it is not possible to match them.

Second, the scan matching in environments with repetitive and similar forms is very laborious. For example, imagine a typical indoor environment composed of a narrow long corridor and offices on both sides. Many locations along the corridor have the same appearance, thus it is very difficult to distinguish between them. It is not a scan matching shortcoming but a perception problem, because it is not possible to solve it with the current information provided by the sensors.

An example of the second case is shown in Figure 4.9. The mobile robot is located in a long corridor without features. The purpose is to match two different scans took from different positions (the distance between them is equal to one meter), but the perceptive information obtained from both locations is very similar. After the scan matching process, the result indicates that both scans have the same origin, which means that the robot is situated in the same place.

It can be very interesting to apply feature detectors (for example, corner and edge detectors) when the scan matching fails. It can solve the previously mentioned shortcomings.

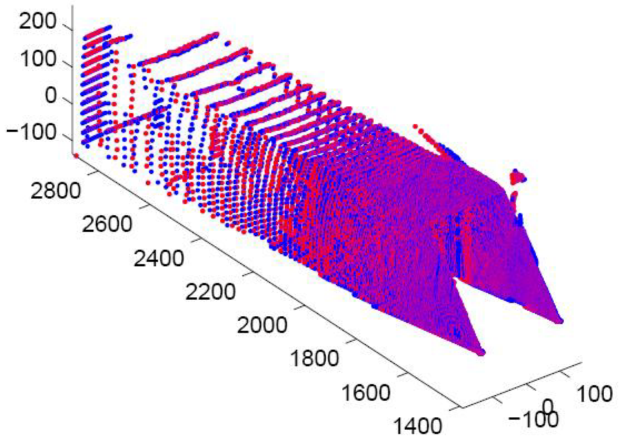

Figure 4.9: Scan matching associated difficulties. The environment is a long corridor and the measurements obtained from two different locations are very similar. The (wrong) result indicates that both scans have the same origin. All units in cm.

4.3 Loop Detection

Being able to detect and correct loops is fundamental for autonomous navigation. The maps generated after scan matching are not consistent enough to conclude that the mapping task has been successfully achieved. This consistency will be given by detecting the known places (loop detection) and eliminating the accumulated error when it exists (loop closure). If the pose estimate has an small orientation error at the beginning of the robot's path, the accumulated error after some time can be very important.

Analyzing the relaxation or loop closure problem, there are some algorithms that obtain robust and reliable solutions, such as TORO [6] or Simultaneous Matching [7]. They distribute the accumulated pose error of pairwise registered scans in order to build a consistent map once the robot has detected that it is in a pre-visited place. However, the loop detection task is still an open problem in robotics. That is the reason why this topic has been studied in this thesis.

This tool has been implemented to improve the map consistency. It is possible to design a feature-based or a point-based algorithm (like scan matching). The second option has not been considered here for several reasons. The scan matching algorithms are based on the minimization of a cost function that is an estimate of the matching quality. This cost function has a high computational cost when working in 3D. Since it is necessary to compare the current scan to all frames, it is not a feasible option from a computational point of view. Besides, the loop detection algorithms must be rotation invariant because a pre-visited place could be reached from different points of view. The scan matching methods present problems when working with strong changes in orientation.

4.3.1 Surface features

The method developed here is based on several ideas. The first one is extracted from the work by Magnusson *et al.* [104]. Their algorithm extracts features from two 3D laser scans and compares them in order to detect if the loop exists. Their descriptor is composed of the geometric forms of the scan (lines, planes, and spheres). Each component is equal to the number of elements (laser measurements) that belong to a feature (for example, plane with orientation).

The cited authors divide the map into a regular grid and calculate the parameters of a normally distributed probability density function describing the local surface shape. This part has been implemented here in a different way.

For each point i of the laser scan, the K nearest neighbors are efficiently searched by using k-d trees. The covariance matrix of the K points is calculated after that. The point i local surface is described by the eigenvectors ($\vec{e_1}$, $\vec{e_2}$, $\vec{e_3}$) and the eigenvalues

(λ_1, λ_2, λ_3) of the covariance matrix.

The local surface type is given by the relation between λ_1, λ_2, and λ_3. Two different variables (t_l and t_p) are defined to distinguish between surfaces:

- Line: $\lambda_1/\lambda_2 > t_l$.
- Plane: it is not a line and $\lambda_2/\lambda_3 > t_p$.
- Sphere: it is neither a line nor a plane.

t_l and t_p are constant thresholds that are used to detect lines and planes respectively.

After that, the planes and lines can be sub-divided into different types depending on the orientation, and the different spheres can be defined depending on the surface roughness. Finally, a number N_s of planar, linear, and spherical features is obtained. A good performance is achieved in the experimental results with one line, seven planes, and one sphere ($N_s = 9$). The feature descriptor defined is:

$$\vec{F} = (f_1, ..., f_{N_s}), \tag{4.19}$$

where each element j of $\vec{F}$ represents how many points belong to feature j.

The matching between scans is obtained by comparing feature descriptors. The most common method is to define a difference giving weights to each component of the descriptor. However, our approach here is treated from a different point of view.

If the matching quality is defined by a value that represents the similarities between a pair of scans, the loop detection is converted into a binary process because the loop exists when the difference is lower than a threshold. Nevertheless, the loop detection variable that we have defined is not binary but continuous. It has ben called LPI (Loop Probability Indicator).

The LPI does not literally represent the probability of matching because the feature descriptors are not probabilities at all. But it is possible to define a variable that equals 100% when the scans perfectly match and whose value decreases when the match is poorer. That is the reason why the term *probability* is included. The LPI initial definition is expressed by the following Equation:

$$LPI = \sum_{i=1}^{N_s} \frac{(\max(f_i, g_i) - |f_i - g_i|)}{\max(f_i, g_i)} \cdot \frac{w_i}{\sum_{i=1}^{N_s} w_i} \cdot 100, \tag{4.20}$$

where f_i and g_i are the components of two different feature descriptors and w_i represents the weight of each component.

The term $(\max(f_i, g_i) - |f_i - g_i|)/\max(f_i, g_i)$ of Equation 4.20 represents the similarities between both scans for a single feature. It has been normalized in order to be 1 when the matching is perfect and lower when the difference between f_i and g_i increases.

Besides, each feature weight has to be defined and normalized. It has to be higher when the number of points that belong to this feature is larger. The weight given to each feature i could be equal to

$$w_i = \max(f_i, g_i), \tag{4.21}$$

choosing the maximum value between f_i and g_i. The weights are normalized by dividing their value by the sum of the maxima of all features. Two different options have also been implemented:

$$\begin{aligned} w_i &= mean(f_i, g_i). \\ w_i &= \sum_{j=1}^{T} \max(f_i, g_i)/T. \end{aligned} \tag{4.22}$$

The first one uses the average value instead of the maximum. The second one not only considers the current frame but the past information. If the robot has received T laser frames during the exploration, each weight is the average of the T weights that are obtained for each laser scan according to Equation 4.21.

If the weights are given by Equation 4.21, Equation 4.20 can be simplified:

$$LPI = \frac{\sum_{i=1}^{N_s} (\max(f_i, g_i) - |f_i - g_i|)}{\sum_{i=1}^{N_s} \max(f_i, g_i)} \cdot 100. \tag{4.23}$$

This option will be used from now on for simplicity.

The described method considers geometric forms (we have called them *surface* features). Each component holds the number of points that belong to each feature. However, there are different types of features that cannot be added. It can be interesting to add different features (we have called them *numerical* features) and change the LPI formula in order to introduce them.

4.3.2 Numerical features

There are interesting characteristics that can be represented by a number. Some examples are: volume, curvature, average distance between points, and so on. A feature descriptor with numerical characteristics is used by Granström *et al.* [105].

Including this type of features, the new feature descriptor can be defined as

$$\vec{F} = (f_1, ..., f_{N_s}, f_{N_s+1}, ..., f_{N_s+N_n}), \tag{4.24}$$

where N_n is the number of *numerical* features.

The LPI value must satisfy the specifications: it has to be equal to 100% in a perfect match and lower when the matching quality decreases. Therefore, it is necessary to define the weights of the new features according to this fact.

Analyzing Equation 4.20, the average value of the weights is equal to

$$W = \frac{\sum_{i=1}^{N_s} \max(f_i, g_i)}{N_s}. \tag{4.25}$$

A weight equal to W will be given to the new features. Nevertheless, different weights can be calculated depending on the new features to be added. The loop indicator can be now divided into two components:

$$\begin{aligned}
LPI_s =& \frac{\sum_{i=1}^{N_s}(\max(f_i, g_i) - |f_i - g_i|)}{[\sum_{i=1}^{N_s} \max(f_i, g_i)](1 + \frac{N_n}{N_s})} \\
LPI_n =& \sum_{i=N_s+1}^{N_s+N_n} \frac{\max(f_i, g_i) - |f_i - g_i|}{\max(f_i, g_i)} \\
& \cdot \frac{(\sum_{i=1}^{N_s} \max(f_i, g_i))/N_s}{[\sum_{i=1}^{N_s} \max(f_i, g_i)](1 + \frac{N_n}{N_s})} \\
LPI_n =& \sum_{i=N_s+1}^{N_s+N_n} \frac{\max(f_i, g_i) - |f_i - g_i|}{\max(f_i, g_i)} \cdot \frac{1}{N_s + N_n} \\
LPI =& (LPI_s + LPI_n) \cdot 100
\end{aligned} \tag{4.26}$$

The loop indicator still meets the requirements: it is equal to 100% in a perfect match and lower when the matching quality decreases. LPI_s is the component that computes the *surface* features. It is calculated following Equation 4.23, but the weights are normalized according to the total number of features $(N_s + N_n)$ by adding the term $1 + N_n/N_s$. LPI_n is used to include the *numerical* features. Each component weight is equal to W and the expression is normalized according to the total number of features .

An advantage of the proposed method is that the weights are calculated online, not needing any machine learning algorithm to fix them like the method by Granström *et al.* [105] does. The autonomous robot mapping task is not traditionally a machine learning problem, thus the previously explained option that has been developed in this paper is a more suitable approach.

Different *numerical* features have to be considered to be a component of the descriptor. Most of them have been taken from the work by Granström. The best performance has been obtained with the following ones:

- Volume: the volumes of the individual laser beams are added to measure the volume of the scan. Each measurement is the centre of a pyramid base with its top in the sensor location (robot's position). If θ_v and θ_h are the laser vertical and horizontal resolution, the length and width of the pyramid base are $l_i = 2d_i tan(\theta_v/2)$ and $w_i = 2d_i tan(\theta_h/2)$. The height is $h_i = d_i$. The volume of the pyramid is $v_i = l_i \cdot w_i \cdot h_i/3$. It is possible to obtain a volumetric feature that does not depend on the angular resolutions, normalizing the volume of each point with the volume of d_{max}:

$$f_{N_s+1} = \frac{1}{Pv_{max}} \sum_{i=1}^{P} v_i = \frac{1}{P} \sum_{i=1}^{P} (\frac{d_i}{d_{max}})^3, \tag{4.27}$$

  where the range d_i is computed as the distance from point i to the sensor location. The total number of points is P.

- Distance: f_{N_s+2} is the sum of the distances between consecutive points: $\delta\vec{p}_i = ||\vec{p}_i - \vec{p}_{i+1}||$, where $\vec{p}_i$ is the 3D position of point i.

- Curvature: if A is the area covered by the triangle with vertexes in p_{i-1}, p_i, and p_{i+1}, and $d_{i-1,i}$, $d_{i-1,i+1}$, and $d_{i,i+1}$ are the pairwise point to point distances, the curvature at i is computed as

$$k_i = \frac{4A}{d_{i-1,i}d_{i-1,i+1}d_{i,i+1}}. \tag{4.28}$$

  f_{N_s+3} is the mean of the vector that contains the curvature for all points.

Besides, there are more candidates that could be chosen as *numerical* features:

- Average range: f_{N_s+4} is the average of d_i normalized by dividing by d_{max}.

- Standard deviation of range: f_{N_s+5} is the standard deviation of d_i normalized by dividing by d_{max}.

- Range kurtosis: it is a measure of the peakedness of the histogram of ranges. It is computed as follows:

$$m_k = \frac{1}{P} \sum_{i=1}^{P} (d_i - \overline{d})^k. \tag{4.29}$$

$$f_{N_s+6} = \frac{m_4}{(m_2)^2} - 3. \tag{4.30}$$

- Centroid: f_{N_s+7} is the mean distance form the centroid of the scan and f_{N_s+8} is the standard deviation.
- Range differences f_{N_s+9} is the mean of the range difference between consecutive points and f_{N_s+10} is the standard deviation.
- Curvature standard deviation: f_{N_s+11} is the curvature standard deviation.

Depending on the number N_n of these new features and the descriptor composition, different behaviors will be observed. It will be necessary to make a study about it, selecting the best settings.

4.3.3 Expression analysis

The final definition of LPI is given by Equation 4.26. Analyzing its value, an additional mechanism has been implemented in order to improve the method capabilities. Different bands have been defined depending on the loop detection variable value:

- $LPI \geq t_1$: positive matching. The loop is detected and the accumulated error must be minimized by a relaxation method.
- $t_2 \leq LPI < t_1$: uncertainty band. There is not enough information to conclude that the loop exists.
- $LPI < t_2$: negative matching. There is no loop in this case.

t_1 and t_2 are thresholds that are used to distinguish between bands.

An advantage of this approach is that an additional method can be applied when LPI is inside the uncertainty band, thus the loop detection quality is improved. The implemented method consists of analyzing a third scan. If the LPI value between two scans (a and b) is inside the uncertainty band, a third scan is considered and the probability is calculated again using the new scan (a and c). This is done by waiting until the robot receives a new laser reading from the next location that is explored. The loop is now detected depending on the new value that includes more information.

The minimum loop size must be also defined to complete the algorithm. If it is too small, consecutive scans will be considered as loops. Therefore, a minimum number of scans between two visits to any location must be fixed.

In addition, the LPI definition given by Equation 4.26 makes this method very flexible because it is not necessary to have any previous knowledge about the environment. It has more associated advantages. As said before, different features and values of N_n can be introduced in the formula in order to choose the best combination. It can be considered as an online learning method because it does not need any previous training or a huge available data set.

When the autonomous robot is navigating around the environment, the available information increases, and the LPI weights can be changed to obtain a better performance. The features average or standard deviation would be a better approximation if there is enough information. For example, the features average of the past scans can be chosen. This option has been implemented in this work, as can be seen in the second line of Equation 4.22.

After several pair comparisons, an estimation of the most important features in the matching process can be done, and the weights of the most important features can be increased while decreasing those of the least important ones.

The described method is adaptive and its weights will change online while the robot is navigating and acquiring new information. Its adaptation to changes of the environment is automatic. For example, if the robot is working on an indoor environment and it navigates to an outdoor place, the adaptive weights change while it is navigating and the adaptation to the new environment is done online, not being necessary to make any additional calibration.

Chapter 5

Experimental Results

This chapter has been divided into two different sections that demonstrate the efficiency of the proposed tools and show their associated characteristics. The RELF-3D localization filter experiments are presented in in Section 5.1, while the experiments related to the mapping tools (scan matching and loop detection) are shown in Section 5.2.

5.1 Global Localization in 3D Environments

All the experiments have been developed in a simulated indoor environment: laboratories, corridors, and offices of the Department of Engineering Systems and Automation of the UC3M. An example of the 3D map can be found in Figure 5.1. The measurement model is the same that was explained in Section 3.2.4.

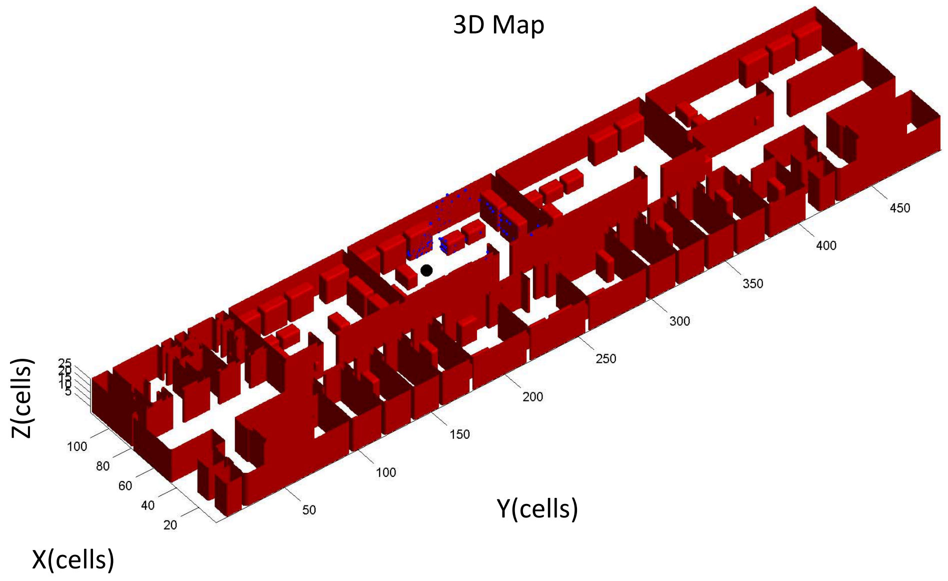

Figure 5.1: 3D simulated map.

The following experiments have been conducted to demonstrate different aspects. The first one, that is shown in Section 5.1.1, tests the capacity of this method to solve the global localization problem. After that, the method accuracy is calculated in Section 5.1.2. The effect produced by the thresholding and discarding mechanisms is studied in Section 5.1.3. In Section 5.1.4, different fitness functions are compared. The complexity and computational cost are computed in Section 5.1.5. Finally, the results of the initial population estimation method are presented in Section 5.1.6.

The parameters used by the DE-based localization algorithm are $F = 0.85$ and $C_r = 0.75$. The laser error is Gaussian-distributed with a 3% average standard deviation over the distance weighted ($\sigma = 0.03 * distance$). The threshold τ is equal to $S_N f_i^k = 0.03 f_i^k$, where S_N is the sensor noise (percentage over the distance weighted) and f_i^k is the fitness function value for the ith population member in the kth iteration. The discarded population is equal to the 5%. The population size has been empirically adjusted in all cases except in Section 5.1.6 where it is generated by the initialization tool.

5.1.1 Effectiveness of the algorithm

In the first experiments, we have tested the capacity of our method to solve the global localization problem. In order to do that, we must tackle the problem of localizing the robot in a random place of the environment only with the information of the sensors at this point (3D data obtained with the laser range finder and movement information).

First, we will consider that the robot is still. The way to check it consists on running the algorithm a number of times, situating the robot at different places, and obtaining the percentage of success. The global localization process starts with the robot situated in a random place of the environment and two different cases and variables define the process: the estimate matches the real pose (the variable *true* is incremented by one) or the estimate and the real pose do not coincide (the variable *false* is incremented by one). After that, a new random place is generated and the previous process (which is defined as a single trial) is repeated. The total number of trials is *total*. The percentage of success is equal to *true* divided by *total*. We have chosen 100 as the number of elements of the population. The laser range finder has an error of 3% over the distance weighted, which is worse than the noise of real devices. The commercial devices have a noise which is approximately equal to a cm. We have not considered the convergence of the algorithm in this experiment, and we let the algorithm run a high number of iterations (500) to see whether it can obtain the solution or not.

The success of running the program under these conditions is 84%. This percentage depends on the environment and our map contains many similar places whose

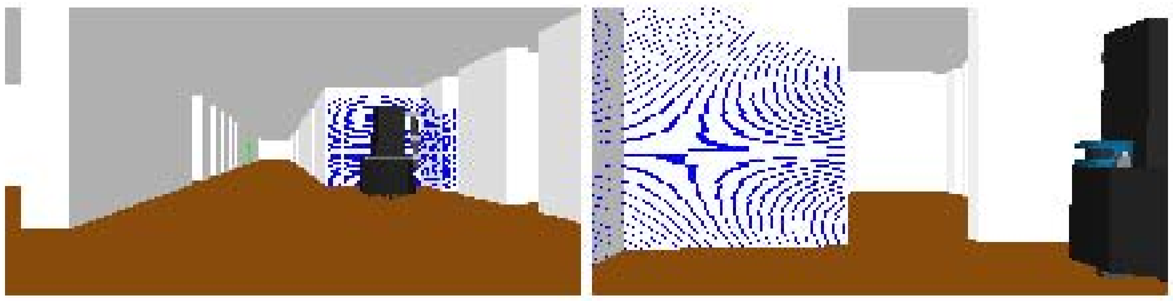
Figure 5.2: Example of localization in a corridor.

correspondent measurements are the same. Therefore, it is not possible to distinguish between several different locations because the sensor measurements are the same. For example, it occurs when the robot is inside an office. That is why the algorithm does not succeed in some cases and the localization method converges to all possible solutions (correct place and similar locations). We consider it as a failure because it does not converge to a single solution. The algorithm returns one solution but it could be a wrong place. It can also be explained using the human beings perception. If you are in an office corner looking at the wall it is not possible to know in which office you are. It can be concluded that the information is not good enough to apply localization techniques in these cases.

This is a promising result, because we have not included the movement of the robot (using a single *step* to localize the robot, with information of one 3D laser scan and no motion information), but it does not reflect a real situation. The purpose of the experiment is to check the capability of our method to solve the global localization problem. Therefore, the RELF-3D algorithm is suitable for solving this. An example of the information that the robot would receive is shown in Figure 5.2, where the 3D environment, the laser readings, and the robot are represented.

However, we will not find this situation in real life, because the robot will not be still. We have also tested the capacity of our method under movement, making different experiments with changes in the maximum iterations per *step*. On one hand, with a high number of iterations per *step*, we will be closer to the solution in one single *step*, but with a lower velocity because we need more time to deal with a single laser reading. As a reminder, the robot uses the first laser reading to locate, with a given number of iterations, and then the second laser reading from another location is introduced in the localization module, including motion information. On the other hand, with a low number of iterations, the robot will need less time to execute a *step*. The number of *steps* to localization against the iterations per step is shown in Figure 5.3.

The population size is 100. The convergence criteria to conclude that the robot

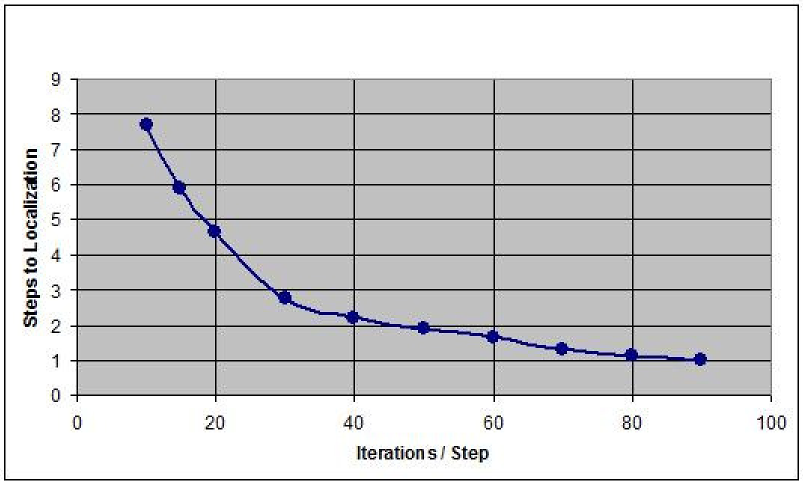

Figure 5.3: Steps to localization vs. iterations/step. Error = 3%. Convergence criteria: worst member fitness value smaller than $3N_s/2$. Algorithm options: random mutation with thresholding and discarding. $N_s = 100$.

is successfully localized is that the worst member fitness value is lower than $3N_s/2$. The average number of *steps* after situating the robot in different places is shown in the figure.

Interesting results can be concluded observing Figure 5.3. It needs more *steps* to succeed with a maximum of 10 iterations per step (lower limit chosen). Therefore, we obtain better results in other cases. The time needed (it depends on the number of iterations) to execute the localization algorithm in a *step* is less, but we do not know the situation of the robot until eight *steps*. It means that the number of laser readings used, and also the distance from the initial location to the current one, is bigger. This could be quite dangerous for the robot, because it goes a long way before being properly localized on the map. If the number of iterations per *step* is 90, the robot will need one single *step*. However, the time needed could be a problem if the time per *step* is a constraint. It can be concluded that different values can be chosen depending on robot requirements. For example, good results are obtained with 40 iterations per *step*. The robot is successfully localized quickly in two steps using a low number of iterations.

Vahdat *et al.* [46] show a similar experiment implementing three different methods: DE, PSO, and MC. However, the number of iterations per *step* is surprisingly low. They need two or three iterations in each *time-step* and about 25 *steps* for robust convergence. The experiment is different from ours because it does not limit the number of *steps* but the number of iterations. It can be because it is not dangerous for their robot and it can wait 25 *steps* until convergence. It is a different result that

Table 5.1: Errors localizing the robot at different points. The cartesian coordinates are in cells and the errors in millimeters and degrees.

	x	y	z	θ	$\hat{x}$	$\hat{y}$	$\hat{z}$	$\hat{\theta}$	e_p(mm)	e_θ(deg)
1	47	358	4	0	47.001	358.000	4.039	0.000	4.76	0.000
2	75	215	14	90	75.004	214.984	13.991	90.016	2.24	0.016
3	13	260	13	10	13.000	260.005	12.991	9.997	1.21	0.003
4	82	240	16	45	82.016	240.004	16.047	44.957	6.06	0.043
5	40	385	15	5	40.015	384.997	14.976	4.981	3.50	0.019
6	48	68	10	180	48.006	68.027	10.009	180.017	3.51	0.017
7	60	60	7	5	59.993	60.015	7.032	5.032	4.31	0.032
8	82	100	11	90	82.015	100.000	11.000	89.985	1.88	0.015
9	60	380	6	0	60.001	380.005	5.972	0.000	3.41	0.000
10	110	130	9	270	110.009	130.007	9.018	269.937	2.64	0.063
11	70	23	9	0	70.001	23.045	9.066	360.000	9.63	0.000
12	110	260	3	250	109.990	260.021	2.915	250.016	10.63	0.016

complements our experiments. They compare convergence rates, convergence times, and individuals needed for convergence, and their conclusion is that DE converges faster, more robustly, and with fewer individuals.

The percentage of success running the program under these conditions is higher (93%). This is a logic result, because we have more information: several laser readings and movement (odometry) information. It means that the algorithm returns the correct location in the vast majority of the cases, a very promising result considering that the places are random and there are many symmetries in the map.

5.1.2 Accuracy of the method

The localization process results in different points of the environment can be seen in Table 5.1. The cartesian coordinate of the point we want to localize in the map is x, and $\hat{x}$ is the estimation obtained. This can be extended to y, $\hat{y}$, z, and $\hat{z}$. The orientation in the horizontal plane (tilt) is θ and $\hat{\theta}$ is the estimation obtained. The sensor noise level is 3% (a gaussian distribution over the laser distance with a standard deviation of 3%). The position error (e_p) is the distance in millimeters between the estimated value and the real position of the robot. The orientation error (e_θ) is the difference between θ and $\hat{\theta}$, in degrees.

We can conclude that the accuracy of the localization process is improved significantly compared with our previous works in 2D maps, and we believe it is the most important advance we have obtained. If we observe the position error column (e_p), the average error is equal to 5.1 mm, and the same measurement with our previous algorithm using 2D sensor data was equal to 2 cm, as can be checked in our published work [45]. The main reason for this reduction is the increase in the amount of information. The 3D laser reading contains nine 2D scans (slices). Thus, the information is multiplied by nine.

It is possible to compare the accuracy results with those obtained by other groups working on localization in 3D environments. The accuracy achieved by Kümmerle *et al.* [51] is about a few centimeters. Se *et al.*'s [58] average Euclidean translation error is equal to 7 cm and the average rotation error is 1°. However, they mention that it can be reduced by using a higher image resolution. Ho *et al.* [56] localize the robot within 1 m of true position and 5° for orientation. Their best results are 9 cm and 0.49°. The localization error mean obtained by Royer *et al.* [57] is equal to 1.9 cm and 0.1°. The results obtained in this paper are more accurate than those approaches.

Furthermore, this improvement has an important consequence for the computational cost. We work with a 3D laser scanner and it has a negative influence on the computational cost. However, the accuracy is higher and we need less iterations to obtain satisfactory results. The negative influence of the 3D laser data can be partly compensated with the positive influence of the accuracy improvement (allowing us to use the algorithm faster with relaxation in the accuracy or convergence requirements).

If we observe the orientation error, we see that its value is very small. The orientation error average was equal to 0.02° for the results shown in Table 5.1. Therefore, it can be concluded that the method presents good features in this aspect.

The influence of the sensor noise level can be observed in Figure 5.4, where the localization error (in centimeters) is represented against the laser noise level (standard deviation of a Gaussian distribution over the laser distance, in percentage of the weighted distance). We obtain acceptable results even with a high noise. As can be observed in the figure, the error is lower than a centimeter with a 15% of sensor noise. It is a great result that leads us to conclude that the algorithm presents good performance in this aspect. Besides, the typical error of this type of devices is around 1%.

5.1.3 Thresholding and discarding effect

A threshold rejection band has been added to avoid the premature elimination of solutions. This mechanism decreases the eagerness of the algorithm, letting it to eliminate a candidate solution from the set only when the offspring candidate is significantly better from a statistical point of view. A discarding operator was also added to accelerate the convergence of the algorithm by eliminating a very low percentage of the worst population individuals at each iteration of the algorithm, and resampling the individual candidate in the vicinity of a better candidate selected randomly between

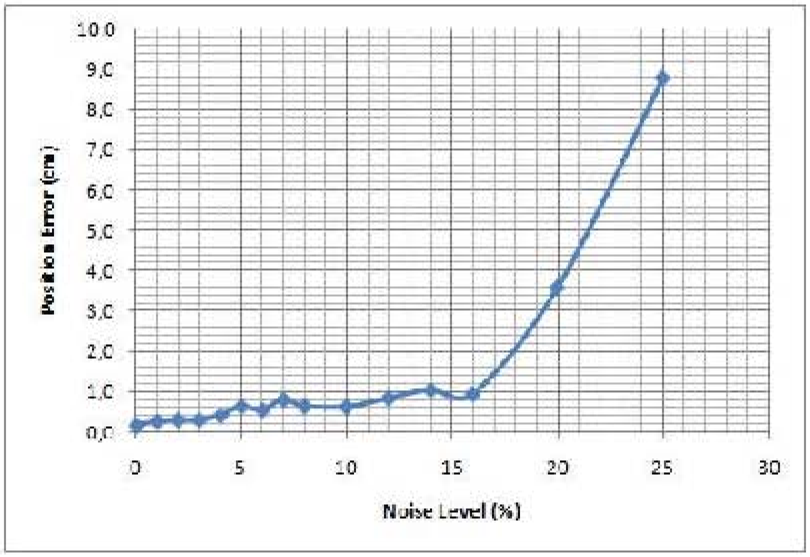

Figure 5.4: Error vs. sensor noise level. The localization error is in cm. The noise level is the standard deviation of a Gaussian distribution over the laser distance, in percentage of the weighted distance.

the best elements of the population.

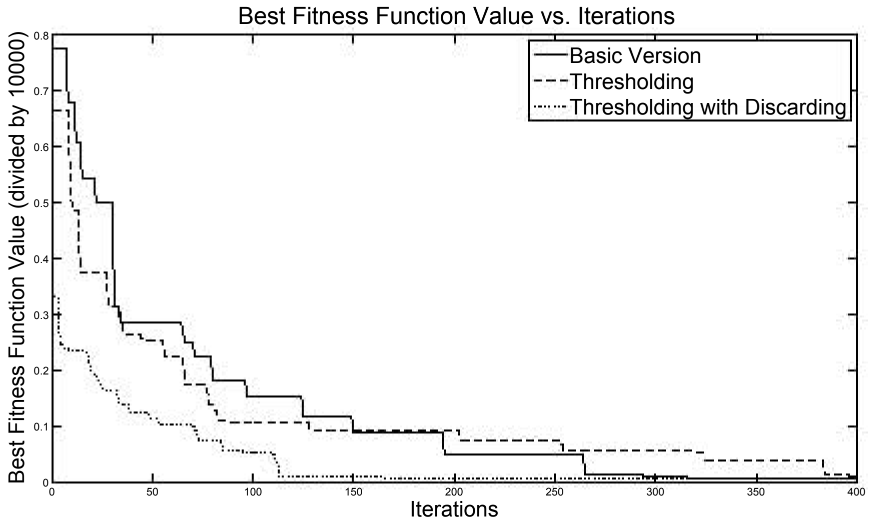

Figure 5.5: Best fitness evolution value vs. iterations. Three different options: without thresholding, with thresholding, and with a thresholding band and 5% of population discarding.

An important effect of the use of a threshold rejection level is the decreasing of the convergence speed, as shown in Figure 5.5. The latest improvement occurs in iteration number 380 in the thresholding option for the best estimation, while it is produced around iteration number 260 in the other case without thresholding. The fitness function is the L2-norm (Equation 3.30) and the observation measurements are in cells.

The use of a discarding mechanism accelerates the convergence of the algorithm with a threshold level. Studying the case with thresholding and discarding, it can be concluded that the convergence is reached earlier.

The algorithm with discarding is even faster than the basic version, while maintaining the capability to avoid premature convergence inside of the noise band, as can be seen in Figure 5.6, which shows the solution of the algorithm in a highly symmetrical environment.

In this type of environment it is not possible to distinguish between all rooms, because the observations are the same for all of them. If the algorithm is optimized in the noise band, it will eventually converge into a room, giving us a wrong result. It will finally converge to one solution, eliminating all the other rooms and concentrating all the population around the optimal solution.

It can be observed that the estimated solution (point in the center, with coordinates $(30.00, 385.00, 5.97, 0.00)$) does not coincide with the real solution $(30, 220, 6, 0)$, but there are still candidates in all rooms. It is a logical result that indicates that the algorithm avoids the premature elimination of solutions.

It is also important to say that the thresholding mechanism does not always make the algorithm converge slower. There are cases where the noise makes the algorithm evolve towards erroneous solutions. Although the number of candidates that evolve is smaller, the improvements are more significant, and the big influence of the best candidates on the population improves the convergence speed. It occurs in situations where noise plays a very important role, as can be seen in Figure 5.7, which shows the evolution of the best fitness function value in a particular case under the influence of an important noise (we use a range laser finder as sensor, and the noise introduced is equal to 10% over the distance weighted). It can be observed that the thresholding increases the convergence speed in this case, being even close to thresholding with discarding option.

5.1.4 Fitness function options

There are several options when choosing an appropriate cost function for our particular problem. The L1-norm and the L2-norm have been studied in this document, and the results are presented below.

In order to test the algorithm characteristics in this aspect, a simulated environment has been considered (Figure 5.1) . This environment is similar to many office indoor areas where all offices are located along the central corridor. This test environment will be used to verify the accuracy and robustness properties of the evolutive localization filter algorithm under two different cost functions, and the effect on the algorithm convergence of a contamination in the noise error. A Gaussian observation error of zero mean and standard deviation σ equal to 3% of the measured distance has been considered.

The first test tries to determine the capability of the algorithm to localize the robot when it is located in one of the distinguishable rooms. The robot's pose is

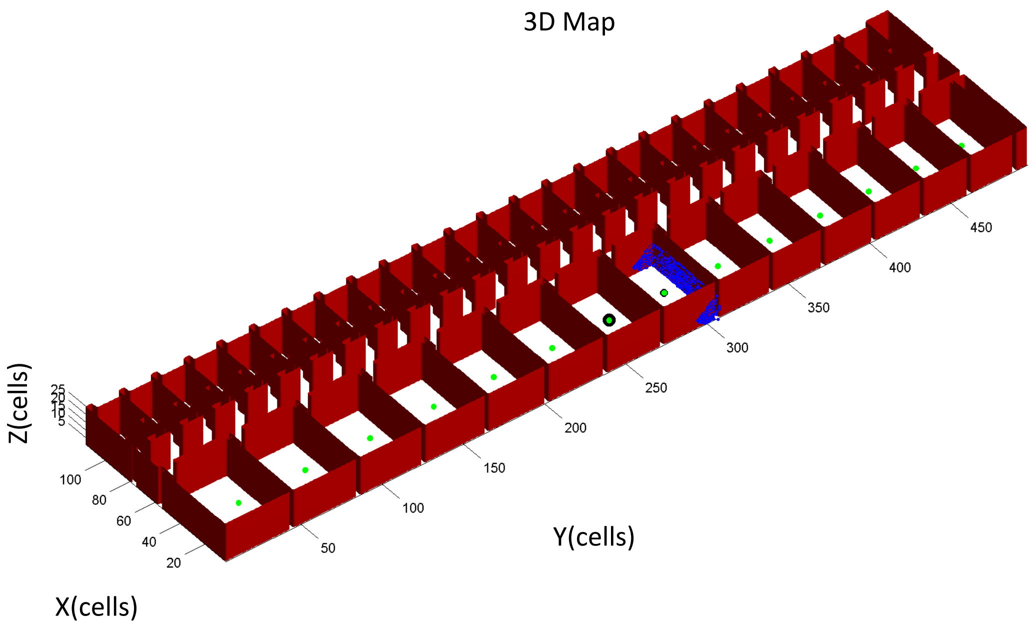

Figure 5.6: Algorithm solution in a symmetrical environment, with thresholding and discarding

.

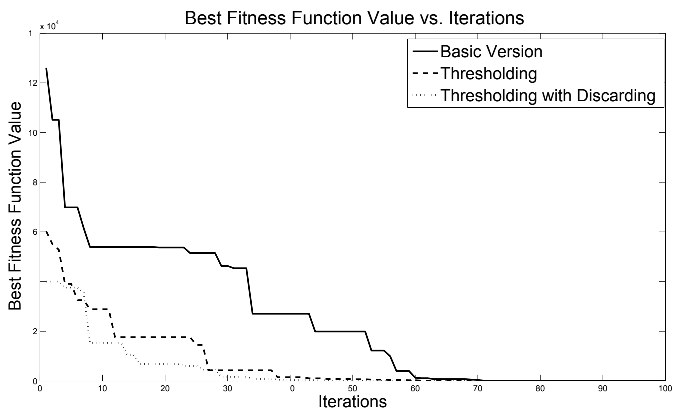

Figure 5.7: Best fitness function value vs. iterations. Three different options: without thresholding, with thresholding, and with a thresholding band and 5% of population discarding. Noisy case (10% over distance).

variable and there is no contaminated noise. The position errors in centimeters are presented in Table 5.2 (columns with subindex NC).

The second test tries to determine the capability of the algorithm to localize the robot when there exists a contaminated Gaussian noise. This behavior has already been studied in our previous work [112]. This situation happens when there are mobile objets or unexpected obstacles. In order to test this situation the robot has been located in one of the distinguishable rooms and the normal Gaussian noise has been contaminated with a uniform distribution between the 25% and 75% of the sensor measurement. The results with a 10% of contamination can be observed in Table 5.2 (columns with subindex C).

Comparing the results presented in the no contaminated case, there is no clear evidence to choose one option, as in both cases the errors present comparable values. Only the position error is considered, because the results were negligible with the orientation error.

However, if we study the case with uniform contamination, the results are much more interesting. The error estimating the pose using the L1-norm (the average error is equal to 1.087 cm) is significantly lower than using the L2-norm (the average error is 8.464 cm). This is a logical result, because the contamination introduced has a bigger influence in the penalization of the cost function value using the square error, being much more robust with the first order one.

Moreover, if we look at the terms in parentheses, which correspond to the population size used in each case, we find another interesting conclusion. It is very often necessary to use a larger population to succeed (with a smaller size the robot was unable to locate) in cases of contamination when using the L2 norm, which adversely

Table 5.2: L1-norm and L2-norm estimation position errors from different locations. Two different situations: no contamination (NC), and 10% of contamination (C). Errors in centimeters and locations (x,y,z,θ) in cells. Each cell size is a cube of 12-cm side. Population size required is in parentheses.

True Pose	$e_{P(L1,NC)}$	$e_{P(L2,NC)}$	$e_{P(L1,C)}$	$e_{P(L2,C)}$
(95,430,12,180)	0.783(200)	1.123(150)	4.040(250)	18.84(200)
(25,190,6,0)	0.113(250)	0.303(250)	0.570(250)	1.222(250)
(50,100,10,0)	0.672(150)	0.665(150)	0.656(150)	0.576(300)
(50,350,13,0)	0.943(200)	1.907(200)	1.646(200)	3.324(200)
(90,425,8,270)	0.652(150)	0.628(150)	0.689(150)	12.121(150)
(80,240,15,270)	0.623(150)	1.037(150)	0.907(150)	4.158(150)
(70,350,8,270)	0.661(200)	0.221(150)	1.605(150)	1.656(500)
(30,225,14,90)	0.333(150)	0.278(150)	0.151(150)	13.059(150)
(30,200,5,90)	0.673(150)	0.811(150)	0.366(150)	13.345(300)
(25,275,10,90)	0.289(150)	0.461(150)	0.246(150)	16.342(300)
Average	0.574	0.743	1.087	8.464

affects the computational cost.

Therefore, we obtain better solutions with the cost function given by the L1-norm in cases where there is a significant contamination (cases with many dynamic objects, pedestrians, etc.).

Besides, it can be appreciated the algorithm present good performance with Gaussian errors up to ten percent in measurement in both cases (with and without contamination).

5.1.5 Complexity and computational cost

An additional issue addressed in our experiments concerns the running time of the RELF-3D algorithm. The computational cost of any algorithm that works in 3D maps is very high, and this is one of the biggest problems found by researchers in this field. The absolute time depends on several factors: the computer platform, the observation prediction model and the sensor data, and the population and iterations number.

The algorithm complexity is $O(n)$, which means that it grows linearly with the population size and the perceptive sensor size. It does not depend on the robot's DOF, which can be easily expanded if necessary.

Table 5.3 shows the time required for RELF-3D to update the estimate value using 9×61-range laser data, 15 iterations/step, and different population sizes. Table 5.4 shows the time required for the RELF-3D algorithm to update the robot's pose

Table 5.3: Computation time for a fixed number of iterations.

Population/iterations	Time(s)
200/15	16.14
100/15	8.77
50/15	4.09
25/15	2.25
10/15	1.04

Table 5.4: Computation time for a fixed population of 50 elements.

Population/iterations	Time(s)
50/5	1.42
50/10	2.95
50/15	4.03
50/20	5.66
50/25	7.21
50/50	12.75

estimate using a fixed population size and changing the iterations per step number. In both cases, the behavior of the algorithm is completely linear. In our experimental tests, the best results in a 865 $m^2 \times 3$ m environment have been obtained with an initial population of 50 elements and 15 iterations.

Analyzing the localization method described in Algorithm 3, the bottleneck from a computational point of view is the function named as *dist_est_3d*. It consists of estimating the 3D laser scan (composed of 9×61 laser beams) from a pose estimate in the simulated environment, and it has to be calculated for each population member. The reading is used by the fitness function to calculate the error.

A critical point in any global localization algorithm is the variation in the computational requirements with environment dimensions. The RELF-3D algorithm requirements in our experiments not only depend on the size of the environment but also on the characteristics of the place where the robot is located. As an example, in the experiments done in our laboratory test site, a 900 $m^2 \times 3$ m environment, the algorithm requires at least 50 samples when it is in the corridor, and 100 when it is in one of the offices.

The computational cost is higher when we use the algorithm in three dimensions, but the improvement of the accuracy is an important result. If we work with a mobile manipulator and tasks with small objects, it will be indispensable to obtain an accuracy of millimeters. If the robot does not need this level of accuracy, we will obtain a reduction in computational cost with less restrictive convergence conditions.

It is not easy to compare our algorithm to other methods. Se *et al.*'s [58] global localization computational times are 0.725 and 0.02 s for two different methods implemented. They show that the SIFT feature extraction is efficient from a computational point of view. However, their method is vision-based and the computational problems that they need to solve are different. Besides, they do not focus on obtaining the maximum accuracy and their map size is 100 m^2 (our map size is 900 m^2). Ho *et al.*'s [56] algorithm needs 4.82 s to obtain the global localization solution. This time is similar to the time obtained in our results. Lingemann *et al.*'s [52] method can be used online because they need 10.38 s to compute a trajectory composed of 2844 scans. However, it is not easy to compare their method to our algorithm because their times are referred to the tracking problem. Finally, the time needed to obtain the measurements can be the most important one in many cases and it has to be considered in real applications. The time delays of the ultrasonic transmitter/receiver modules are approximately equal to 0.5 s in the paper published by Tsai *et al.* [53].

5.1.6 Initial population estimation

It has to be remarked that there are two different options when choosing the initial population size. It has been empirically adjusted in the previous experiments, but we have also developed an initialization mechanism in Section 3.2.6 that is based on the information perceived by sensors. The method performance has been demonstrated here with three different experiments.

The first one demonstrates the capability of the method to solve the global localization problem when the robot is situated in a large area. The second experiment is similar, but the robot is now in a medium-size map. Finally, the estimate is compared with the best solution in different cases, representing the probability of success against the initial population size. All experiments have been conducted in a planar map for simplicity.

Large map

A simulated indoor map has been chosen to test the characteristics of the method. It is a large indoor environment and there are significant differences in the number of elements of the initial population when the robot is situated in different places. The computational cost can vary from a few seconds to more than twenty minutes. The map size is $960 \times 600 = 576000$ cells, which is equivalent to an area of 8433 m^2 (the cell is a 12.1 cm-side square). This map is shown in Figure 5.8.

The results of the global localization process for the first observation when the robot is situated in ten different places are shown in Table 5.5. The left columns represent the cartesian coordinates and the orientation in cells and degrees, respectively. N_p is the number of elements obtained with the initialization method. A characteristic of the place is described in the next column, and the elapsed time needed in each

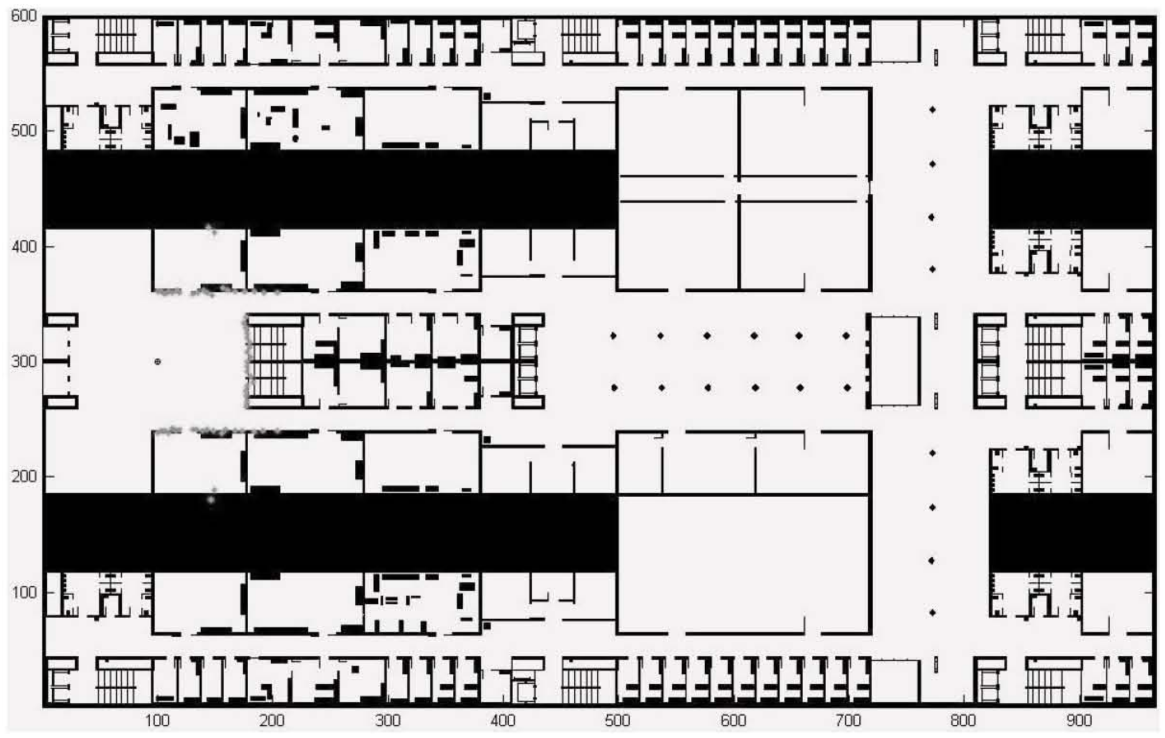

Figure 5.8: Map of the environment (large area). All units in cells.

case is shown in the right one.

It is a map with important symmetries, which is an important drawback that increases the number of elements. For instance, if the robot is located at $(50, 60, 0)$, there are two different locations with exactly the same observation vector and it is impossible to distinguish between them without additional movement of the robot and more perceptive information. The other possible location is approximately $(850, 60, 0)$. The true pose will converge to one of them depending on the stochastic evolution of the localization method.

In case 7 of the table, there are eight possible locations with the same observation vector. It is a corridor with columns in the middle, and there are four identical places where the robot is looking in one direction and other four where it is looking in the other one. It is not possible to distinguish between them because the range of the simulated laser scanner is not large enough. In order to solve the problem, the robot must obtain another observation vector after movement.

It can also be observed that there are important differences in the computational cost, which leads us to conclude that the correct estimation of the initial population

Table 5.5: Number of elements of the initial population in different places of a large environment. The cartesian coordinates are in cells, the orientation in degrees, and the time in seconds. Sensor error: 2% over the distance weighted.

	x	y	θ	N_p	Characteristics	Elapsed time
1	600	80	90	200	Big room	24.51
2	600	130	0	272	Big room	30.22
3	50	60	0	1080	Narrow corridor	162.58
4	100	300	0	1500	Wide corridor	290.86
5	650	400	0	1518	Medium-size room	335.81
6	400	50	90	2004	Narrow corridor	375.85
7	750	100	90	1710	Big corridor with columns	377.89
8	300	550	0	5496	Very narrow corridor	831.43
9	330	580	270	20700	Small office	$\rightarrow \infty$
10	130	580	270	79038	Small office	$\rightarrow \infty$

number of elements is a crucial factor in this aspect. It should be added that it is the computational cost for the global localization problem in a large environment with only one observation vector and no movement. Once this problem has been solved and all candidates have converged to the true solution, this number can be drastically reduced. The number of iterations to converge is also reduced because the algorithm now starts with the previous results as initial population instead of a random distribution over the map. These factors have a great influence on the computational cost and the algorithm could be used online. If there is no reduction in the population size, it usually takes a few seconds, and less than one second when the candidates are re-sampled in the true solution area. However, it is not the subject of this work and it has been explained in our previous work [45].

The typical situation in mobile robotics includes robot movement. This situation has not been studied here because it is not the purpose of this experiments, which focus only on the initialization problem. This fact has also been studied before [80].

There are some cases where the estimated number of elements is too high and it is impossible to work with a population of this size. For example, when the robot is at $(130, 580, 270)$, the initial population size is equal to 79038 elements. It is impossible to work with this number from a computational point of view. It is logical because the robot will be able to locate itself when there is enough information to do it. If it is in a corner looking at the wall, it will not be no possible to solve the global localization (like human beings, because we cannot know where we are in these cases). The robot must turn around until the perceived information is maximized, and the initial population number will decrease to an acceptable value.

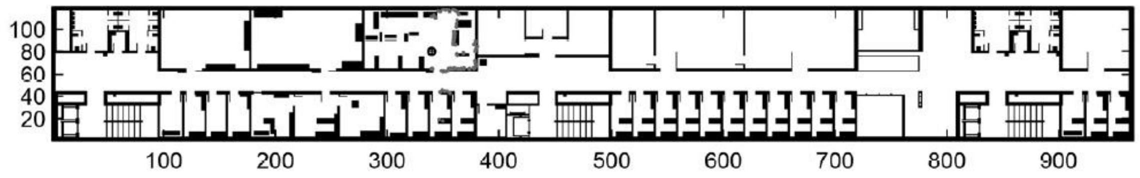

Figure 5.9: Map of the environment (medium-size map). All units in cells.

Table 5.6: Number of elements of the initial population in different places of a medium-size environment. The cartesian coordinates are in cells, the orientation in degrees, and the time in seconds. Sensor error: 2% over the distance weighted.

	x	y	θ	N_p	Characteristics	Elapsed time
1	800	45	90	78	Wide corridor	11.57
2	800	50	0	132	Wide corridor	15.11
3	450	50	90	116	Wide corridor	19.53
4	200	70	0	250	Big office	30.60
5	650	110	270	336	Big office	37.20
6	830	60	180	390	Medium-size corridor	68.67
7	400	60	0	582	Medium-size corridor	64.95
8	350	100	90	1156	Small office	110.62
9	230	30	270	2254	Small office	172.40
10	190	30	0	4338	Small office	791.64

Medium-size map

The second environment is a medium-size area shown in Figure 5.9. The map dimensions are 960×120 cells, and the total area is equal to 1687 m^2. The results are shown in Table 5.6.

The time requirements in this experiment are substantially lower. The map is smaller and the result is obtained in 11 s in the most favorable case. The influence of the map size on the population estimate can be deduced from Equation 3.41.

The computational cost does not only depend on the population size. There are places where the algorithm converges faster with a larger population (cases two and three of the table). It is because it is easier to achieve the global minimum when the basin of attraction is bigger and more abrupt. The symmetries have a negative influence in this aspect because it is more difficult to converge to the true pose when there are similar locations.

Analyzing both experiments, it can be concluded that the estimation of the initial population size is robust enough to achieve the solutions when the robot is located in different places of medium-size and large environments.

Comparison between estimate and best solution

In the first experiments, the initialization method has been tested and it has been concluded that the proposed method is good enough to localize the robot in a robust way. After that, it has been thought that it is necessary to have statistical evidences to validate the hypotheses. In other words, it is necessary to know how good the results are.

Besides, the computational cost is one of the most important challenges and shortcomings of particle-based localization methods, and the objective is to obtain good performance in the localization process with a minimum number of elements. This requirement is studied in the following experiment, which is very simple and intuitive.

The number of elements of the initial population is fixed (without the initialization method) and the algorithm is executed obtaining the percentage of success. It means that the probability of being correctly localized with a constant initial population size is calculated. After that, the number of elements is changed and the previous process is repeated. The goal is to represent the percentage of success against the initial population size. Finally, the results are compared with those obtained with the initialization method.

The simulated indoor environment of Figure 5.9 has been chosen in this experiment and the results obtained are shown in Figure 5.10.

The population size must be large enough to have the highest probability of localization. Nevertheless, the computational cost increases with this number. Therefore, the best estimate is the minimum number of elements that maintains the highest probability.

Studying the graph, an interesting conclusion can be reached. There are some places where it is more difficult to localize the robot and it does not only depend on the initial population size (it was also discussed in the other experiments). It can be because there is not enough information (the robot is in a corner) or the robot is in a symmetrical place. If there are two different places with the same aspect, it will be impossible to distinguish between them to localize the robot with only one observation vector. That is the why the maximum probability is not the 100% in some cases. The important conclusion is that the proposed method estimates a number of elements that is in the maximum probability band. Besides, the discontinuities have also a negative influence, because it is more difficult to reach the basin of the global minimum.

Six different cases have been represented and the most difficult one is the example

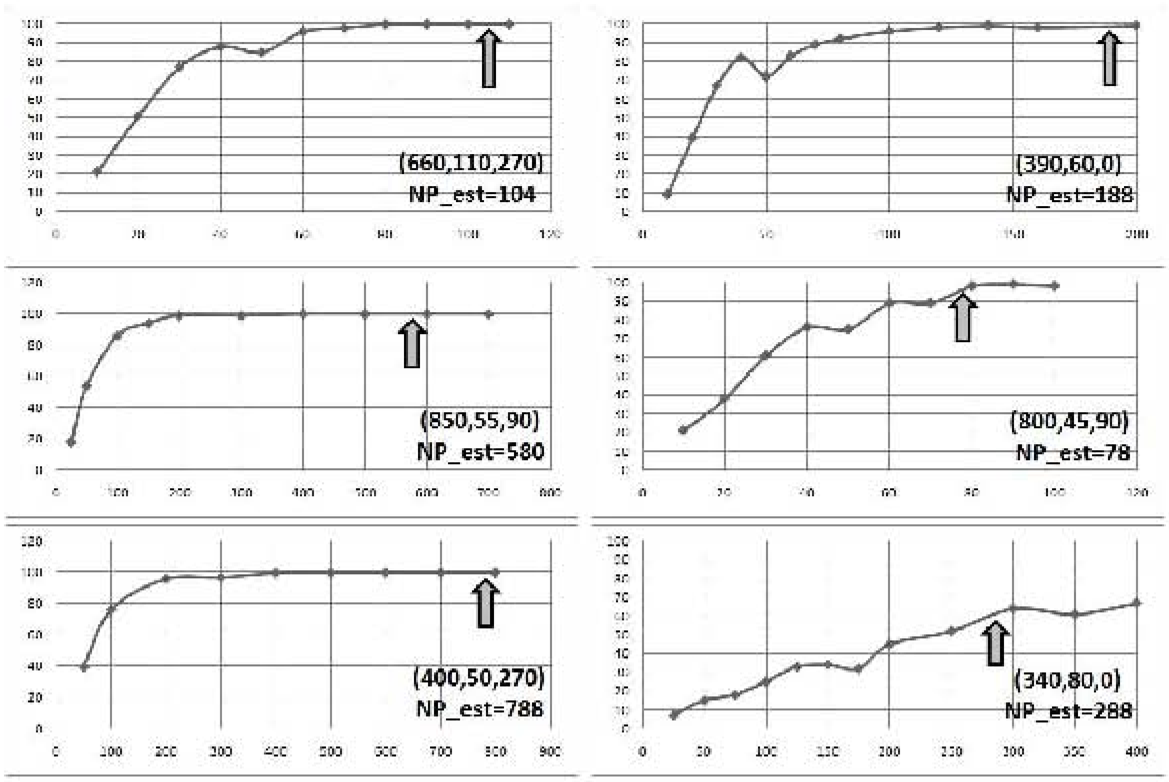

Figure 5.10: Probability of a correct global localization (%) vs. population size in different places of the environment.

where the robot is located at $(340, 80, 0)$. This case presents symmetries problems because it is situated in one office and there are similar ones. Besides, at least five discontinuities can be found in the laser reading. As was said previously, the percentage of success is not equal to 100%. The factor that is introduced to avoid the discontinuities effect limits this problem, but it is not sufficient in those cases with many complications. The robot is unable to always obtain the true position, but the estimate is in the maximum probability band. This case does not have a unique solution and it is impossible to solve it with a single observation vector. It has been introduced to show the reader what happens in these situations.

Some problems have been found when the population size is very large. In some cases, the estimate is too conservative because the calculations have been made to obtain a robust formula. The results are robust but the population size implies problems with the computational cost. It can be avoided by limiting the upper limit of the population depending on the computational requirements for a specific task. The number of candidates can also be penalized when the population is over a specified value.

An interesting conclusion has been reached after these experiments. When the number of elements is small or medium, the estimated number is optimum. If there is not much information and the estimated population is large (more than 500 elements), the formula is robust but too conservative and the size may be smaller. It has to be noted that the results are applicable to this environment. Observing Figure 5.10, good results could be obtained with fewer elements. An additional correction factor could be introduced in the formula when the number of elements is larger than a threshold. A good performance could be obtained with a factor equal to 0.6 when the population size is larger than 500.

5.2 Mapping in 3D Environments

5.2.1 Data sets

The $Hannover2$ data set has been recorded at the university campus of Leibniz Universität Hannover, Germany. It contains 924 3D scans (with 360° field of view), covering a trajectory of about 1.24 km. Each scan is approximately composed of 16600 points. The maximum range is $d_{max} = 30$ m. Thanks to Oliver Wulf for making available these data[7].

The $UC3M$ data set obtained by MANFRED-2 has been recorded at the university campus of the UC3M. It contains 150 3D scans covering a 900 m^2 indoor environment (Department of Engineering Systems and Automation of the UC3M). Each scan is approximately composed of 45000 points. The horizontal (tilt) resolution is 0.25°, and the amplitude of a 2D scan is 190°. The vertical (pan) resolution is 1° and its amplitude is 70°. The maximum range is $d_{max} = 15$ m.

5.2.2 Scan matching

Two different experiments have been conducted to verify our DE-based scan matching method. The algorithm has been configured with the following parameters: $F = 0.8$, $Cr = 0.75$, $S_N = 0.02$, $N_P = 20$, and maximum iterations equal to 50. An additional convergence criterion that follows the ideas developed for the RELF-3D algorithm has been implemented. The best element fitness value ($min(e^k)$) is checked and the process is stopped when its value is equal or lower than $3N_P/2$.

The $UC3M$ data set is used in the first experiment to study the error in local environments composed of several laser frames. The purpose is to observe the visual appearance of the map that is built after registration.

Some results in different situations are shown in Figure 5.11. The distances between consecutive scans are equal to 30 cm. The local map after joining four different scans when the robot is located in a corridor is represented in the upper part. The high accuracy can be observed in the horizontal projection. Two different labs are modeled in the middle and bottom figures. The middle one is composed of four scans while the bottom one is formed by six laser frames.

In general, the high accuracy of our method lets us conclude that the algorithm can be used in manipulation tasks. It is not easy to measure the scan matching error because the robot's truth pose is not available in indoor environments. However, the error is equal or lower than a centimeter when analyzing the horizontal projections of the local maps. Similar errors have been obtained after applying an ICP-based

[7]`http://kos.informatik.uni-osnabrueck.de/3Dscans/`.

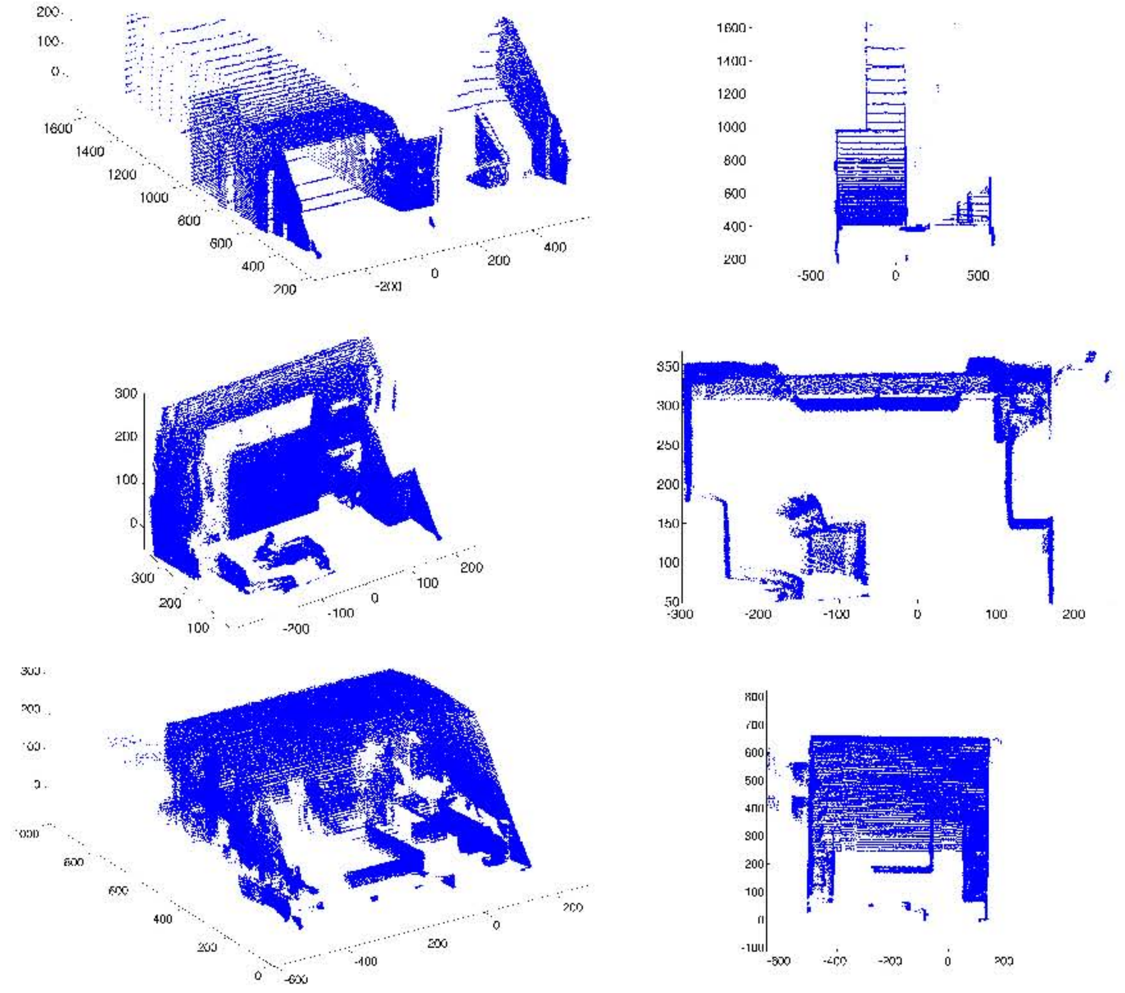

Figure 5.11: Scan Matching. Different local maps obtained after registration. $UC3M$ data set. Top figures: university corridor. Middle and bottom figures: different labs. Left: 3D view. Right: horizontal projection. All units in cm.

method in the corridor. Our method accuracy is better than the ICP-based scan matching accuracy when the robot is located in the labs.

A map of a big area composed of multiple measurements has been analyzed in a second experiment. The $Hannover2$ data set has been utilized in this case. Figure 5.12 shows a map built after registration, which is composed of 500 laser scans.

The robot's trajectory starts in a point of the rectangle that is located in the bottom part of the figure and ends in the upper part of the figure. It can be observed that the map has an acceptable appearance, but the accumulated error is specially important in the last part or the robot's path. Although accurate results have been

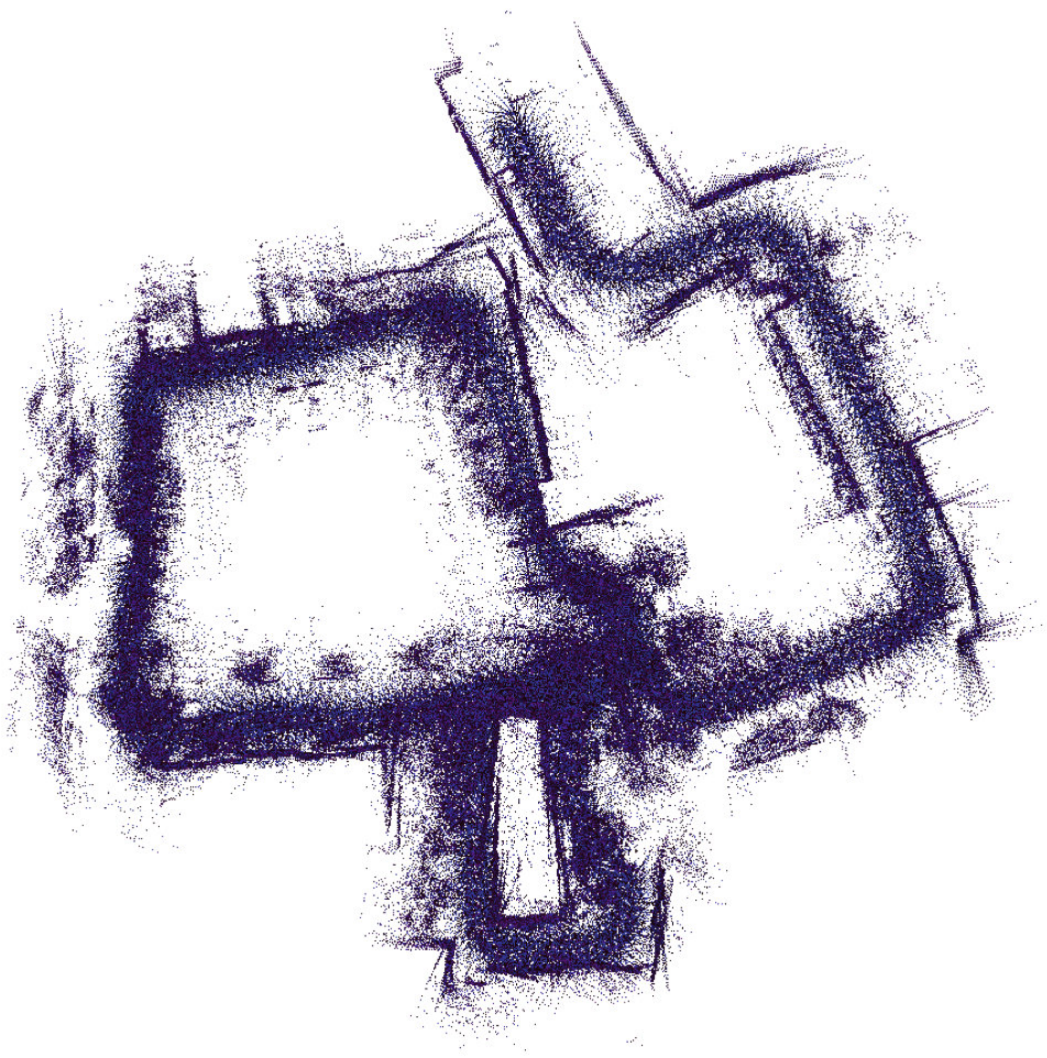

Figure 5.12: Scan Matching. Map of an area composed of 500 laser scans.

obtained after registration of scan pairs, the accumulated error makes the development of a loop detection and closure algorithm necessary.

The average time needed to register a newly acquired scan and incorporate it to the model is equal to 1.83 s when using the DE-based scan matching. This time is equal to 1.73 s for the ICP-based algorithm. The preprocessing step is included in those times. Both times are low enough to use the algorithms online, taking care of the minimum time between two consecutive scans.

The algorithm complexity is $O(iter \cdot N_P \cdot n \cdot \log n)$, which means that it grows linearly with the population size (N_P), the iterations ($iter$), and the sensor size after preprocessing (n). The term $\log n$ corresponds to the average of the NN search accelerated with k-d trees. It does not depend on the DOF, which can be easily expanded, if necessary.

5.2.3 Loop detection

The algorithm has been tested using the $Hannover2$ data set. The configuration parameters are: $K = 10$, $t_l = t_p = 0.1$, $N_s = 9$, and the minimum loop size is equal to 25 scans. Different values of N_n and configurations of the *numerical* features have been tested. Thresholds t_1 and t_2 have been also studied.

The loop detection matrix between all possibilities, given a robot's trajectory composed of 375 consecutive scans, is shown in Figure 5.13. The thresholds are set to $t_1 = 93\%$ and $t_2 = 90\%$. The positive matching is represented by white points, grey means uncertainty band, and black is negative matching. The LPI values between all pairs have been calculated. It is a symmetric matrix where the axis indicates the scan number (which are consecutively numbered starting by the first reading). The robot's path can be observed at the bottom of the figure.

Analyzing the robot's path, several loops must be detected. The first loop detected is located around the coordinates $(110, 0)$ of the matrix (or $(0, 110)$ because the matrix is symmetrical). There is not only one white point because the robot's path after the first loop detection coincides with the previous one. The loop size variable is reset after detection in a real applications, therefore no more loops around these coordinates will be detected after the first positive. It is the first time that the robot is navigating around a pre-visited place and it is correctly detected.

After that, the largest close trajectory is described by the robot and the starting point is visited again. This second visit that is also classified as positive matching has approximate coordinates equal to $(340, 0)$. Finally, a third loop is detected around coordinates $(110, 340)$. It is a logical detection because in both cases the robot is visiting the starting point. No more loops are appreciated in the matrix, which is a correct result because there are no more coincident locations in the robot's path.

Since all loops are correctly detected and no false detection is appreciated, it can be concluded that the loop detection algorithm presents an optimal performance.

An additional question that needs to be answered is how to fix the thresholds t_1 and t_2. A preliminary heuristic study has been developed in this paper. Figure 5.14 shows the probability histogram between 150 different scans of the $Hannover2$ data set. All possible combinations have been computed. The LPI descriptor is composed of three *numerical* features (volume, distance, and curvature) and nine

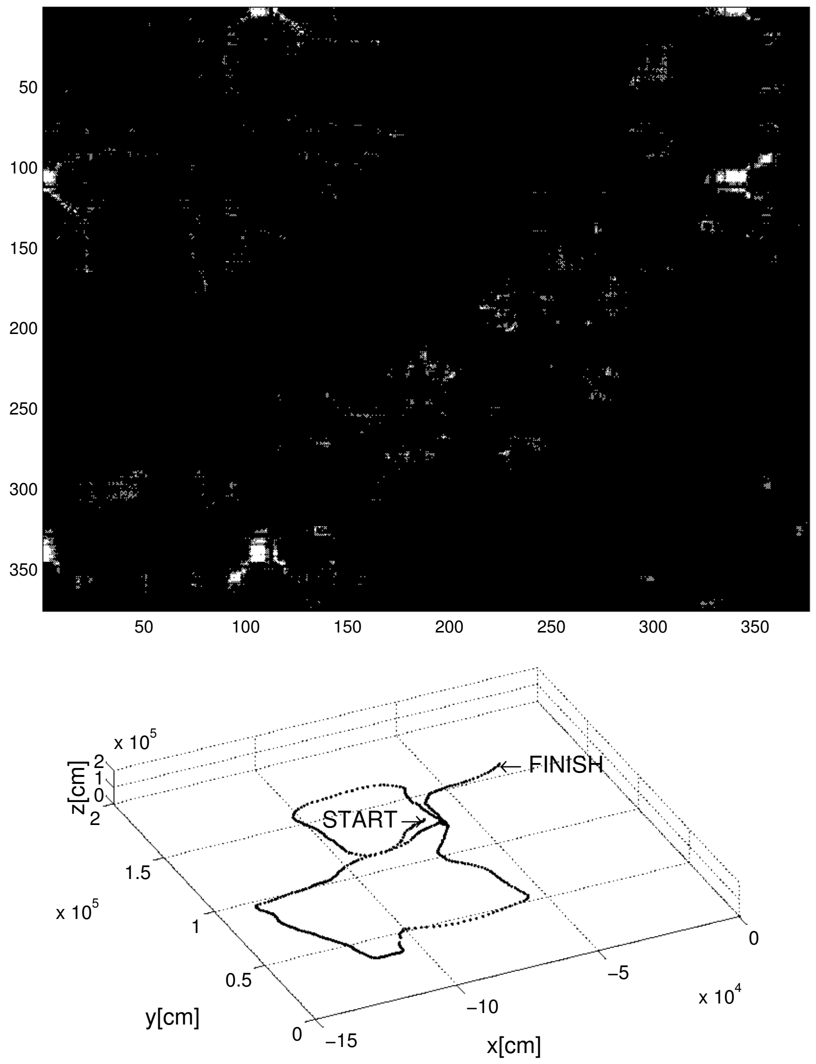

Figure 5.13: Loop detection matrix between 375 consecutive scans ($t_1 = 93\%$, $t_2 = 90\%$): white represents positive matching, grey means uncertainty band, and black is negative matching. The robot path is at the bottom.

surface features.

The histogram is composed of two Gaussian-shaped distributions. If the loops (positive matching) are contained in the right one, the thresholds can be chosen depending on the right distribution properties. The distribution average is equal to 95.04% and the standard deviation is 1.66%. A conservative choice for t_1 could be the lower limit of the probability distribution, which is around the 90%. But this will be discussed in the next experiment.

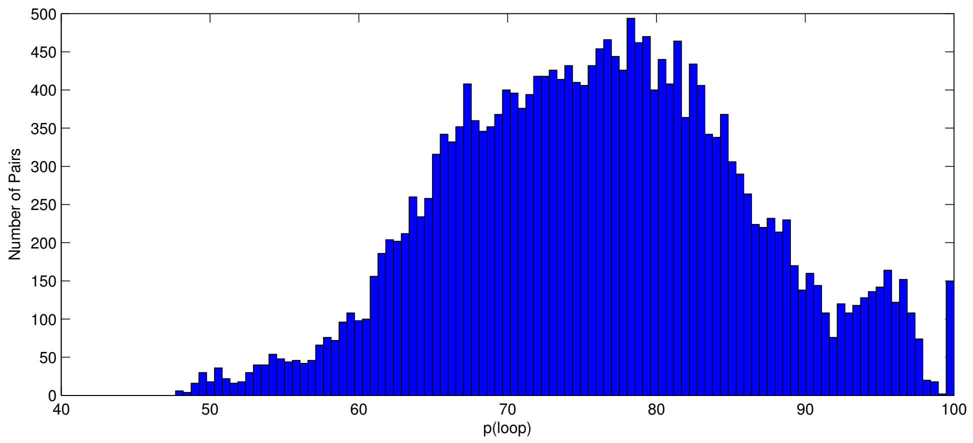

Figure 5.14: Probability histogram between 150 different scans of the $Hannover2$ data set.

A different experiment has been designed to quantify the loop detection performance. The ground truth distance of the $Hannover2$ data set is available, and a distance threshold (d_{gt}) has been defined in order to calculate two probabilities:

- Probability of Detection (PD): number of pairs classified as positive matching and a distance between them lower than d_{gt} divided by the total number of pairs located closer than d_{gt}.

- Probability of False Alarm (PFA): number of pairs classified as positive matching and a distance between them greater than d_{gt} divided by total number of pairs located closer than d_{gt}.

If the ground truth distance between a pair of scans is lower than the threshold, the loop detection algorithm must classify this pair as positive. The PD must be the highest possible, but high values for the PFA are critic, understanding the PFA as the probability of detecting a loop when it does not exist. In addition, one single false detection makes the whole mapping process collapse. Therefore, the threshold t_1 will be the highest PD value that keeps the PFA at the minimum.

It has to be remarked that d_{gt} has been used as a tricky tool to check the algorithm capabilities. It is not possible to check it in indoor environments because this distance is not available. The threshold d_{gt} has been fixed to 6 m.

The influence of the most important variables on PD and PFA has been analyzed. The first aspect to be studied are the weights of the LPI function. Since three different

Table 5.7: Highest probability of loop detection with the minimum probability of false alarm depending on N_n and the uncertainty band. Both values in percentage. PD(PFA). $w_i = \max(f_i, g_i)$

N_n	$t_1 = t_2$	$t_1 = t_2 + 3$	$t_1 = t_2 + 10$
0	23.53 (0.66)	27.13 (1.19)	22.93 (0.86)
3	26.53 (0.86)	29.73 (1.26)	32.80 (1.20)
12	28.00 (0.60)	33.20 (1.07)	**35.53** (0.80)

Table 5.8: Highest probability of loop detection with the minimum probability of false alarm depending on N_n and the uncertainty band. Both values in percentage. PD(PFA) $w_i = mean(f_i, g_i)$

N_n	$t_1 = t_2$	$t_1 = t_2 + 3$	$t_1 = t_2 + 10$
0	26.27 (1.19)	29.67 (1.20)	32.4 (1.13)
3	31.80 (0.66)	32.60 (1.13)	**38.67** (1.00)
12	28.20 (1.20)	30.67 (1.00)	33.53 (1.06)

weights have been considered (Equations 4.21 and 4.22), each one will be represented in a different table.

The second factor is the number of *numerical* features (N_n). The feature descriptor is composed of 9 *surface* features and different possibilities for N_n have been tested: $N_n = 0$, $N_n = 3$ (volume, distance, and curvature mean), and $N_n = 12$ (volume, distance, curvature mean, curvature standard deviation, range standard deviation, range mean, range kurtosis, centroid, distance to centroid mean, distance to centroid standard deviation, range differences mean, and range differences standard deviation).

The third factor is the uncertainty band. If the LPI value between a pair of scans is inside the uncertainty band, a third scan is considered. The loop is now detected depending on the new LPI value using the new acquired scan ($LPI \geq t_1$). It means that when the robot is checking if it is navigating around a previously visited place and the LPI value is inside the uncertainty band, it waits for a new laser reading and uses it to improve its loop detection capabilities. Three different options have been considered: no uncertainty band ($t_1 = t_2$), narrow band ($t_1 = t_2 + 3$), and wide band ($t_1 = t_2 + 10$).

The highest PD values that keep the PFA at the minimum depending on the previously explained factors have been represented in Tables 5.7, 5.8, and 5.9. The experiment consists of decreasing t_1 (and t_2, because this value is a function of t_1) while checking the PFA. The value of t_1 that is chosen as critic is the minimum value that keeps the PFA at the minimum. The PD is calculated for this critic value of t_1. Only these critic values are represented for simplicity.

The improvement introduced by the uncertainty band can be observed in the

Table 5.9: Highest probability of loop detection with the minimum probability of false alarm depending on N_n and the uncertainty band. Both values in percentage. PD(PFA) $w_i = \sum_{j=1}^{T} \max(f_i, g_i)/T$

N_n	$t_1 = t_2$	$t_1 = t_2 + 3$	$t_1 = t_2 + 10$
0	22.93 (1.07)	25.80 (0.93)	25.20 (0.66)
3	26.53 (0.60)	29.93 (0.93)	30.20 (0.80)
12	28.47 (0.86)	32.80 (1.00)	**34.73** (0.60)

tables. The results with a wide band are slightly better than the results when a narrow band is chosen. The algorithm performance is also improved by the introduction of the *numerical* features. However, it is not possible to conclude if a feature descriptor with $N_n = 12$ is better than a feature descriptor with $N_n = 3$. Finally, there are not important differences when comparing the results using different weights.

The best results are obtained using the average between features as weight, three *numerical* features, and a wide uncertainty band. In this case, the probability of detection is equal to $38,67\%$. This probability is high enough to be successfully applied to loop detection. The results are similar or even better than those obtained by other authors. Magnusson *et al.* [104] show a $PD = 35.3\%$ with the same data set; Granstrom *et al.* [105] obtain a $PD = 63\%$ with no false positives, but their algorithm is included in a machine learning scope. It is a complete different focus and both algorithms cannot be compared because their method needs a large amount of initial data to build the classifier. The loop detection problem cannot be included in this scope because an autonomous robot has to be able to build a robust model even with no initial information.

Besides, the distance threshold (d_{gt}) is equal to 6 m, which is not a very restrictive assumption. The information received from two different locations with distance between them equal to 6 m can be very different. If this value is reduced to 3 m, the PD is increased to 65.2% while keeping the PFA at the minimum. This is a very promising result.

Finally, the PD and the PFA are represented as a function of t_1 for the best option of the previous experiment in Figure 5.15. There are two graphics that correspond to $d_{gt} = 6$ m and $d_{gt} = 3$ m. We have built this graphic in order to calculate the value that is shown in Table 5.8. It can be observed in the left part of the figure that the highest PD that keeps the PFA at the minimum is the same that is shown in Table 5.8 (38.67%). The optimal value of t_1 is around a 90%, which is similar to the heuristic value deduced before in this section. The increase of the PFA starts when t_1 is below this value.

The computational cost of the implemented method is an important aspect because it is necessary to handle a large amount of information provided by the 3D

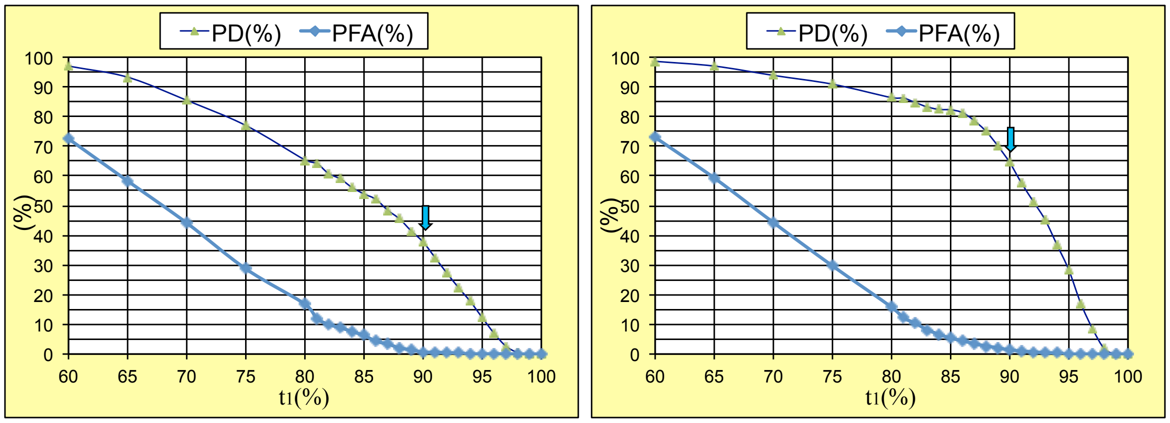

Figure 5.15: PD and PFA as a function of t_1. Settings: $w_i = mean(f_i, g_i)$, $N_n = 3$, $t_1 = t_2 + 10$. Left: $d_{gt} = 6$ m. Right: $d_{gt} = 3$ m.

sensors. The loop detection average time is equal to 0.36 s including data reduction, feature extraction, and probability calculation. The time complexity is $O(n)$, where n is the laser scan size.

Chapter 6

Conclusions and Future Developments

Three different algorithms related to the mapping problem in 3D environments are the main contributions of this work. These algorithms solve the following problems: global localization, scan matching, and loop detection. The most important conclusions of each method are detailed in this chapter.

6.1 Global Localization

The last version of our global localization method based on evolutionary concepts has been presented in this document. It has been designed to work in 3D environments with six DOF at maximum. It is based on DE, which is a particle-based evolutionary algorithm that evolves in time to the solution that yields the cost function lowest value. The ability of the algorithm to successfully perform its task is demonstrated, and the influence of various improvements, such as thresholding and discarding, is also discussed.

As we have demonstrated by the results obtained in the tests carried out, this new version is able to improve the accuracy in an order of magnitude in comparison with the results obtained with the original method in planar maps. The increase in the amount of information allows us to obtain an accuracy of millimeters, which is below the accuracy of the laser manipulation tasks with a mobile manipulator. This feature is one of the most important conclusions of the presented work.

It is necessary to emphasize that the increment in the computational cost due to the increase in the amount of handled information can be limited, relaxing the accuracy and convergence requirements. The computational cost is an important limitation to be addressed because it is a common problem in this type of methods. It can be seen in Chapter 5 where the computational cost is compared to those obtained by other methods. Besides, the population reduction after convergence makes it possible to use the algorithm online.

Due to the stochastic nature of the search algorithm for the robot's best pose estimate, the algorithm is able to cope with a high level of sensor noise with low degradation of the estimation results.

A thresholding mechanism has been also implemented. The consequence is the capability of the RELF-3D to avoid the premature convergence towards some areas, not selecting the new population members that can be originated by the noise. The introduction of a threshold level has an important disadvantage: the decrease of the convergence speed. It was therefore necessary to develop a discarding mechanism to improve the convergence speed while maintaining the positive aspects of the thresholding. The convergence speed after discarding is even higher than with the original method.

A further study to compare the L1-norm and the L2-norm as a cost function has been done in this work. We can conclude that with a gaussian noise the behavior is similar in both cases, while the use of the first order error is useful in certain situations, such as those where there are dynamic objects, people, outliers, and so on.

In addition, the DE has many other characteristics that make its use very interesting: it can deal with non-linear state space dynamics and noise distributions, it does not require any assumptions on the shape of the posterior density, and so on.

One of the most important shortcomings in population-based localization algorithms is the initialization problem, which consists of the estimation of the population size that solves the localization with a reasonable computational cost. This problem has traditionally been adjusted in an empirical way, choosing a number of elements that works well in a particular case depending on many factors: the environment, the sensor characteristics, the type of method, etc.

An initialization method based on the information contained in the observation vector has been implemented. The initialization problem for our evolutionary-based global localization filter has been solved in a satisfactory way.

Some factors that affect the initial population size required to localize the robot have been studied. It has been deduced that symmetries of the environment, sensors information, and sensor overlapping are crucial factors to determine the initial population and guarantee that the population-based localization algorithm works in a robust way.

An empirical formulation (depending on the cited factors) to determine an appropriate size for the initial population has been demonstrated to be efficient in different environments.

The proposed method has found some difficulties when there are many symmetries in the environment or the available information is low, which is logical if we take into account the characteristics of the problem addressed.

As shown in the experimental results, the initial population size is large enough to deal with the localization problem but the minimum possible in order to optimize the computational cost.

6.2 Scan Matching

A DE-based scan matching algorithm for 3D environments (6 DOF) has been implemented. If the cost function of DE is properly chosen, it is possible to solve the scan matching problem. The high accuracy and computational efficiency of the proposed method have been demonstrated with experimental results.

The local accuracy is high enough to apply this algorithm in manipulation tasks. It is a logical result because a high accuracy has been obtained by a similar method for robot global localization.

The algorithm could be used online depending on the time needed between two consecutive readings. The computational cost is the most important shortcoming of the 3D mapping and our algorithm presents a good performance in this aspect because it does not depend on the DOF but on the perceptive sensor number of measurements and the population size.

The DE has many other characteristics that make its use very interesting: the computational resources are focused on the most relevant areas, the algorithm is able to cope with a high level of sensor noise with low degradation of the estimation results, and so on.

6.3 Loop Detection

A loop detection algorithm that extracts the most important features from two different scans has been implemented in order to obtain an indicator that is used as a threshold to detect when the robot is visiting a known place.

The LPI has been demonstrated to be an efficient tool to be used in mapping problems. All true loops are correctly detected and no false detections are appreciated when the mobile robot is covering a long trajectory and there is one place that is visited several times.

The results are similar or even better than those obtained by other research groups when analyzing the probability of detection. The introduction of different types of features and the uncertainty band improve the algorithm performance and make it a more versatile method because it admits different settings.

Since the LPI formula does not depend on any previous state and its weights are updated online, the algorithm can be classified as an online learning method, which is a more appropriate approach for an autonomous robot when exploring its surroundings. The mobile robots do not have a wide amount of perceptive data at the beginning, which makes it difficult to apply machine learning algorithms to build loop detection classifiers.

In addition, a wide variety of scan properties can be contained in this descriptor and different weights have been tested. First, the *surface* features include the geometric forms of the scan (lines, planes and spheres). Second, the *numerical* features are values that describe other several properties, such as: volume, average range, curvature, etc. A detailed study of the descriptor and the influence of different factors have been shown in the experimental results.

The computational cost of the proposed method makes it possible to use the algorithm in real-time applications.

6.4 Future Developments

This work presents some challenges to be accomplished in the future:

- It is necessary to check the DE-based global localization algorithm in different situations and environments in order to optimize its performance.
- The computational cost has to be improved to make the proposed methods applicable to larger and more complex environments.
- There are additional factors and information models that have to be considered to generate a more efficient formula when initializing the population.
- Although the loop detection problem has been successfully solved, it is necessary to address the loop closure problem.
- The introduction of adaptive weights depending on the incoming information and the feature importance (features with more influence on the matching process). The future indicator defined in this way will change its weights depending on the environment.
- The versatility of LPI formula allows the introduction of very different characteristics. It is possible to try new values and combinations of features, selecting the best options.
- There are more and more sensors that are available in the present time. It is necessary to study these new sensors and their integration in the mapping process.
- The scan matching algorithms traditionally have troubles with sharp turns. It can be interesting to study the behavior of our algorithm in this aspect.

Appendix A

Experimental Platform

The experimental platform MANFRED-2, fully developed at the Robotics Lab of the Carlos III University of Madrid, is presented in this appendix. Most of this information and figures have been taken from the work by Álvarez [113].

The mobile robot MANFRED-2 is a mobile manipulator whose purpose is to serve as experimental platform for R&D in the mobile robots area.

One of the main objectives of this research is to build an autonomous robot for an indoor office area. In other words, MANFRED-2 must be able to navigate autonomously in an environment typically composed of a corridor and offices. For example, one specific task that the robot must perform is to move from one room to another by opening a door.

This robot has been built because it is necessary to have an experimental platform with a robust and reliable hardware that allows researchers to focus on the real problem: the implementation of an artificial intelligence that allows the robot to be autonomous and perform multiple tasks.

The robot design is inspired by planetary rovers and communications satellites. These systems are composed of several subsystems that need to be interconnected to make the whole system work. These subsystems are: onboard computer, power distribution system, sensors, drive system, etc. More instruments to explore the surroundings, such as articulated arms, can also be implemented depending on its application, but it is necessary to distinguish between the mobile platform and the inserted accessories. One important characteristic is that the subsystems are designed as independent units or boxes that are interconnected to each other by an internal wiring.

Summarizing, the design of MANFRED-2 is based on independent units that are interconnected to each other by using electric and mechanical interfaces. This modular concept facilitates the integration, repair, and future expansion of the robot.

MANFRED-2 is presented in Figure A.1. It was also shown in Figure 1.2. It has at most eight DOF. It is composed of a differential-type mobile base with two DOF and an anthropomorphic light arm with six DOF. It can execute multiple tasks. The most typical ones are opening and passing through doors, obstacle avoidance, and picking up and manipulating objects. In order to do that, the robot needs all the basic capabilities to move safely and independently around the environment, motor coordination between the base and manipulator, and sensory coordination to manipulate objects.

As was previously said in this appendix, any robotic system consists of a set of subsystems that enable (through networking) meeting the objectives for which it was designed. These modules use the environment information to generate data that are used to develop the movement skills in the robot's base and the robotic arm. The main components of the systems that constitute the mobile manipulator are described in the following sections.

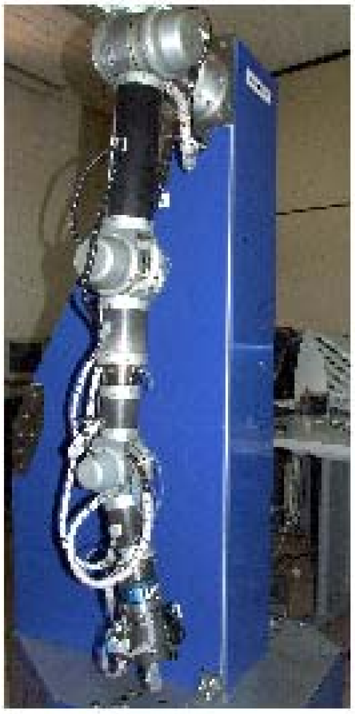

Figure A.1: MANFRED-2, mobile manipulator with robotic arm.

A.1 Mechanical Design - Robot Structure

The design of the mobile robot must meet the following specifications: high mobility, mechanical and electrical robustness, high repeatability in its movements, and easy integration and repairing (modular concept).

A brief description of the mechanical design of MANFRED-2 and a breakdown of the most important elements are given in this section. The mechanical design of the robot's base is also based on the robustness and reliability that must satisfy the robot when it is performing a task. It is crucial that the the robot movement does not cause instability or inaccuracy.

The base has also been designed following a modular philosophy which has two important advantages: it is easy to access to all elements of the mobile robot, and the change of elements due to repairs or improvements is immediate.

The general design also focuses on the improvement of the structure rigidity. The force distribution is more balanced than the distribution of the previous version (MANFRED). The location of the base elements has been optimized in order to counterbalance when the robotic arm is executing critical tasks, which means that the distribution of the elements in the robot's base gives stability to the mobile robot.

Some mechanical characteristics and their associated advantages are given below. Some of them are compared to the previous version of the mobile robot.

- When the arm is at rest, it does not collide neither interfere with the base. If the system runs out of power, the arm can fall freely without damage to itself or to the base.
- The gravity center of the base has been moved closer to the ground. This implies an improvement in the stability.
- The main mast has been extended to the bottom plate and more columns have been placed between the plates. These changes give more rigidity to the system.
- It has independent carcasses that are easy to remove and place. It is easier to access to any component of the mobile robot.
- An internal communication system from the mast to the bottom plate has been designed. This system is simple and facilitates the changes or incorporation of new elements.
- All switches, buttons, and safety mushrooms are located in a single panel. This allows an easy and fast access to each element of the control and security systems.
- A second robotic arm that will be added to the robot has been taken into account, trying to make its future implementation as simple as possible.
- A height adjustment system for the drive wheels has been designed. This allows an accurate calibration.

The robot's weight and the weight of each one of its components are shown in Table A.1. It is important to remark that most of the weight is concentrated in the bottom part, which benefits the stability.

MANFRED-2 is formed by a metal structure that can integrate all the components needed for operation (Figure A.2). It can be divided into three parts:

- Mobile base:

  The robot's base is composed of two steel platforms with a diameter of 61 cm and a height around 65 cm. It is equipped with wheels that allow movement. The battery system that generates the power to operate autonomously is also stored in the base.

Table A.1: MANFRED-2's weight.

Element	Unit weight (kg)	Total weight (kg)
Batteries	15.40	61.60
Aluminum structure	29.00	29.00
Drivers	0.68	5.44
Computer	5.00	5.00
Electronic devices	2.75	2.75
DC-DC Converters	2.00	2.00
Carcasses	2.50	2.50
Caster wheels	0.42	1.26
Drive wheels	7.00	14.00
Wiring	6.00	6.00
Total		129.55

The motion system is included in the base. It has five wheels: three of them are support wheels to improve the stability and facilitate the movement, and the other two are drive wheels with brushless motors and their corresponding servo-amplifiers. The drive wheels generate a differential displacement that allows the robot to turn around its axis.

The power supply system that gives autonomy to the robot consists of batteries that are located in the base. There are four batteries of 12 V connected in series that provide a voltage of 48 V. The selected batteries are Power-Sonic PS-12450 B (Figure A.3), which provide an output voltage of 12 V and a capacity of 45 Ah.

In addition, as a security system, the robot has a monitoring system through a PIC16F818 microcontroller that measures the voltage provided by the batteries and the current flowing through them. This system can continuously communicate the power status to the control computer, as well as stopping the motors in a controlled way in case of low voltage or too high current.

- Body:

  An structure that forms the robot body and holds multiple components has been mounted on the base. The body contains all the wiring for connecting several subsystems: arm to computer, power from battery to motors, and external sensors. It has also the servo amplifiers associated with the arm.

  This structure serves as dock for the robotic arm, the laser sensor, and the computer vision cameras. The onboard computer that is responsible for add intelligence to the robot is also inside this part of the robot. This computer

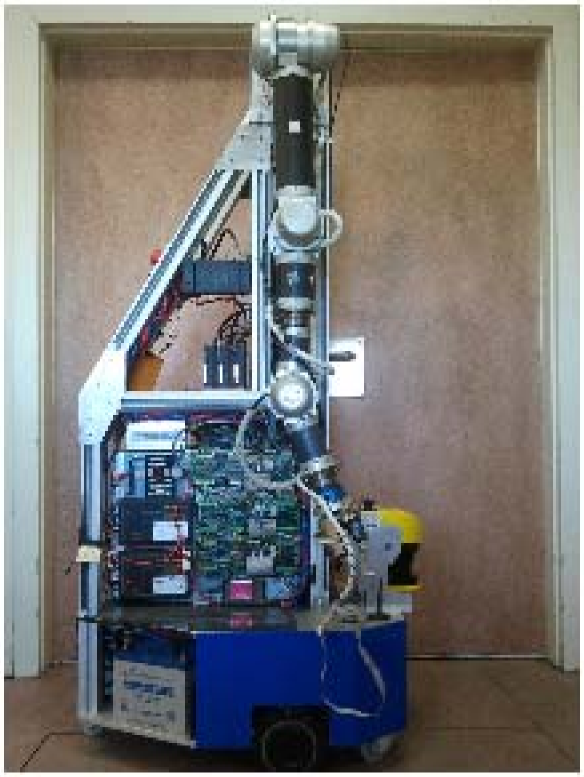

Figure A.2: MANFRED-2, lateral view.

has the PMAC2-PCI card installed, which is a controller card that can control jointly the eight DOF corresponding to the base and the manipulator arm.

- Robotic arm:

  The manipulator arm LWR-UC3M-1 is an essential element of the robot. It is composed of rigid elements connected by revolution joints. Each joint gives an additional DOF to the robot. The total number of DOF is six for the arm. It has been designed to provide a remarkable flexibility to perform manipulation tasks (grasping and and movement of objects) by combining the available DOF.

  The robotic arm that is presented in Figure A.4 has been fully developed by the Robotics Lab of the Carlos III University of Madrid. Its main characteristics are:

  1. Kinematic redundancy similar to the human arm.
  2. Weight: 18 kg.

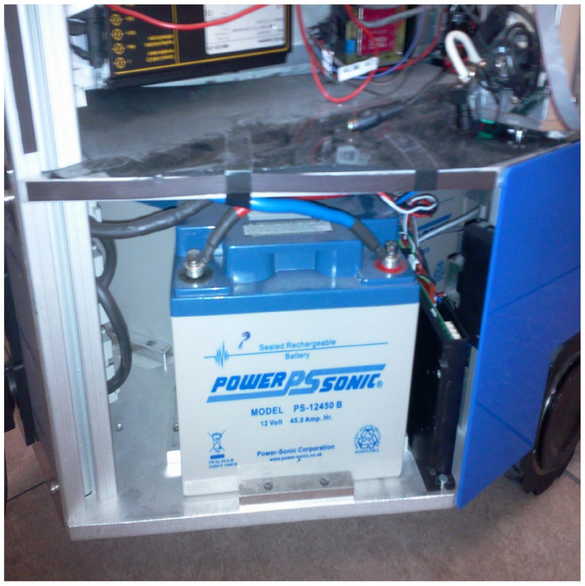

Figure A.3: Power supply system.

3. Maximum load capacity: 4.5 kg at the end of the arm.
4. Load/weight ratio: between 1 : 3 and 1 : 4.
5. Range: around 955 mm.

The developed arm is mounted on the lateral side of the mobile robot in such a way that the computer vision and the laser telemetry systems are not obstructed by the arm. The arm joints are composed of DC brushless motors and Harmonic Drives that reduce the speed and increase the torque.

Since the installed encoders obtain relative information (they provide information about the motor current position with respect to an initial or home position), an initial *home* function must be executed in order to fix the robotic arm initial position. This facilitates the conversion between relative and absolute positions. This function has been designed using the programming language of the PMAC2-PCI. It establishes that the initial position of the robotic arm is that one in which it is pointing straight to the ground. This position has been chosen because it requires a low energy consumption because most of the engines are not doing any work.

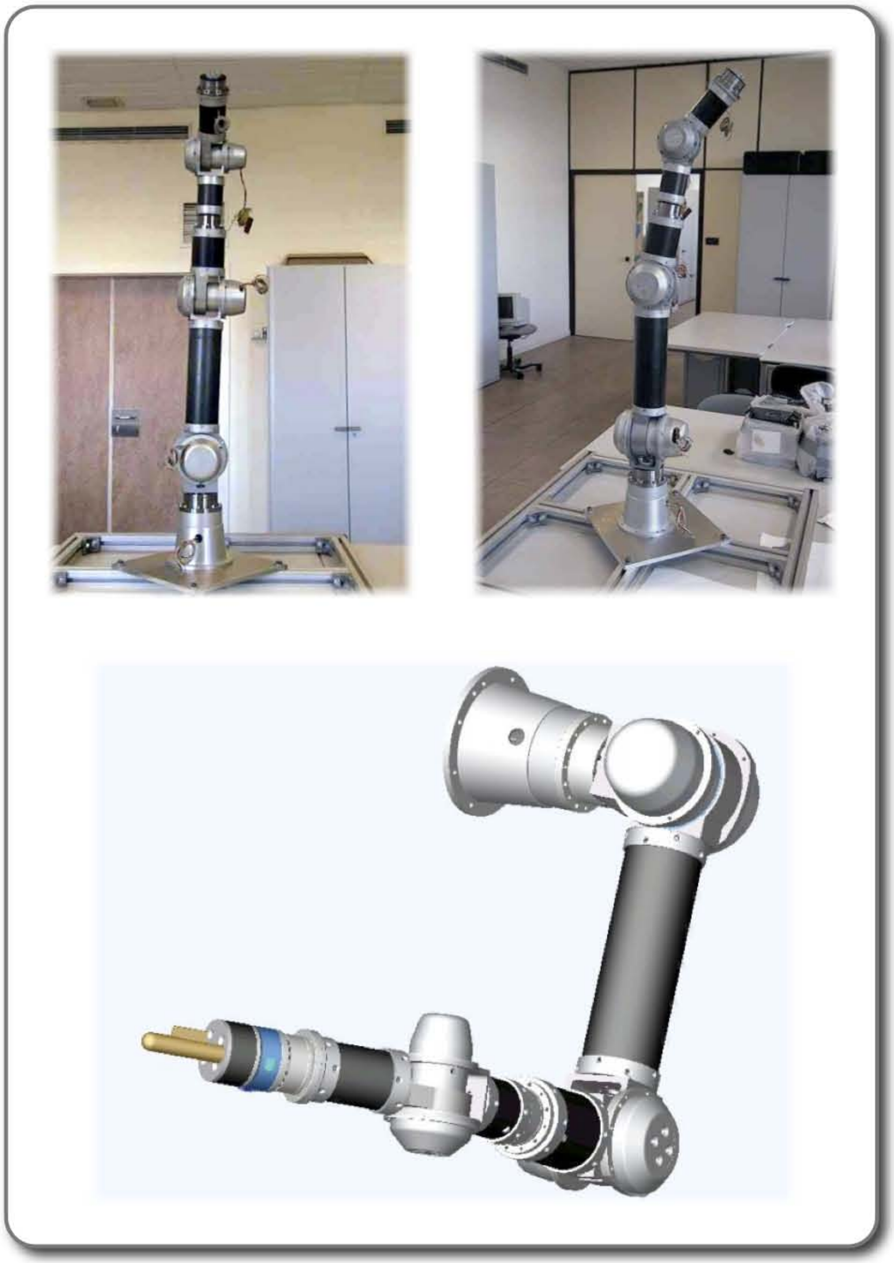

Figure A.4: LWR-UC3M-1(robotic arm).

Figure A.5: Hokuyo UTM-30LX (laser range finder).

A.2 Sensory System

The sensory system can transform the physical variables that characterize the environment into a data set that will be processed by other modules, such as the localization system, the security system, and the motion planner, in order to increase the robot intelligence and be able to execute certain tasks. This information will be provided by the robot's sensory system, which consists of the following elements:

- Laser telemetry subsystem:

  Its aim is to provide the robot with information about its surrounding environment by measuring the distance to objects. This information is primarily used in navigation and localization in order to model the workspace.

  It is possible to use 2D or 3D data depending on the task characteristics and the complexity and degree of occupancy of the workspace.

  This subsystem is composed of the following laser range finders:

  1. Hokuyo UTM-30LX with 270 opening degrees (Figure A.5) located in the rear of the vehicle. It has a detection range that varies from 100 mm to 30 m and a 25 ms period. Its angular resolution is equal to 0.25°. It is connected to the computer through a USB2.0 interface. Its power consumption is 700 mA and 12 V, which makes it suitable for battery-powered systems such as MANFRED-2.
  2. SICK PLS with 180 opening degrees (Figure A.6). The original measurements are 2D, but we have added a motor that lets it rotate up and down

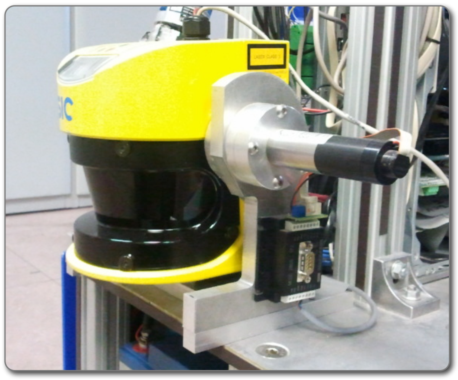

Figure A.6: SICK PLS (laser range finder).

Table A.2: SICK PLS technical characteristics.

Maximum range	80 m
Angular resolution	0.25° - 0.5° - 1° (variable)
Time response	26 ms
Distance resolution	10 mm
Transfer rate	500 kbaud
Power requirements	24 V - 6 A

(±45°), being able to obtain 3D measurements (it can also be observed in the figure). The technical characteristics are summarized in Table A.2.

The 2D telemetry (horizontal plane parallel to the ground) can be used during navigation around environments with few obstacles to safe computational time.

This sensor records 361 measurements in a planar sweep with medium resolution (separation between measurements equal to 0.5°). The SICK PLS measurement error is lower than 20 mm. This error is influenced by two parameters: the measuring distance and the angle of the laser beam shot (from 0° to 180°).

- Computer vision subsystem:

  This subsystem helps in the manipulation of objects in 3D environments, which is one of the abilities of MANFRED-2. In order to do this, it is necessary to recognize the object to be manipulated, estimate its position and orientation

Figure A.7: Color cameras. Left: SONY EVI-D100. Right: SONY B/N XC-ES50CE.

relative to the mobile manipulator, and determine the grasping point. It also facilitates other tasks, such as opening doors, navigation, and localization.

The computer vision subsystem is composed of the following elements:

1. Color camera: SONY EVI-D100 (Figure A.7). This camera is employed to recognize objects and estimate their positions relative to the robot. It is located in the front of the mobile robot body.
2. Color camera: SONY B/N XC-ES50CE (Figure A.7). This is a mini-camera that is situated on the wrist of the robotic arm. It is used in manipulation tasks when the extreme of the arm is close to the object to be manipulated and the field of vision of the other camera is obstructed by the arm.
3. Time-of-flight camera (Kinect): the robot also incorporates a camera with time-of-flight technology (Figure A.8) that obtains a 3D image composed of an array of distances to different objects and color information. This information can be fused with the data of the other cameras in order to improve the manipulation capabilities.

- Force/torque sensor:

  MANFRED-2 has a JR3 force/torque sensor (model 67M25A-U560, Figure A.9) at the end of the robotic arm. Its purpose is to interact with the environment in manipulation tasks. This sensor is situated between the end of the arm and the clamp or terminal element.

  This device has the following features:

Figure A.8: Time-of-flight camera: Kinect.

Figure A.9: JR3 67M25A-U560 (force/torque sensor).

  - Maximum load capacity: 11 kg.
  - Weight: 175 gr.
  - Maximum operating frequency: 8 kHz.

  The JR3 sensor provides force and torque data in three axes that can be used in the force control loop of the mobile manipulator. It is based on a strain gauge system and a Digital Signal Processor (DSP) acquisition system that allow measurements with high bandwidth and signal-noise ratio. The main purpose of this sensor is to perform manipulation tasks based on force or torque control, such as opening doors, pulsation of switches, manipulating objects, etc.

- Motion sensors:

  The main function of these sensors is to obtain information about the robot location and the arm posture. This information is obtained by encoders that are mainly coupled to the rotation axes of the motors. The relative or absolute position of each motor is computed by using this information. The motion sensors are high-resolution optical encoders of the HP company with reference HEDS550.

  These motion sensors are complemented by inductive sensors that perform an

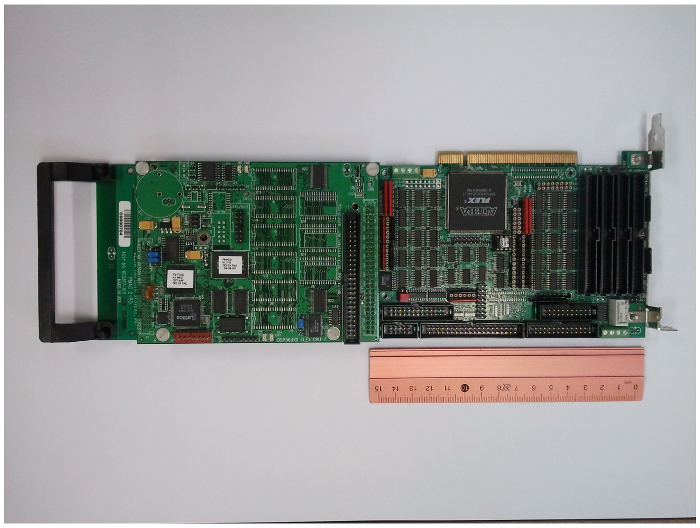

Figure A.10: PMAC2-PCI (controller card).

initial routine that is usually named as *home* in order to establish the absolute position of each joint of the arm. This routine improves the safety and minimizes the power consumption. The inductive sensors have a diameter equal to 3 mm and a detection distance equal to 1 mm. Their basic principle is based on the inductive detection of ferromagnetic materials by flux variation caused by their presence near the sensor's detection area.

A.3 Control System

MANFRED-2 has eight different motors to move its base (2) and its robotic arm (6). It is necessary to have a continuous control of these engines when the robot is navigating or it is moving its arm. This control is carried out by the PMAC2-PCI controller card (Figure A.10).

The PMAC2-PCI is a Programmable Multi-Axis Controller card developed by Delta Tau Data Systems[8]. It is a high performance device that can simultaneously control up to eight axes with high precision. It has a high performance/price ratio, with more than 1000 configuration variables and the high computing capacity of its DSP. The DSP that is incorporated in the PMAC2-PCI is the DSP56002 of 24 bits and operation frequency of 40 MHz.

[8]`http://www.deltatau.com`

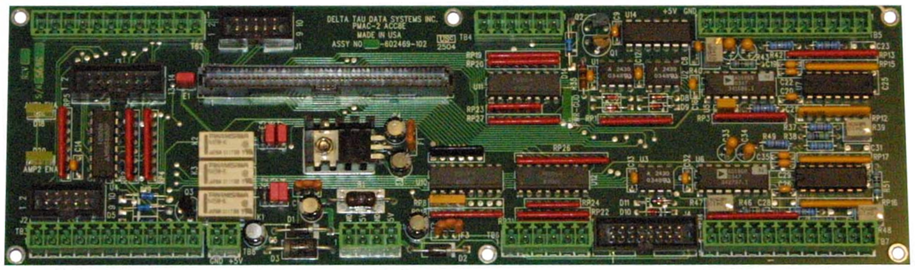

Figure A.11: ACC-8E. Interface between the PMAC2-PCI and the devices.

This card offers multiple ways to control the motors. However, it has not been designed to be connected directly to the devices. There is a set of additional cards that can be used as interfaces . These cards are also offered by Delta Tau Data Systems.

In the case of MANFRED-2, the additional card is the ACC-8E (Figure A.11). Since each card can interact with two motors, it is necessary to implement four of them. Each ACC-8E card is connected to the PMAC2-PCI through a 100-pin bus that is called JMACH. Each ACC-8E card has four 18-bit Digital-to-Analog Converters (DAC) that command two analog input drivers and must be fed with 15 V. It has also two inputs to read the encoders and five inputs per axis that capture different types of events: error signal, *home* signal (starting position), motor limits (two signals), and user-defined signal (external events for a specific application).

The configuration of the PMAC2-PCI is a very laborious and tough task. There are two available manuals, the “Software reference manual” and the “PMAC2 user manual”, together with a program provided by the manufacturer, the “PEWIN32 PRO”, which runs under Windows. This software offers a set of tools to modify all the configuration parameters of the PMAC2-PCI. Some of these tools are:

1. Terminal: it sends commands to the card in ASCII coding.
2. Watch window: it is a window where it is possible to view the variable values in real time.
3. Tunning Pro: it configures the PMAC2-PCI parameters, such as: PID controllers, filters, DAC calibration, and so on.
4. Position window: it displays the position of the motors, in counts of encoder, and also their speed and tracking errors.

Finally, the controller card allows different types of programs:

- Motion programs: the most common task of the controller card is to move the motors according to a particular sequence of commands. These programs are executed line by line by the controller card. They are called by a specific command and, after that, they are executed once. It is possible to make a call to another program or terminal commands. The controller card can store and execute up to 256 motion programs.

- Programmable Logic Controller (PLC): the PLC programs exist because there are some programs that must be executed continuously. For example, there is a PLC program that computes the robot's position given the encoders information. These programs are written in the same way that the motion programs, except that they are defined as PLC in their title. They are called and executed in each cycle of the controller card.

- Motion commands: it is possible to send motion commands to the PMAC2-PCI through the terminal. They are simple commands that allow the motion of each motor. These commands were initially implemented to test the controller card, but they can perform simple movements in a motion program.

A.4 Software

A.4.1 MATLAB

MATLAB (abbreviation of MATrix LABoratory)[9] is a numerical computing environment developed by MathWorks. It is oriented to projects that imply high computation resources and graphical display. It allows multiple actions, such as: manipulation of matrix and vectors, handling and plotting of functions and data, implementation of algorithms, creation of graphical interfaces, and interfacing with programs in other languages (C, C++, Java, and Fortran).

One additional advantage of this tool is that it is very easy to learn, not being necessary to study a new language because the solutions are expressed by an easy syntax (similar to C).

MATLAB includes a wide range of pre-built functions called "toolboxes". These toolboxes perform multiple operations of multiple areas of engineering and simulation, such as: signal processing, control, statistics, financial analysis, symbolic mathematics, neural networks, fuzzy logic, system identification, dynamic systems simulation, and so on. An additional package called "Simulink" offers a graphical interface for these toolboxes. It allows the simulation of dynamic models.

[9]More information can be found in `http://www.mathworks.es/products/matlab/`.

This tool is widespread in engineering, science, and economics. It has been reported that it had around one million users in 2004. It is also widely used in academic and research institutions.

All these features make MATLAB a suitable tool to be used for our purposes. All the algorithms developed in this work have been implemented in MATLAB.

A.4.2 ROS

ROS (Robot Operating System)[10] is an open code operating system for robots developed by Willow Garage. As it is said in its website, "it provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license (Berkeley Software Distribution, family of permissive free software licenses)".

ROS is based on a set of processes or nodes that are individually executed and linked by a communication infrastructure provided by ROS. This communication can be synchronous (client-server) or asynchronous (continuous data sending). The different data can be grouped into packages that are shared allowing a distributed collaboration.

The most remarkable characteristics are the following: light and easy to export (it has been exported to OpenRAVE, Orocos, and Player), programming language independent (it can be implemented in the most common languages, such as C++ and Python), easy error correction (because it has a testing unit), and appropriate in big systems with multiple modules.

It currently only works with Unix-based platforms. It has been extensively tested on Ubuntu (operating system of MANFRED-2).

ROS has been implemented in MANFRED-2. All modules developed for the robot must follow its guidelines.

[10]More information can be found in `http://www.ros.org/wiki/`.

Bibliography

[1] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, p. 341–359, December 1997.

[2] J. J. Leonard and H.Durrant-Whyte, "Mobile Robot Localization by Tracking Geometric Beacons," *IEEE Transaction on Robotics and Automation*, vol. 7, pp. 376–382, 1991.

[3] R. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," in *Proceedings of the Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI'86)*, (New York, NY), pp. 267–288, Elsevier Science, 1986.

[4] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker, "A System for Volumetric Robotic Mapping of Abandoned Mines," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'03)*, 2003.

[5] B. Siciliano and O. Khatib, *Springer handbook of robotics.* Springer, 2008.

[6] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'08)*, (Pasadena, CA, USA), 2008.

[7] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg, "6D SLAM - Preliminary Report on Closing The Loop in Six Dimensions," in *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV'04)*, (Lisbon), 2004.

[8] L. Moreno, S. Garrido, F. Martín, and M. L. Muñoz, "Differential Evolution approach to the grid-based localization and mapping problem," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, (San Diego, USA), pp. 3479–3484, 2007.

[9] S. Thrun, “Robotic mapping: A survey,” in *Exploring Artificial Intelligence in the New Millenium*, Morgan Kaufmann, 2002.

[10] U. Frese and G. Hirzinger, “Simultaneous Localization and Mapping - A Discussion,” in *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 2001.

[11] J. Folkesson and H. I. Chistensen, “Outdoor Exploration and SLAM using a Compressed Filter,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’03)*, 2003.

[12] S. Thrun, W. Burgard, and D. Fox, “A probabilistic approach to concurrent mapping and localization for mobile robots,” *Machine Learning and Autonomous Robots*, vol. 31, no. 5, pp. 1–25, 1997.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[14] M. W. M. G. Dissanayake, P. N. S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A Solution to the Simultaneous Localization and Map Building Problem,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[15] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[16] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, “Simultaneous Localization and Mapping with sparse extended information filters,” *The International Journal of Robotics Research*, vol. 23, p. 693, 2004.

[17] S. Thrun, W. Burgard, and D. Fox, “A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping.,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’00)*, 2000.

[18] F. Lu and E. Milios, “Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans,” *Journal of Intelligent and Robotic Systems*, vol. 20, pp. 249–275, 1997.

[19] C. Früh, M. v Ehr, and R. Dillmann, “Aufbereitung von 3D Laserscans für ein autonomes, mobiles Messystem,” in *In Proceedings of the Fachgespräch Autonome Mobile Systeme (AMS’00)*, 2000.

[20] H. Zhao and R. Shibasaki, "Reconstructing Textured CAD Model of Urban Environment Using Vehicle-Borne Laser Range Scanners and Line Cameras," in *Second International Workshop on Computer Vision Systems*, 2001.

[21] O. Wulf, K. O. Arras, H. I. Christensen, and B. A. Wagner, "2D Mapping of Cluttered Indoor Environments by Means of 3D Perception," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)*, (New Orleans, USA), April 2004.

[22] S. Thrun, M. Montemerlo, and A. Aron, "Probabilistic Terrain Analysis For High-Speed Desert Driving," in *Robotics: Science and Systems*, (Cambridge, USA), 2006.

[23] A. Nüchter, K. Lingemann, and J. Hertzberg, "6D SLAM-3D Mapping Outdoor Environments," *Journal of Field Robotics*, vol. 24, pp. 699–722, 2007.

[24] P. J. Besl and N. D. McKay, "A method for registration of 3d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[25] A. Nüchter, S. Gutev, D. Borrmann, and J. Elseberg., "Skyline-based Registration of 3D Laser Scans," *Journal of Geo-spatial Information Science, Special Issue with selected papers from the 3D City Modeling and Applications Workshop*, vol. 14, pp. 85–90, June 2011.

[26] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, "Globally consistent 3D mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, pp. 130–142, 2008.

[27] R. Triebel, P. Pfaff, and W. Burgard, "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*, 2006.

[28] D. M. Cole and P. M. Newman, "Using Laser Range Data for 3D SLAM in Outdoor Environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'06)*, 2006.

[29] V. Sequeira, K. Ng, E. Wolfart, J. Goncalves, and D. Hogg, "Automated 3D reconstruction of interiors with multiple scan-views," in *Proceedings of the SPIE's 11th Annual Symposium, Electronic Imaging '99, The Society for Imaging Science and Technology*, (San Jose, California, USA), 1999.

[30] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer, "AVENUE: Automated Site Modeling in Urban Environments," in *Proceedings of the third International Conference on 3D Digital Imaging and Modeling (3DIM '01)*, (Quebec City, Canada), 2001.

[31] A. Georgiev and P. K. Allen, "Localization methods for a mobile robot in urban environments," *IEEE Transaction on Robotics and Automation*, vol. 20, pp. 851–864, October 2004.

[32] M. Hebert, M. Deans, D. Huber, B. Nabbe, and N. Vandapel, "Progress in 3-D Mapping and Localization," in *Proceedings of the 9th International Symposium on Intelligent Robotic Systems*, (Toulouse, France), 2001.

[33] M. Magnusson and T. Ducket, "A Comparison of 3D Registration Algorithms for Autonomous Underground Mining Vehicles," in *Proceedings of the Second European Conference on Mobile Robotics*, (Ancona, Italy), 2005.

[34] P. Biber and W. Straßer, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, 2003.

[35] P. Biber, H. Andreasson, T. Duckett, and A. Schilling, "3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04)*, (Sendai, Japan), 2004.

[36] S. Se, D. G. Lowe, and J. J. Little, "Local and Global Localization for Mobile Robots using Visual Landmarks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, (Hawaii, USA), 2001.

[37] I. J. Cox and Blanche, "An Experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Transaction on Robotics and Automation*, vol. 7, pp. 193–204, 1991.

[38] J. L. Crowley, "World modelling and position estimation for a mobile robot using ultrasonic ranging," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'89)*, (Scottsdale, AZ, USA), 1989.

[39] P. Jensfelt, *Approaches to mobile robot localization in indoor environments*. PhD thesis, Royal Institute of Technology, Sweden, 2001.

[40] W. Burgard, D. Fox, D. Henning, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'96)*, 1996.

[41] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, no. 11, pp. 391–427, 1999.

[42] J. Reuter, "Mobile robot self-localization using PDAB," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00)*, 2000.

[43] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte Carlo Localization for Mobile Robots," *Artificial Intelligence*, vol. 128, pp. 99–141, 2001.

[44] F. Dellaert, D. F. W. Burgard, and S. Thrun, "Monte Carlo Localization for Mobile Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'99)*, pp. pp. 1322–1328, 1999.

[45] L. Moreno, S. Garrido, and M. L. Muñoz, "Evolutionary Filter for Robust Mobile Robot Localization," *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 590–600, 2006.

[46] A. R. Vahdat, N. N. Ashrafoddin, and S. S. Ghidary, "Mobile Robot Global Localization using Differential Evolution and Particle Swarm Optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC'07)*, 2007.

[47] K. O. Arras, J. A. Castellanos, and R. Siegwart, "Feature-based multi-hypothesis localization and tracking for mobile robots using geeometric constraints," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'02)*, (Washington DC, USA), pp. 1371–1377, 2002.

[48] D. J. Austin and P. Jensfelt, "Using multiple Gaussian hypotheses to represent probability distributions for mobile robot localization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00)*, (San Francisco, USA), pp. 1036–1041, 2000.

[49] I. J. Cox and J. J. Leonard, "Modeling a dynamic environment using a Bayesian multi hypothesis approach," *Artificial Intelligence*, vol. 66, pp. 311–44, 1994.

[50] S. I. Roumeliotis and G. A. Bekey, "Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00)*, (San Francisco), pp. pp 2985–2992, 2000.

[51] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, "Monte Carlo Localization in Outdoor Terrains using Multi-Level Surface Maps," *Journal of Field Robotics*, vol. 42, pp. 213–222, 2008.

[52] K. Lingemann, A. Nüchter, J. Hertzberg, and H. Surmann, “High-speed laser localization for mobile robots,” *Robotics and Autonomous Systems*, vol. 51, pp. 275–296, 2005.

[53] C. C. Tsai, H. H. Lin, and S. W. Lai, “Multisensor 3D Posture Determination of a Mobile Robot Using Inertial and Ultrasonic Sensors,” *Journal of Intelligent and Robotic Systems*, vol. 42, p. 317–335, 2005.

[54] L. C. Lai, T. L. Lee, H. T. Fan, and C. J. Wu, “A Nonlinear Programming Method for 3D Localization of Mobile Robots,” in *Proceedings of the International Conference on Advanced Robotics*, 2005.

[55] M. Agrawal and K. Konolige, “Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS,” in *Proceedings of the International Conference on Pattern Recognition (ICPR’06)*, 2006.

[56] N. Ho and R. Jarvis, “Vision Based Global Localisation Using a 3D Environmental Model Created by a Laser Range Scanner,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’08)*, (Acropolis Convention Center, Nice, France), September 2008.

[57] E. Royer, M. Lhuillier, M. Dhome, and J. M. Lavest, “Monocular Vision for Mobile Robot Localization and Autonomous Navigation,” *International Journal of Computer Vision*, vol. 74, pp. 237–260, 2007.

[58] S. Se, D. G. Lowe, and J. J. Little, “Vision-Based Global Localization and Mapping for Mobile Robots,” *IEEE Transactions on Robotics*, vol. 21, p. 3, 2005.

[59] M. Sridharan, G. Kuhlmann, and P. Stone, “Practical Vision-Based Monte Carlo Localization on a Legged Robot,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’05)*, vol. 3, p. 3366, 2005.

[60] D. G. Lowe, “Object Recognition from Local Scale-Invariant Features,” in *Proceedings of the Seventh International Conference on Computer Vision (ICCV’99)*, 1999.

[61] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, “SURF: Speeded Up Robust Features,” *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.

[62] M. Tomono, “A scan matching method using euclidean invariant signature for global localization and map building,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’04)*, 2004.

[63] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, pp. 119–152, 1994.

[64] J. Nieto, T. Bailey, and E. Nebot, "Recursive scan-matching SLAM," *Robotics and Autonomous Systems*, vol. 55, pp. 39–49, 2007.

[65] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," in *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling*, 2001.

[66] A. Diosi and L. Kleeman, "Laser Scan Matching in Polar Coordinates with Application to SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, 2005.

[67] G. Weiss and E. Puttkamer, "A map based on laser scans without geometric interpretation," in *Proceedings of the 4th International Conference on Intelligent Autonomous Systems (IAS-4)*, 1995.

[68] O. Bengtsson and A. Baerveldt., "Localization by matching of range scans - certain or uncertain?," in *Eurobot'01 - Fourth European Workshop on Advanced Mobile Robots*, (Lund, Sweden), September 2001.

[69] F. T. Ramos, J. Nieto, and H. F. Durrant-Whyte, "Recognising and modelling landmarks to close loops in outdoor SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'07)*, (Rome, Italy), 2007.

[70] M. Cummings and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, pp. 647–665, June 2008.

[71] M. Cummings and P. Newman, "Probabilistic appearance based navigation and loop closing," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'07)*, 2007.

[72] M. Cummings and P. Newman, "Accelerated appearance-only SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'08)*, 2008.

[73] M. Cummings and P. Newman, "Highly scalable appearance-only SLAM—FAB-MAP 2.0," in *Proceedings of Robotics: Science and Systems*, (Seattle, WA), 2009.

[74] A. E. Johnson, *Spin-images:A representation for 3-D surface matching.* PhD thesis, Carnegie Mellon University, 1997.

[75] D. F. Huber, *Automatic three-dimensional modeling from reality.* PhD thesis, Carnegie Mellon University, 2002.

[76] M. Magnusson, H. A. A. Nüchter, and A. J. Lilienthal, "Automatic Appearance-Based Loop Detection from Three-Dimensional Laser Data Using the Normal Distributions Transform," *Journal of Field Robotics*, vol. 26, p. 892–914, 2009.

[77] K. Granström, T. B. Schön, J. I. Nieto, and F. T. Ramos, "Learning to close loops from range data," *The International Journal of Robotics Research*, vol. 0, p. 0, 2011.

[78] Y. Freund and R. E. Schaphire, "A Decision-Thepretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[79] M. Bosse and J. Roberts, "Historgram Matching and Global Initialization for Laser-only SLAM in Large Unstructured Environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'07)*, (Roma, Italy), 10-14 April 2007.

[80] F. Martín, L. Moreno, S. Garrido, and D. Blanco, "Localization in 3D Environments Using Differential Evolution," in *Proceedings of the International Symposium on Intelligent Signal Processing (WISP'09)*, 2009.

[81] F. Martín, L. Moreno, S. Garrido, and D. Blanco, "High-Accuracy Global Localization Filter for three-dimensional Environments," *Robotica*, vol. 0, pp. 1–16, 2011.

[82] S. Thrun, "Bayesian landmark learning for mobile robot localization," *Machine Learning*, vol. 33, no. 1, pp. 41–76, 1998.

[83] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods. Volume I: Basics.* John Wiley and Sons, 1986.

[84] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization.* Springer, 2005.

[85] G. Sywerda, "Uniform crossover in genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[86] M. A. Potter and K. A. DeJong, "A cooperative co-evolutionary approach to function optimization," *Proceedings of parallel problems solving from nature 3*, vol. 0, pp. 249–257, 1994.

[87] D. Zaharie, "Critical values for the control parameters of differential evolution algorithms," in *Proceedings of MENDEL 2002, 8th international conference on soft computing*, (Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, Czech Republic), pp. 62–67, June 5–7 2002.

[88] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning.* Addison Wesley Publishing Company, 1989.

[89] T. Krink, B. Filipic, and G. B. Fogel, "Noisy optimization problems - a particular challenge for differential evolution?," in *Proceedings of the Congress on Evolutionary Computation (CEC'04)*, 2004.

[90] S. Markon, D. V. Arnold, T. Back, T. Beielstein, and H.-G. Beyer, "Thresholding-a selection operator for noisy ES," in *Proceedings of the Congress on Evolutionary Computation (CEC'01)*, 2001.

[91] F. Martín, C.G.Uzcátegui, L.Moreno, and D.Blanco, "Accelerated Localization in Noisy 3D Environments using Differential Evolution," in *Proceedings of the International Conference on Genetic and Evolutionary Methods*, (Las Vegas, USA), July 2010.

[92] L. Moreno, D. Blanco, M. L. Muñoz, and S. Garrido, "L1–L2-norm comparison in global localization of mobile robots," *Robotics and Autonomous Systems*, vol. 59, pp. 597–610, 2011.

[93] T. Jansen, K. A. D. Jong, and I. Wegener, "On the Choice of the Offspring Population Size in Evolutionary Algorithms," *Evolutionary Computation*, vol. 13, no. 4, pp. 413–440, 2005.

[94] A. Doucet, "On squential simulation-based methods for Bayesian filtering," tech. rep., CUED/FINFENG/TR 31,Cambridge University, Dept of Engineering, Cambridge, UK, 1998.

[95] L. Marchetti, G. Grisetti, and L. Iocchi, "A Comparative Analysis of Particle Filter based Localization Methods," in *Proceedings of the RoboCup Symposium*, 2006.

[96] D. Fox, "KLD-Sampling: Adaptive Particle Filters and Mobile Robot Localization," tech. rep., University of Washington, 2001.

[97] D. Koller and R. Fratkina, “Using learning for approximation in stochastic processes,” in *Proceedings of the International Conference on Machine Learning (ICML’98)*, 1998.

[98] G. Grisetti, C. Stachniss, and W. Burgard, “Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling,” in *Proceedings og the IEEE International Conference on Robotics and Automation (ICRA’05)*, 2005.

[99] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, “Information Based Adaptive Robotic Exploration,” in *Proceedings of the IEEE/RSJ International Conference on lntelligent Robots and Systems (IROS’02)*, 2002.

[100] B. Grocholsky, *Information-Theoretic Control of Multiple Sensor Platforms.* PhD thesis, Australian Centre for Field Robotics, Department of Aerospace, Mechatronic and Mechanical Engineering, The University of Sydney, 2002.

[101] R. Rocha, J. Dias, and A. Carvalho, “Cooperative multi-robot systems: A study of vision-based 3-D mapping using information theory,” *Robotics and Autonomous Systems*, vol. 53, pp. 282–311, 2005.

[102] B. Jensen, J. Weingarten, S. Kolski, and R. Siegwart, “Laser Range Imaging using Mobile Robots: From Pose Estimation to 3d-Models,” in *Proceedings of the 1st Range Imaging Research Day*, pp. 129–144, 2005.

[103] J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos, “The SPmap: a probabilistic framework for simultaneous localization and map building,” *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 948–952, 1999.

[104] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, “Appearance-Based Loop Detection from 3D Laser Data Using the Normal Distributions Transform,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’09)*, (Kobe International Conference Center, Kobe, Japan), May 2009.

[105] K. Granström and T. B. Schön, “Learning to Close the Loop from 3D Point Clouds,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’10)*, (Taipei International Conference Center, Taipei), October 2010.

[106] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, pp. 509–517, 1975.

[107] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein., *Introduction to Algorithms.* McGraw-Hill, 2009.

[108] D. T. Lee and C. K. Wong, "Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees," *Acta Informatica*, vol. 1977, pp. 23–29, 1977.

[109] R. E. Bellman, *Dynamic programming.* Princeton University Press, 1957.

[110] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry (Algorithms and Applications).* Springer-Verlag, 1998.

[111] H. Surmann, A. Nüchter, and J. Hertzberg., "An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45 (3-4), pp. 181–198, 2003.

[112] F. Martín, M. L. Muñoz, S. Garrido, D. Blanco, and L. Moreno, "L1-norm global localization based on a Differential Evolution Filter," in *Proceedings of the International Symposium on Intelligent Signal Processing (WISP'09)*, 2009.

[113] D. Álvarez, "Controlador cartesiano para el brazo LWR-UC3M-1 del robot MANFRED con detección de contacto," Master's thesis, Carlos III University of Madrid, 2011.