Filipe Manuel Marques Quintas

Laser-based Localization Methods for Mobile Robots: a SLAM Perspective

Dissertação de Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Coimbra, Setembro, 2017

Universidade de Coimbra

University of Coimbra

Faculty of Sciences and Technology

Department of Electrical and Computer Engineering

Laser-based Localization Methods for Mobile Robots: a SLAM Perspective

Filipe Manuel Marques Quintas

A dissertation presented for the degree of

Master of Science in Electrical and Computer Engineering

Coimbra, 2017

Localization Methods for Mobile Robots using Laser Data: a SLAM Perspective

Supervisor:

Prof. Doutor Urbano José Carreira Nunes

Co-Supervisor:

Master Luís Garrote

Jury:

Prof. Doutor Paulo Jorge Carvalho Menezes

Prof.ª Doutora Ana Cristina Barata Pires Lopes

Prof. Doutor Urbano José Carreira Nunes

A dissertation submitted in partial satisfaction of the requirements for the degree of Master of Science in Electrical and Computer Engineering

Coimbra, 2017

Acknowledgements

First of all, I would like to thank Prof. Urbano Nunes for all the conditions and help provided during not just this dissertation, but also during the whole master's degree. I must also thank all my ISR laboratory colleagues for helping every time I needed during this work, specially to Luís Garrote which advised me during the entire work and without which it would not be possible to complete this dissertation.

I'm grateful to the support provided by the project P2020 AGVPOSYS (*Automated-Guided-Vehicle with innovative indoor positioning system for the factory of the future*; co-funded by FEDER, through programs PT2020 and Centro 2020), in co-promotion between Active Space Technologies and University of Coimbra, project in which this dissertation was inserted, and also to ISR-UC for giving me the conditions to make this dissertation possible.

This journey would certainly be impossible without the support of my parents who I specially thank for always being there and giving me everything i needed to reach this point, my sister who has always looked out for me and lead me into the right direction, my girlfriend who I had the luck to have by my side during this journey and my friends who have been with me the whole time. This is just the endpoint of a long journey, so it is impossible to thank by words to all these persons that have been with me since the beginning. I can only hope i still get this support for what's to come.

For those who helped me completing this dissertation, those who have been there for me all the time, those who had the patient to always support me and those who I was happy to come across in this journey: thank you!

Abstract

A robot navigation system is composed of two essential modules: localization and mapping. Localization consists on tracking the robot's pose (position and orientation) in a known environment, while mapping is focused on building a map of the environment. Although there have been many techniques of Simultaneous Localization and Mapping (SLAM) developed over the last years, these modules can be analysed independently. This dissertation focuses on the localization module.

One of the technologies that has been emerging over the last century is the use of Automated Guide Vehicles (AGVs) for industrial purposes (pick up and move materials around a warehouse). These vehicles are usually guided by wires in the floor, magnetic tapes or laser. Even though using wires or magnetic tapes may be more reliable, laser navigation does not require the application of any material on the environment which constitutes a more versatile solution. This dissertation focuses on developing a robot's localization system using only laser data. This is accomplished using a scan matching algorithm (Iterative Closest Point) to match a map built from environment natural features (essentially corners) with a known map of the environment. An algorithm to extract corners is developed, which is later compared with already validated feature extraction methods available in OpenSLAM. Lastly, the developed algorithm is applied in the framework of the HectorSLAM method in order to give the localization technique more reliability and robustness.

Keywords: SLAM, Localization, Mapping, AGV, Scan Matching, the HectorSLAM.

Resumo

O sistema de navegação de um *robot* é constituído por dois módulos essenciais: localização e mapeamento. A localização consiste em determinar a pose do robot (posição e orientação) num ambiente previamente conhecido, enquanto que o mapeamento foca-se na construção de um mapa do ambiente. Embora existam várias técnicas de SLAM (localização e mapeamento em simultâneo), estes dois módulos podem ser analizados independentemente. Esta dissertação foca-se essencialmente no sistema de localização do *robot*.

Uma das tecnologias que tem estado emergente nos últimos anos é o uso de AGVs (veículos que se deslocam sem ação humana direta) em ambientes industriais, principalmente para deslocar objetos. Este tipo de veículos é, geralmente, mobilizado usando fios no chão, fita magnética ou laser. Embora o seguimento com fios ou fita magnética possa ser mais fiável, a navegação por laser dispensa o uso de materiais adicionais no ambiente, o que constitui uma solução mais versátil. Esta dissertação foca-se em desenvolver um sistema de localização que usa apenas as leituras do laser. Isto é conseguido utilizando um algoritmo de *Scan Matching*, que faz a correspondência entre um mapa de *features* naturais (essencialmente cantos) e um mapa previamente conhecido do ambiente. É desenvolvido um algoritmo de extração de cantos que é posteriormente comparado com outros já validados disponíveis no OpenSLAM. Finalmente, este algoritmo desenvolvido é utilizado em conjunto com o HectorSLAM com o propósito de aumentar a fiabilidade e robustez do sistema de localização.

Palavras-chave: SLAM, Localização, Mapeamento, AGV, *Scan Matching*, the HectorSLAM.

"The important thing is to never stop questioning."

Albert Einstein

Contents

List of Acronyms

AGV Automated Guided Vehicle

AMCL Augmented Monte Carlo Localization

FALKO Fast Adaptive Laser Keypoint Orientation-invariant

FLIRT Fast Laser Interest Region Transform

FT Feature Tracking

ICL Iterative Closest Line

ICP Iterative Closest Point

ISR Institute of Systems and Robotics

LIDAR Laser Interferometry Detection and Ranging

MCL Monte Carlo Localization

PF Particle Filter

PSM Polar Scan Matching

RANSAC RANdom SAmple Consensus

RMSE Root-Mean-Square Error

ROS Robot Operating System

SD Standart Deviation

SLAM Simultaneous Localization and Mapping

List of Figures

List of Tables

Chapter 1

Introduction

Robotic systems are now widely used for industrial purposes. Although they were initially designed to perform dangerous or unsuitable duties for human workers, nowadays they are used to perform large scale repetitive actions, such as welding, packing or moving materials. Material transportation in a large scale industry can either be done by having an automatic warehouse system or by using automatic guided vehicles (AGVs). Regarding AGVs localization systems, there are many solutions based on different sensors, where the most deployed solutions are based on magnetic tape tracking or laser triangulation using artificial landmarks (beacons).

Magnetic tape tracking consists in using a sensor on the robot to follow a line on the floor (Figure 1.1a) and is one of the most used navigation systems in industry. The large use of magnetic-based guidance comes from its simplicity and low price. However, this technology is not flexible, since a modification of the robot's path demands a modification on the floor tapes topology as well. For this reason, this approach does not adapt well when frequent changes occur in the environment's layout. This flexibility problem can be solved with the use of laser triangulation. In this case, a laser is used to detect artificial landmarks spread around the environment (Fig. 1.1b) and estimate a robust and precise robot's localization. This comes with an increased cost from the laser used and the installation of the artificial landmarks. Alongside with these solutions, it is usually installed a security laser, to avoid the collision with objects that cross the robot's pathway. This dissertation reports the main results on the development of a laser-based localization system for an AGV, as it will be described shortly.

(a) Magnetic tape tracking.[1] (b) Laser triangulation.[2]

Figure 1.1: Two different approaches for AGVs localization system.

A robot localization system is designed to give a reliable and robust estimation of its position and orientation, using data gathered from its sensors. This localization is very important when used on industrial environments, because there is a high demand for quality and most solutions are still very conservative. This happens because a failure in localization can be extremely problematic, as it not only can break the production levels of an industry, as can also compromise human safety, even with the use of a safety laser. The main goal of this dissertation is to develop a robust localization system based only on laser scan data, using natural landmarks instead of the application of artificial ones. A first algorithm will be developed to extract natural features from the environment (corners) and estimate the robot's pose using a scan matching method (Iterative Closest Point). Afterwards, this algorithm will be evaluated on different scenarios to test its robustness. Lastly, it will be implemented along side with the HectorSLAM with the purpose of trying to improve its performance.

This chapter is an introductory chapter, presenting the context and motivation in which this dissertation was done, the main goals of it and a brief overview on how it is organized and what should be expected from the other chapters.

[1] http://www.thefabricator.com/product/materialshandling/agv-designed-for-tight-turns-narrow-aisle-material-handling

[2] http://uk.rs-online.com/web/generalDisplay.html?id=infozone&file=automation/safety-agvs

1.1 Context and Motivation

The use of "intelligent" robots has been increasing over the past years, either for industrial purposes or not. With the increasing application of robots in highly complex contexts, it is crucial that their navigation systems are as reliable and robust as possible. There has been a great development over the years to turn AGVs localization methods more flexible without increasing their pose estimation error. Laser-based localization is a potential approach for industrial AGVs applications.

There has been a great evolution of localization methods, most of them relying on multimodal sensory data. Usually, a navigation system receives data from many sensors, such as laser scanners, wheel encoders (used for odometry) and/or inertial sensors (IMU). It is an interesting challenge to develop a localization system based only on laser data due to the fact that it does not require additional sensors installed on the robot's platform and does not either require any materials applied to the environment, such as beacons or magnetic tape. This work is inserted in the project P2020 AGVPOSYS (*Automated-Guided-Vehicle with innovative indoor positioning system for the factory of the future*), in co-promotion between Active Space Technologies and University of Coimbra, which mainly consists in developing an AGV localization system using laser-based techniques.

1.2 Objectives

As stated before, most of localization systems use data from many sensors, such as laser range-finder, wheels' encoders or inertial sensors. The main purpose of this dissertation is to developed an algorithm which is able to perform a robust localization using only laser data. The first goal of this work is to evaluate its performance amongst other algorithms already validated in literature. After that, the developed method will be used along side with an available SLAM algorithm (HectorSLAM) with the purpose of trying to give a contribution to this algorithm.

Summing up, the main goals of this dissertation:

- The first objective is to develop a feature (corners) extraction algorithm based on laser scan data gathered to build a map, using line segmentation and intersection methods;

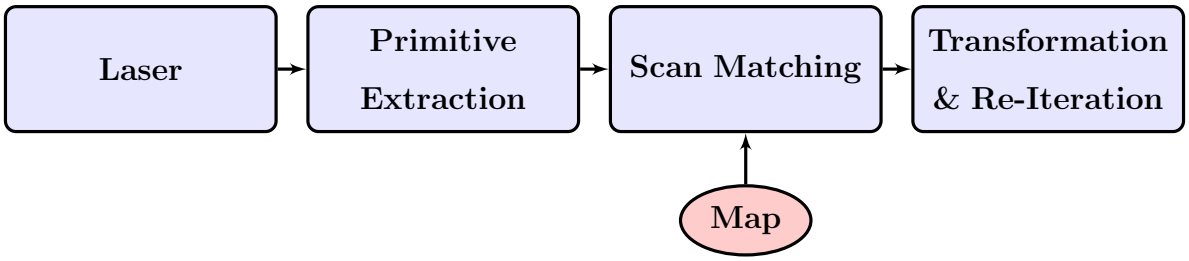


Figure 1.2: Generic pipeline for a localization method based on scan matching.

- After that, pose estimation is done using a scan matching algorithm (ICP, Iterative Closest Point), using the previous map and a known feature map from the environment;
- Finally, the algorithm developed will be used alongside the HectorSLAM in order to evaluate if it is able to enhance the SLAM method.

Before starting to analyse the state of art of this dissertation and also to better understand the outline of it, Fig. 1.2 presents the generic pipeline of a localization algorithm based on scan matching, which is the core of this dissertation. As it can be seen, the first two main goals of this dissertation are focused on the second and third block of this diagram, respectively. It will now be seen where each of this diagram blocks fit in the outline of the dissertation.

1.3 Outline of the dissertation

Chapter 2 (State of Art)

This chapter gives a brief overview over the existing methods in literature concerning indoor localization. It is focused on methods that are used in the developed algorithm and on methods used in this dissertation to compare results with it. Two SLAM techniques openly available in ROS (Robot Operating System), GMapping and the HectorSLAM, are presented with the last one being more focused as it will be explored ahead on this work.

Chapter 3 (Developed Methods)

Chapter 4 presents the algorithms designed and developed on this dissertation. For each algorithm, a detailed description is given and samples of the developed

source code are also presented. It is divided in three main sections: segmentation and natural features (corners) extraction (second block of Fig. 1.2); scan matching with ICP algorithm (third and fourth block of Fig. 1.2); finally, a localization method consisting on the HectorSLAM approach integrating the corners extraction developed algorithm.

Chapter 4 (Experimental Results)

In this chapter, an analysis about the results obtained is done. Both results from developed algorithms and algorithms already available in literature are presented and compared. Different scenarios are tested to have a better evaluation of the developed algorithm's behaviour. Once this dissertation focuses on localization, most of the results presented concern the position error of the robot within different positions around the environment.

Chapter 5 (Conclusion and Future Work)

The main conclusions are drawn in this chapter are presented, both concerning the development of a feature extraction algorithm and its application in the framework of the HectorSLAM. Lastly, a few directions will be given for future research in order to improve the work developed in this dissertation.

Chapter 2

State of Art

This dissertation consists in two major parts: the first one focused on segmentation and primitive extraction, which enables the construction of a feature map based on extracted primitives (corners); the second one concerns on mobile robots localization, *i.e.* estimate the robot's pose using scan matching.

Since segmentation and primitive extraction are the major parts of this dissertation, it is important to present some of the most commonly used techniques. Section 2.1 describes the most used algorithms in this area.

It is also important to give a brief overview of some of the existing localization methods before discussing the solution proposed in this dissertation. The two major issues about localization are how to represent the sensory data gathered by the robot and how to estimate the robot's pose using that data and internal maps. Section 2.2 presents some of the most commonly used environment representations and Section 2.3 presents an overview of some localization methods. Section 2.4 gives a brief introduction to SLAM and presents two well-known SLAM approaches: Gmapping and the HectorSLAM.

2.1 Segmentation and Primitive Extraction

Segmentation is the first step after laser data acquisition and consists on identifying and separating sets of segments of the laser data extracted from the environment. Primitive extraction is used to provide more detailed information about the environment and its particularly useful to build a feature map of it. Both of these

techniques will now be briefly analysed with the purpose of giving a better context to the work developed in this dissertation.

Before starting to explore both segmentation and feature extraction methods, it is important to give a brief overview of the laser scan, in order to understand the nomenclature used ahead. A range scan is a list of points corresponding to the intersection of a laser beam with objects in the robot's environment within a constant angle increment. Considering a full 270º scan of n measurements, each laser scan point P is defined by a radial (r) and angular (θ) components. Each point i ($i = 1, ..., n$) is then defined by the polar coordinates (r_i, θ_i), where θ_i is usually calculated using the angular increment $\Delta\theta$ given by laser scan specifications. This polar coordinates can afterwards be converted into cartesian ones:

$$\begin{cases} x_i = r_i \cdot \cos(\theta_{min} + (i-1) \cdot \Delta\theta) \\ y_i = r_i \cdot \sin(\theta_{min} + (i-1) \cdot \Delta\theta) \end{cases} \tag{1}$$

2.1.1 Segmentation

Segmentation methods can be subdivided into two categories [28]: Point-distance-based (PDBS) and Kalman Filter-based (KFBS) segmentation methods. A method for segmentation based on Kalman Filter (KF) is described in [7] and will not be explored here since this type of method is not used during this dissertation.

Point-Distance-Based Methods (PDBS)

This type of methods use the euclidean distance between consecutive laser points and a distance threshold to form the condition to detect breakpoints. If $D(r_i, r_{i+1}) < D_{thd}$ then points are in the same segment, else the segment is separated, where

$$D(r_i, r_{i+1}) = \sqrt{r_i^2 + r_{i+1}^2 - 2 r_i r_{i+1} cos\Delta\theta}$$

As for the distance threshold D_{thd}, different approaches can be done, as described in [28]. One of the most known approaches [11], uses

$$D_{thd} = C_0 + C_1 \min\{r_i, r_{i+1}\}$$

where $C_1 = \sqrt{2(1 - cos\Delta\theta)} = D(r_i, r_{i+1})/r_i$ and C_0 is a constant parameter used for noise reduction. Based on this approach, [33] introduces a new parameter β with

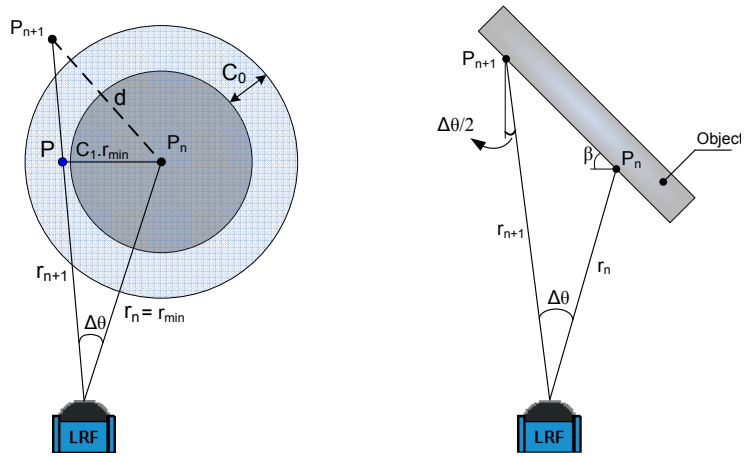


Figure 2.1: Geometric representation of both PDBS approaches to find the value of D_{thd}, described below [27].

the purpose of removing the dependence of segmentation from the distance between LRF and the objects. The threshold is then given by:

$$D_{thd} = C_0 + \frac{C_1 \cdot \min\{r_i, r_{i+1}\}}{\cot(\beta) \cdot (\cos(\Delta\theta/2)) - \sin(\Delta\theta/2)}$$

Figure 2.1 shows a geometric representation with these variables for each of the methods describes above.

Another method widely used for segmentation (and also line extraction) is the Iterative End-Point Fit (IEPF) [14]. This is a recursive algorithm that starts by defining a line between the first and last points of a set of points. After that, the point with the maximum distance to that line is detected and if that distance is bigger than a threshold ϵ, then the segment is divided into two segments. This algorithm, also known as Ramer-Douglas-Peucker algorithm [30] [13], is widely used in robotics to segment laser range data and will be explored later on this dissertation.

2.1.2 Geometrical Primitive Extraction

As pointed in [28], geometrical primitive extraction can be divided into two techniques: *clustering* (e.g. Hough-based approaches) and *least-squares* methods. Since this dissertation only uses the second one, only that type of methods will be analysed. In this subsection, it will also be given a brief overview of two interest point extractors: *Fast Laser Interest Region Transform* (FLIRT) and *Fast Adaptive Laser Keypoint Orientation-invariant* (FALKO), both based on primitive extraction and

which is used in later sections in this dissertation.

Primitives extracted can either be lines, circles or ellipses, which can be defined by the following equation:

$$ax^2 + bxy + cy^2 + 2dx + 2ey + f = 0 \ , \tag{2}$$

where a and c are different from zero. From this equation, the geometrical primitives can be defined:

- Line: $a = b = c = 0$;
- Circle: $a = c$ and $b = 0$;
- Ellipse: $(b^2 - ac) < 0$.

In this dissertation only line extraction is considered since most of the experimental test rooms used can be uniquely represented by lines, and for that reason this is the type of extraction that will be analysed now.

Linear Regression Method

From equation (2), a specific line equation can be defined as $2dx + 2ey + f = 0$, or in a more generic form:

$$y = mx + b$$

One of the most known methods for line extraction is the linear regression method (LRM). Following the notation in [40], line parameters m and b can be obtained from:

$$m = \frac{S_{xy} \cdot n - S_x S_y}{S_{xx} \cdot n - S_{xx}^2} \text{ and } b = \frac{S_y - m.S_x}{n}$$

where, considering a set of points (1), the regression parameters can be defined as follows:

$$S_x = \sum_{i=1}^{n} x_i, \quad S_y = \sum_{i=1}^{n} y_i, \quad S_{xx} = \sum_{i=1}^{n} x_i^2, \quad S_{yy} = \sum_{i=1}^{n} y_i^2, \quad S_{xy} = \sum_{i=1}^{n} x_i y_i$$

Even though there are many feature extraction techniques in literature [22], this dissertation focused on two feature extraction methods openly available in OpenSLAM [35]. Since they were used as comparison to the developed algorithms, they will now be analysed in more detail in order to understand its utilization later.

Fast Laser Interest Region Transform

In [39], Tipaldi and Arras propose a fast interest region extractor for 2D range data. FLIRT library [35] has four different multi-scale detectors: one range-based detector, two normal-based detectors and one curvature-based detector, which are going to be briefly described.

The range-based detector finds interest points using a blob detector on the raw range data gathered by the 2D laser scanner. A blob is a region of 2D laser data in which some properties are approximately constant. This detector was inspired by the SIFT features proposed in [23].

Both normal-based detectors use a local approximation of the normal direction at each laser point, instead of the raw range data, which results in what can be defined as *normal signal*. The normal direction at each point is estimated by least squares fitting of a group of measurements around that point. These normal-based detectors distinguish themselves with the fact that one responds to edges in the *normal signal* and the other responds to blobs, in a similar way as the range-based detector.

Finally, the curvature-based detector which uses the range data from laser scan to define a curve in Cartesian space. This detector does not fit very well in an environment with planar geometric features (absence of curves) and so it is not useful in office-like indoor environments.

Fast Adaptive Laser Keypoint Orientation-invariant

Two keypoint extractors from laser scan data for robot localization and mapping are introduced in [18]: FALKO and OC (*Orthogonal Corner*). These two algorithms were developed to detect interest points like those defining corners. While FALKO uses a neighbourhood region and an efficient corner extraction algorithm, OC exploits the orthogonal alignment of points.

In case of office-like environments composed by walls and objects, *Orthogonal Corner Detector* obtains keypoints from the intersection of orthogonal line pairs. This algorithm is based on *Hough Transform*. It finds a dominant direction $\bar{\theta}$ using an *Orthogonal Hough Spectrum* [18] and once it is found, all the scan points are rotated by $-\bar{\theta}$ and tested as candidate keypoints. New sets of points are calculated based on these rotated points and corners are then defined using those sets.

Similar to OC keypoints, in FALKO a set of candidate keypoints is also defined. This set of points is divided in two subsets and these subsets are analysed in order to reduce candidate points. After that, a quantized orientation is calculated for each point of those subsets and a distance function between this quantized orientations is obtained. Corners are then found based on that distance function.

2.2 Map Representations

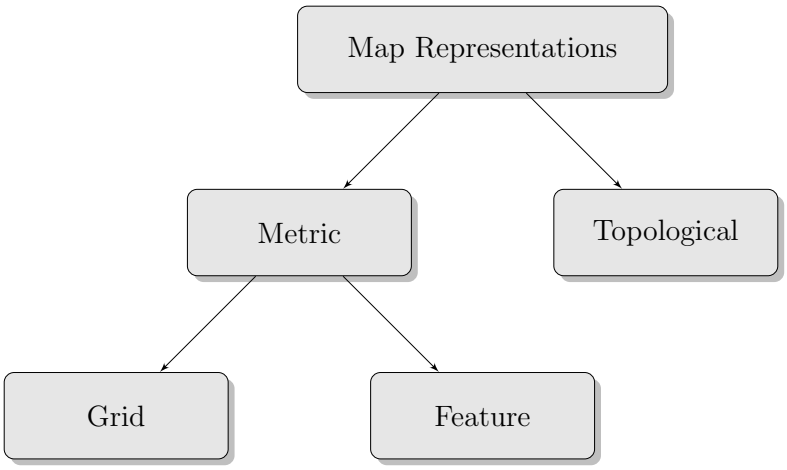
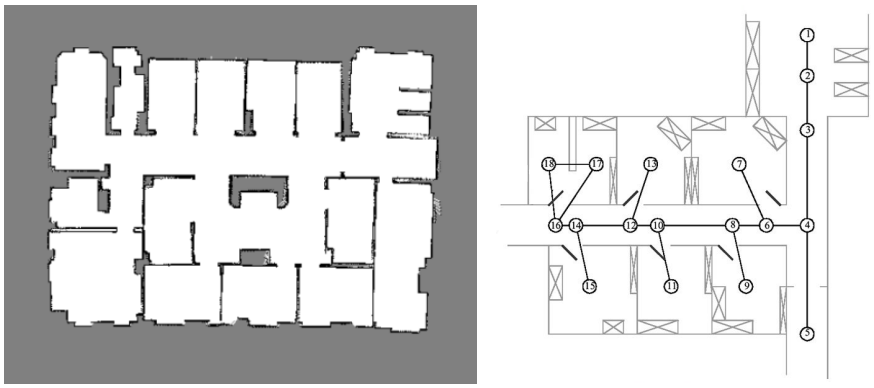


Figure 2.2: Most commonly used map representations.

There are many ways to represent the information we have about the environment. This section focuses on two distinct map representations [37]: topological and metric, which can be distinguished in grid-based and feature-based maps (Fig. 2.2).

In feature-based maps, the world is represented by geometric features, such as points and lines. Instead of building a map based on features, grid maps (Fig. 2.3a) represent environments by evenly-spaced cells, *i.e.* the world is divided into a grid. Each cell characterizes a small area of the world and contains the probability of that area being occupied. Topological approaches represent robot environments by graphs. Nodes in the graph correspond to places of importance, such as distinct places or landmarks. The nodes are connected by edges if there is a direct connection between them, as shown in Fig. 2.3b.

Occupancy grids are the most used type of metric maps, in particular 2D occupancy grids. In these maps, the environment is represented by a matrix of cells, each cell with a value attached that represents the belief of that cell being occupied. Not



(a) Occupancy grid.[1] (b) Topological map.[2]

Figure 2.3: Occupancy grid and topological map representations.

only occupied, but unknown and free space are also represented. Since the geometry of a grid corresponds directly to the geometry of the environment, metric maps are usually easy to build and maintain even in large environments. However, due to the fact that, in most cases, the resolution of the grid has to be fine enough to get all the details, these maps may require large amounts of memory and computation time.

Unlike grid-based approaches, topological maps work better in large scale environments, where there are more landmarks. The compactness of topological representations allows fast trajectory planning and these approaches usually recover better from errors than grid-based maps because these maps require a constant monitorization and compensation. However, as stated in [37], topological approaches may have difficulties in distinguishing places that look alike because the model is based on landmarks or distinct sensory features. Both representations have strengths and weaknesses, which are summarized in Table 2.1.

2.3 Localization: Scan matching algorithms

Localization is one of the biggest problems in robotics, because nearly all robotic tasks require the robot to know its position and the position of the objects that it manipulates. Mobile robot localization consists in determinating the pose of a robot relative to a given map of the environment, usually known as the ground truth map.

[1] `http://robotang.co.nz/projects/robotics/custom-player-plugins/`

[2] `http://slideplayer.com/slide/4975998/`

	Metric	Topological
Advantages	Easy to build, represent and maintain	Do not require exact robot's position
	Recognition of places is non-ambiguous	Allow fast trajectory planning
	No explicit model needed to handle information	Less complex and more compact
Disadvantages	Memory problems (space-consuming)	Recognition of places often ambiguous
	Highly susceptible to odometry errors	Harder to maintain consistency in larger environments
	High computational cost	May fail to recognize geometrically nearby places

Table 2.1: Advantages and disadvantages of grid-based and topological approaches based on [37].

According to [38], localization can be seen as a problem of coordinate transformation. The map of the environment has a global coordinate system and the robot has its own local coordinate system. The purpose of localization is to establish a correspondence between both coordinate systems, allowing the robot to know its position in the environment. This process is easy if the robot's pose is known, which can be sensed directly from a pose measuring sensor, such as an inertial or odometric sensor. However, most robots do not possess this kind of sensor (at least not noise-free). Therefore, the pose has to be estimated from data gathered by other sensors, as stated in [38].

One possible way to do localization is to use laser scan matching [12]. In laser scan matching, the position and orientation of the robot are estimated based on the transformation between the current laser scan (or map) and a reference scan. Figure 2.4 shows the outline of a localization algorithm based on scan matching. Scan matching tries to find the best alignment between a current laser scan k and a previous scan $k-1$ (or a known reference map of the environment). Because the laser data is expressed in polar coordinates, a preprocessing module has to convert these coordinates into cartesian coordinates, resulting in a 2D point cloud. The result of scan matching will be a transformation composed by a rotation matrix R and a translation vector t, which can be applied to update the current robot's pose estimation. This transformation is also be applied in a post-processing module to rectify the laser scan so this can be used as an input for the scan matching algorithm.

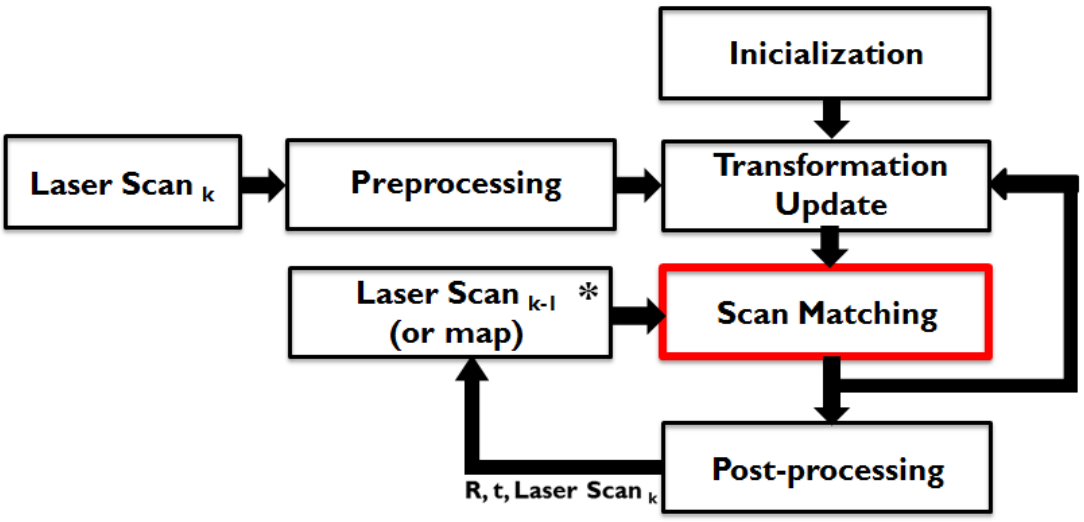


Figure 2.4: Outline of a localization algorithm based on scan matching (* laser scan $k-1$ already has its coordinates converted).

Even though this process might seem easy at first, aligning two scans perfectly to find the exact robot's pose might be impossible. According to [24], there exist two types of problems between laser scans. The first one concerns the existence of small differences between the laser scan and the real environment, which result from random sensing noise and can produce a cumulative error through the robot's movement. The second type of difference is caused by occlusion, *i.e.* areas seen in some scans might not be seen in others.

There are many scan matching algorithms in the literature [26] [20]. One of the most applied algorithms is the *Iterative Closest Point* (ICP), initially proposed in [6], in which two point clouds are matched based on the Euclidean distance between each point of those clouds. Many approaches derived from ICP, such as *Iterative Closest Line* (ICL) proposed in [10]. This algorithm matches each point to the two closest points in the reference point cloud and uses the line defined by those points to calculate the transformation between the two point clouds. In [24], a variant of ICP algorithm is proposed. In addition to choosing the closest point on the current scan [6], an additional point with the same distance from the origin as the reference point matched is also chosen, allowing a more accurate estimation of rotation and translation between scans. The big disadvantage of all these methods is the high computational cost which results from the search of point correspondences done

in each iteration. The *Polar Scan Matching* (PSM) approach introduced in [12] uses the laser scanner's polar coordinate system to avoid an expensive search for corresponding points.

2.3.1 Iterative Closest Point

Iterative Closest Point algorithm is an algorithm implemented to minimize the euclidean distance between two point clouds. Following [6] notation, a data shape P is registered (moved) to be in best alignment with a model shape X. In this dissertation, the data shape P will be the data gathered by laser range-finder and the model shape X will be the known map of the environment.

As stated before, there are many variants of the ICP algorithm. One of the most robust registration methods is weighting of point-pairs [31]. In this method, matched points are assigned with a weight based on the distance between them, *i.e.* lower weights are assigned to pairs with greater point-to-point distance. In [4] a re-weighted least squares method is proposed, in which the weighting of point-pairs is obtained from a M-estimation criterion. This was one of the algorithms used in this dissertation and so a more detailed overview will now be given.

The algorithm starts with two sets of points: a set of n data points $\{p_i\}_{i=1}^{n}$ relative to the laser scan, that will be designated by P and a set of m model points relative to the reference map, designated by X, both $\in \mathbb{R}^2$. The distance metric d between an individual data point p and a model shape X will be denoted as

$$d(p, X) = \min_{y \in X} ||p - y|| \ .$$

A robust M-estimate of the rigid body transformation between the two sets of points using a robust criterion function $\varrho : \mathbb{R} \to [0, \infty[$ is obtained by solving

$$\min_{R,t} f(R, t) \ ,$$

where

$$f(R, t) = \sum_{i=1}^{n} \varrho(d(Rp_i + t, X)) \ ,$$

and R is the rotation matrix and t the translation vector that form the rigid body transformation between data point set P and model point set X.

The purpose of a robust criterion function ϱ is to reduce undesired influence of data with significant errors on the estimation. Three different criterion functions are used in [4]: Huber's function ϱ_{Hu}, Cauchy's function ϱ_{Ca} and Tukey's bi-weight function ϱ_{Tu}. Even though they are all robust, these criterion functions have different characteristics and their use might depend on the characteristics of the point clouds used. Tukey's bi-weight criterion offers the best protection against undesired data (outliers). Cauchy's criterion has also a good protection against the influence of this points. Huber's criterion does not completely ignore outliers, but it is still a robust option. These three criterion functions are the following:

$$\text{Huber's criterion function: } \varrho_{Hu}(r) = \begin{cases} \frac{1}{2}r^2 & \text{if } |r| \leq k_{Hu} \\ k_{Hu}|r| - \frac{k_{Hu}^2}{2} & \text{if } |r| > k_{Hu} \end{cases}$$

$$\text{Cauchy's criterion function: } \varrho_{Ca}(r) = \frac{k_{Ca}^2}{2} \log \left[1 + \left(\frac{r}{k_{Ca}}) \right)^2 \right]$$

$$\text{Tukey's bi-weight criterion function: } \varrho_{Tu}(r) = \begin{cases} \frac{k_{Tu}^2}{6} \left[1 - \left(1 - \frac{r^2}{k_{Tu}^2} \right)^3 \right] & \text{if } |r| \leq k_{Tu} \\ k_{Hu}|r| - \frac{k_{Hu}^2}{2} & \text{if } |r| > k_{Tu} \end{cases}$$

where r is the distance between a an individual point p and a model shape X ($d(p, X)$) and the parameter k is chosen so that the asymptotic variance

$$V(\psi, F) = \frac{\int \psi^2 dF}{(\int \psi' dF)^2}$$

is close to one, given an assumption of normal distribution $\mathcal{N}(0, 1)$ for r and a probability density function F of the normal distribution $\mathcal{N}(0, 1)$ [5].

A new criterion function was introduced in [5], the Welsch's criterion function. This function has a better protection against outliers than Cauchy's function and can be defined as:

$$\text{Welsch's criterion function: } \varrho_{We}(r) = \frac{k_{We}^2}{2} \log \left[1 + \left(\frac{r^2}{k_{We}^2}) \right) \right]$$

Defining $\Psi(r)$ as the derivative of $\varrho(r)$, $\Psi(r) = \varrho'(r)$, a weight function w is

defined as

$$w(r) = \begin{cases} \dfrac{\Psi(r)}{r} & \text{if } r \neq 0 \\ \lim\limits_{r \to 0} \dfrac{\Psi(r)}{r} & \text{if } r = 0 \end{cases}$$

It is now possible to define the weight functions of the criterion functions:

Huber's weight function: $w_{Hu}(r) = \begin{cases} 1 & \text{if } |r| \leq k_{Hu} \\ \dfrac{k_{Hu}}{|r|} & \text{if } |r| > k_{Hu} \end{cases}$

Cauchy's weight function: $w_{Ca}(r) = \dfrac{1}{1 + \left(\dfrac{r}{k_{Ca}}\right)^2}$

Tukey's bi-weight weight function: $w_{Tu}(r) = \begin{cases} \left(1 - \dfrac{r^2}{k_{Tu}^2}\right)^2 & \text{if } |r| \leq k_{Tu} \\ 0 & \text{if } |r| > k_{Tu} \end{cases}$

Welsch's weight function: $w_{We}(r) = \exp\left(-\dfrac{r^2}{k_{We}^2}\right)$

From the weight functions and the point clouds provided to this algorithm, it is possible to calculate the rotation matrix and translation vector using a singular value decomposition. First, a variable C is defined as

$$\mathrm{C} = \frac{1}{\hat{w}} \sum_{i=1}^{n} \left[w_i p_i y_i^T \right] - \bar{p}\bar{y}^T \in \mathbb{R}^2 \ ,$$

where

$$\hat{w} = \sum_{i=1}^{n} w_i \ , \quad \bar{p} = \frac{1}{\hat{w}} \sum_{i=1}^{n} w_i p_i \ , \quad \bar{y} = \frac{1}{\hat{w}} \sum_{i=1}^{n} w_i y_i$$

The rotation matrix R is then found using a SVD decomposition of C, *i.e.* C=UΣV^T. From this decomposition, R can be obtained from R =VU^T (or R =-VU^T if det(VU^T <0)). Once the rotation matrix R has been obtained, the translation vector t is easily calculated from $t = \bar{y} - R \cdot \bar{p}$. The transformation obtained is then updated through ICP iterations.

2.4 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is the process by which a mobile robot can build and update a map of the environment and, simultaneously, use the generated map to compute its location [2] [9]. A pioneer approach of SLAM [21] uses an Extended Kalman Filter (EKF) to match *geometric beacons* (natural environment features that can be successively observed) with a known map, which originated the first SLAM algorithm EKF-SLAM. In [25] a FastSLAM algorithm is introduced. This algorithm applied a particle filter fusion approach aiming to provide faster and more accurate results in comparison with previous SLAM algorithms.

There are many SLAM techniques openly available in ROS. An evaluation of the most commonly used techniques was done in [32], where the HectorSLAM and GMapping have shown good experimental performance. This dissertation focuses on the HectorSLAM, since we are primarily interested in a solution using only laser range data. However, a brief description of Gmapping is also done in order to make this dissertation more complete.

2.4.1 Gmapping

Gmapping is a laser-based SLAM algorithm as described in [16]. This algorithm is based on an improved Rao-Blackwellized particle filter and an adaptive re-sampling. A normal particle filter increases its complexity as the environment increases as well because more particles are needed to estimate the robot's localization. The key idea behind Rao-Blackwellized approach is to marginalize out a subset of the state space, which can be handled in a more efficient way by using a Gaussian representation.

Let x_t be the robot's position at time t, z_t the observation measurements and u_t the odometry measurements obtained by the mobile robot. Using this notation, $p(z_t|x_t, m)$ represents the probability of a particular measurement if the robot's pose and the environment map m are known. As stated in [16] "*for each particle i the parameters $\mu_t^{(i)}$ and $\Sigma_t^{(i)}$ are determined individually for K sampled points $\{x_j\}$ in a interval $L^{(i)}$*". The Gaussian parameters are estimated as follows [16]:

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} x_j \cdot p(z_t|m_{t-1}^{(i)}, x_j) \cdot p(x_j|x_{t-1}^{(i)}, u_{t-1})$$

$$\Sigma_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} p(z_t|m_{t-1}^{(i)}, x_j) \cdot p(x_j|x_{t-1}^{(i)}, u_{t-1}) \cdot (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T$$

where K is the number of sampled points and $\eta^{(i)}$ is a normalization factor. Using this distribution, the importance weights are computed as follows:

$$w_t^{(i)} = w_{t-1}^{(i)} \cdot \sum_{j=1}^{K} p(z_t|m_{t-1}^{(i)}, x_j) \cdot p(x_j|x_{t-1}^{(i)}, u_{t-1}) = w_{t-1}^{(i)} \cdot \eta^{(i)}$$

An important aspect of a particle filter is the re-sampling process, in which low weight samples are usually replaced by samples having a high weight. Because re-sampling is a critical step, it is important to find a good criterion for deciding when to perform the re-sampling step. For details see Algorithm 1 - Improved RBPF for Map Learning [16].

2.4.2 HectorSLAM

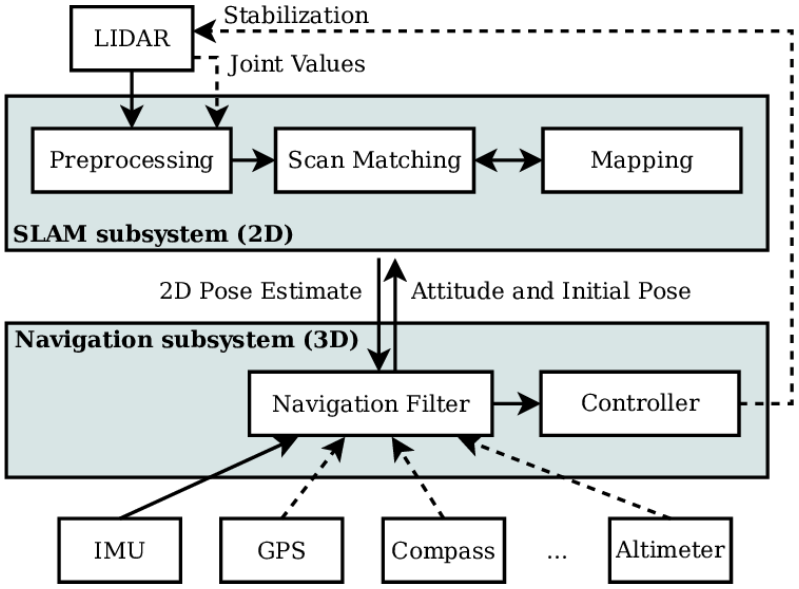


Figure 2.5: HectorSLAM system overview (from [19]).

The other main SLAM algorithm available in ROS is the HectorSLAM [19]. This SLAM method combines a 2D SLAM system based on scan matching with a 3D navigation system based on an inertial measurement unit (Fig. 2.5). This SLAM algorithm does not require wheels encoders which allows a larger use of it and a good solution for systems that do not use odometric information.

As it can be seen in Fig. 2.5, the 2D SLAM system is based on scan matching. As described in [19], the purpose of this scan matching is to find the rigid transformation $\xi = (p_x, p_y, \psi)^T$ that minimizes

$$\xi^* = \underset{\xi}{argmin} \sum_{i=1}^{n} \left[1 - M(S_i(\xi))\right]^2$$

where $S_i(\xi)$ are the coordinates of scan endpoints $s_i = (s_{i,x}, s_{i,y})^T$ and $M(S_i(\xi))$ returns the map value at the coordinates given by $S_i(\xi)$. To solve this minimization problem, the following Gauss-Newton equation is used:

$$\Delta\xi = H^{-1} \sum_{i=1}^{n} \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[1 - M(S_i(\xi))\right]$$

with

$$H = \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}\right]$$

Gradient based approaches have the risk of getting stuck in local minima. To avoid these situations, a multi-resolution map representation is used in the HectorSLAM [19]. Different maps are kept in memory and simultaneously updated using the pose estimates generated by the alignment process. This procedure ensures that the map is always consistent and the computational effort remains low.

Chapter 3

Developed Methods

For a first localization approach, the Iterative Correspondence Point (ICP) algorithm was used. The implementation of this algorithm in this work consists in four main stages: first, laser scan data is gathered and preprocessed; after that, a primitive feature extraction algorithm is used to extract corners; these corners are used along the environment measured corners in the ICP algorithm; finally, transformation matrix returned by ICP is used to estimate the robot's position over time. Fig. 3.1 represents the flow of the process. All the algorithms presented were implemented and tested in Matlab and will now be analysed.

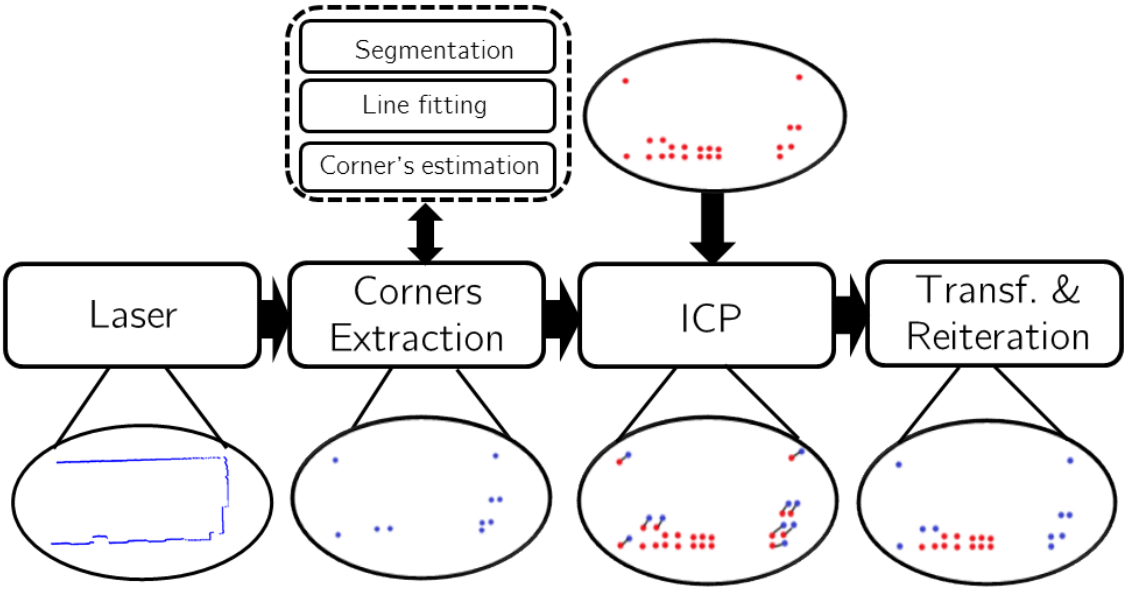


Figure 3.1: Pipeline of the developed algorithm based on scan matching with ICP.

3.1 Segmentation and Primitive Extraction

In order to extract primitives, *i.e.* corners and lines, laser scan data has to first be segmented. After that, a line fitting algorithm can be applied to turn segments into lines and, at last, corners can be extracted from the intersection between those lines.

A first approach for segmentation was implemented using a Point-Distance-based method (PDBS), as described in [28]. The Euclidean distance between two consecutive scanned points can be obtained from:

$$D(r_i, r_{i+1}) = \sqrt{r_i^2 + r_{i+1}^2 - 2r_i r_{i+1} cos\Delta\alpha}$$

where r_i and r_{i+1} are the laser measurements of frames i and $i+1$, respectively, and $\Delta\alpha$ is the sensor angular resolution. For each pair of consecutive scanned points, if $D(r_i, r_{i+1}) < D_{thd}$ then points i and $i+1$ are in the same segment, else they are in separate segments. D_{thd} is the threshold condition, which is sensitive to the test scenario. Algorithm 1 summarizes how segmentation was implemented.

Algorithm 1: Algorithm for laser scan data segmentation

```
m: number of laser scan frames
n: number of readings in each frame
for i = 1 to m do
    for j = 1 to n − 1 do
        D(j) = sqrt(r_j^2 + r_{j+1}^2 − 2 r_j r_{j+1} cosΔα)
    end
    Segment k ← x(i, j), y(i, j)
    if D(j) > D_thd then
        k ← k + 1
    else
        k ← k
    end
end
```

As it will be seen ahead, using Algorithm 1 by itself, does not give the desired results because it works based on the distance between two consecutive points of the laser scan data. This happens due to the fact that some segments may contain interest points like corners which will result in a poor corner extraction after a line

fitting algorithm is applied. To overcome this problem, the Ramer-Douglas-Peucker algorithm [30] [13] was used on these segments (containing interest points) with the purpose of breaking them and improving the result of a line fitting algorithm and therefore the result of corner extraction.

The Ramer-Douglas-Peucker, or just Douglas-Peucker (DP) algorithm, is an algorithm for reducing the number of points in a curve that is approximated by a series of points. As already stated, some segments that result from the initial segmentation algorithm (Algorithm 1) can still be split. For the purpose of this work, the Douglas-Peucker algorithm was initially used to split those segments, as shown in Fig. 3.2.

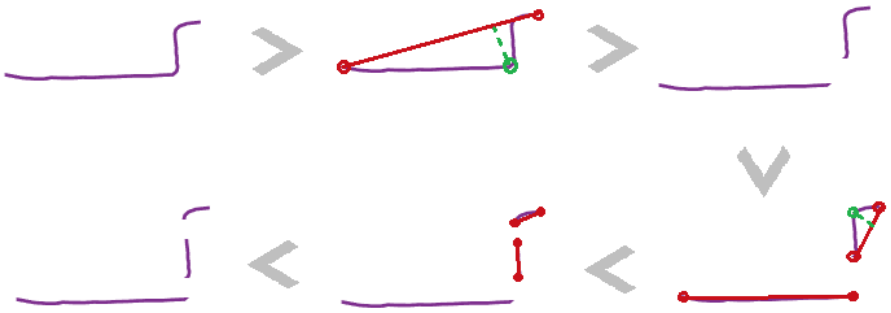


Figure 3.2: Splitting a segment using Douglas-Peucker algorithm.

Douglas-Peucker algorithm takes on a list of points (segment) and a user-defined parameter ϵ. As shown in Fig. 3.2, a line is drawn between the first and last point of the segment and then the perpendicular distance between each point and the line is calculated from

$$d(P_0, P_1, P_2) = \frac{|(y_2 - y_1) \cdot x_0 - (x_2 - x_1) \cdot y_0 + x_2 \cdot y_1 - y_2 \cdot x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}} \ ,$$

where $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ are the points that form the line and $P_0(x_0, y_0)$ is the point used to calculate the distance to the line. If that distance is greater than threshold ϵ, the original segment is split into two. The algorithm is recursively applied. Algorithm 2 shows how Douglas-Peucker was implemented. The input of this algorithm is a $3 \times n$ list of points, where the first two rows of this list are, respectively, the x and y coordinates and the third row is the index of those coordinates.

After segmentation is done, a line fitting algorithm was implemented. The main purpose of this procedure was to convert segments into lines so that corners could

Algorithm 2: Ramer-Douglas-Peucker algorithm used for line segmentation

Data: List of $3 \times n$ Points & regulation parameter ϵ

```
for i = 2 to n − 1 do
    d = perpendicularDistance(Point i, Line(Point 1, Point n)) // Perpendicular
        distance between Point i and line formed by first and last points of
        input
    if d > d_max then
        idx = i ;                                              // save point index
        d_max = d ;                                      // save new max distance
    end
    // If max distance is greater than epsilon, recursively simplify
    if d_max > ϵ then
        recursiveResult1 = Ramer-Douglas-Peucker(Points 1 to idx, ϵ)
        recursiveResult2 = Ramer-Douglas-Peucker(Points idx to n, ϵ)
        Result = [recursiveResult1(1 to length(recursiveResult1)-1),
        recursiveResult2(1 to length(recursiveResult2)-1)
    else
        Result = [Point 1, Point n]
    end
end
```

be extracted from the intersection of those lines. A Least Square Fitting algorithm was used for it.

The main problem of a simple Least Square Fitting algorithm is the fact that it does not work properly with vertical lines. As it can be seen from the previous equations, the calculation of the slope largely depends on the variation of x. For vertical lines the results are not good because there is not much x variation. To solve this problem, a slope inversion can be done, as shown in Algorithm 3.

As a comparative algorithm, it was also used a robust RANSAC fitting [15], implemented in *Matlab* by Yan Ke (Algorithm 4). RANSAC is an iterative algorithm which is commonly used on point sets containing outliers. It fits a new line in each iteration using two random points of the segment and then saves the line's parameters if the line contains a minimum number of inliers. The more iterations, the more probable it is to find a better line (robust, with less outliers). However, increasing the number of iterations comes with an additional computational effort,

Algorithm 3: Least Square Fitting algorithm approach for vertical lines

Data: List of $(x_1, y_1), ..., (x_n, y_n)$ points corresponding to a segment

```
// Calculation of initial slope
```

$m_i = \frac{S_{xy} \cdot n - S_x S_y}{S_{xx} \cdot n - S_{xx}^2}$;

```
// Check if it is a vertical line
```

if *StandartDeviation(S_x)>StandartDeviation(S_y)* **then**

  $m = 1/m_i$; `// Inversion of slope`

else

  $m = m_i$;

end

```
// Calculation of line intercept
```

$q = \frac{S_y - m \cdot S_x}{n}$

which may cause high computational times for bigger scenarios, as it will be discussed after.

Another regression method was used with the purpose of achieving better regression results. Theil-Sen estimator, developed by Theil [36] and Sen [34], is a robust regression method that is almost insensitive to outliers. This method determinates the slope m of the line, fitting a set of points, by calculating the median of the slopes of all possible lines that can be generated by all pairs of points.

All the above mentioned regression methods (Least-Square Fitting, RANSAC and Theil-Sen estimator) have shown similar results when used in our experiments.

After obtaining regression lines, it is possible to predict corners based on the intersection of these lines. To find corner candidates, consecutive lines must fulfil two conditions: the euclidean distance between two consecutive points of two distinct consecutive segments (d_{seg}) must be less than a certain threshold (defined with a value of 0.1); and the angle between lines has to be greater than a defined threshold angle (for most tests, a angle of 45º was used). For most of the tests performed, the distance condition d_{seg} was not necessary, since the corners only existed between consecutive segments. However, in more complex scenarios, corners might not only come from the intersection of consecutive segments. Those scenarios will probably require the intersection between all segments and, for that reason, a distance condition is mandatory.

Before doing scan matching with the results obtained from this algorithms, two

Algorithm 4: Random Sample Consensus to fit a line (RANSAC Algorithm)

Data: List of $(x_1, y_1), ..., (x_n, y_n)$ points corresponding to a segment, an inlier distance threshold $thInlr$ and number of iterations $iterNum$

```
for i=1 to iterNum do
    // 1.  Fit using 2 random points
    pointSample = Select2RandomPoints(Points);
    line = GenerateLine(pointSample);
    // 2.  Count the inliers
    inliers = ComputeInliers(line,Points);
    numInliers = length(inliers);
    if numInliers > thInlr then
        Compute and save line parameters ρ1 and θ1;
        // Final fitted line ρ = sin(θ)·x + cos(θ)·y
    end
end
// 3.  Choose line with most inliers
[∼, index] = max(numInliers);
ρ = ρ1(index);
θ = θ1(index);
```

more algorithms were implemented (FLIRT and FALKO), to compare results with the ones developed. Both FLIRT and FALKO libraries (analysed before in Section 2.3) have their source code available, provided by OpenSLAM [35]. FLIRT library is composed of four different detectors and two descriptors. For the purpose of this work, two detectors were chosen according to how well they fitted in the laser scan data used. FALKO library was used specifically to extract corners from laser scan data and to compare them with the extracted corners from the developed algorithms . This library is also available in [35] and has two detectors, OC and FALKO, which have been previously discussed.

3.2 Scan Matching

To do scan matching, a reference set of points has to exist. This can either be gathered from laser scan data or directly measured from the environment. The reference map used during tests was built from tape measurements of the environ-

Algorithm 5: Intersection between lines and corner extraction

Data: Parameters m (slope) and b (intercept) of both lines

// Intersection point of lines l_1 and l_2

$x_{intercept} = \dfrac{b_2 - b_1}{m_1 - m_2}$;

$y_{intercept} = \dfrac{m_1 \cdot b_2 - m_2 \cdot b_1}{m_1 - m_2}$;

// Considering n segments of laser scan data

$j = 1$ // corner index

for i=2 **to** n **do**

  // Calculate angle and distance between segments

  $\theta_{intercept} = \left| \tan^{-1}\left(\dfrac{m(i) - m(i-1)}{1 + m(i) \cdot m(i-1)} \right) \right|$;

  $index = \text{length}(S_{i-1})$;

  $d_{seg} = \text{dist}\big[S_i(x_1, y_1), S_{i-1}(x_{index}, y_{index})\big]$);

  // Check if there might be a corner between segments

  if $\theta_{intercept} > 0.8 \ \wedge \ d_{seg} < 0.1$ **then**

    $\text{corners}(j)=(x_{intercept}, y_{intercept})$;

    $j = j + 1$;

  end

end

ment. For a scan matching algorithm to be effective, specially an ICP algorithm, the reference set of points has to be as closest to the real environment as possible.

To implement a localization method based on an ICP algorithm, two different approaches were made: a first approach used the corners previously detected to match with the corners measured from the scenario represented in Fig. 4.4; in a second approach, the full laser scan points were used to match with a set of linearly generated points between map corners. The main purpose of this second approach's implementation was to have a better evaluation of the first one. Because of that, this second implementation had to be as robust as possible so it could be used as a reference. Since all laser data is used, using a set of linearly generated points between map corners instead of just the map corners, allows the ICP algorithm to have more matching points and, consequently, better and more robust results.

The first approach followed was to use ICP algorithm with the previous extracted corners. These corners are used with the environment measured corners to find a rigid body transformation consisting in a rotation matrix $R \in \mathbb{R}^2$ and a translation

vector $t \in \mathbb{R}^2$ that fits a set of N points (extracted corners) $\{p_i\}_{i=1}^{N}$ to a set of model points X (measured corners). After that, a second approach was implemented, where all the laser scan data was used instead of only corners. The main purpose of this approach was to understand if using only corners would bring an advantage to this work. The algorithms of both approaches will now be analysed and the results discussed in Chapter 4.

Algorithm 6 resumes the implemented ICP algorithm, which uses corners as features. This algorithm is a re-weighted ICP approach where weights are estimated using criterion functions (see Section 2.3). The selected criterion function was the one that performed better in the given set of data points used in the algorithm.

Algorithm 6: Iterative Closest Point algorithm using corners extracted

Data: List of n data points D and n model points X

for *iter = 1* **to** *maxIter* **do**
  for $i = 1$ **to** n **do**
    for j=*1* **to** m **do**
      $d = dist(D(i) - M(j))$;
      if $d < d_{min}$ **then**
        $d_{min} = d$;
        $residuals(i) = d$;
      end
    end
  end
  $weights(1...n)$ =criterionFunction($residuals$);
  $sumWeights = \sum weights$;
  $D_w = \frac{(D \cdot weights)}{sumWeights}$; ; `// Apply weights to Data points`
  $X_w = \frac{(X \cdot weights)}{sumWeights}$;; `// Apply weights to Model points`
  $C = D \cdot weights \cdot X' - (sumWeights \cdot D_w) \cdot X'_w$;
  $[U, S, V] = \text{svd}(C)$; `// SVD decomposition`
  $R_i = V \cdot U'$;; `// Calculate rotation matrix`
  $T_i = D_w - R_i \cdot X_w$;; `// Calculate translation vector`
  $D = R_i \cdot D + T_i$; ; `// Apply transformation`
  `// Update transformation (TR and TT are the output tf)`
  TR = $R_i \cdot$ TR;
  TT = $R_i \cdot$TT + T_i;
end

3.3 Applying Corner Features in the HectorSLAM

In order to give a SLAM perspective to this dissertation, a SLAM method was integrated along side with the corner extraction algorithms explored. To do this integration, a robust SLAM method had to be used in order to achieve the best possible results. As stated before, in [32] an evaluation of some SLAM methods available in ROS is done, where the HectorSLAM and Gmapping show the most satisfactory results. The HectorSLAM was chosen due to the fact that, even though this method can incorporate data from many sensors, it is designed to be able to work efficiently with only laser data unlike Gmapping which needs odometric data as well.

The HectorSLAM, like any other SLAM method, has two main components: localization and mapping. Considering that this dissertation is focused on localization, some changes were made to this SLAM method in order to not use its mapping part and so work as a localization method. Another change regarding the use of a feature (corners) extraction algorithm was done in order to evaluate the SLAM method's behaviour when using features instead of the raw range data. Both changes will now be presented and explained.

3.3.1 Preprocessing

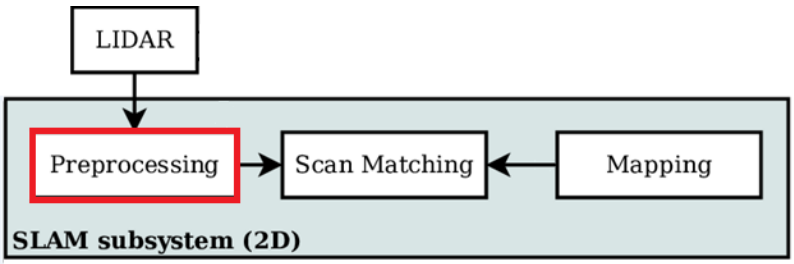


Figure 3.3: Main modules of the HectorSLAM: preprocessing module highlighted in red was modified to integrate corner features extracted from laser range data.

The HectorSLAM has a preprocessing module (Fig. 3.3), which is mainly used to convert the laser scan data into a point cloud of scan endpoints *i.e.* convert the polar coordinates from the laser scan data into cartesian coordinates. This dissertation addresses corner feature extraction methods and its application in the HectorSLAM's preprocessing unit.

In order to perform corner features extraction, FALKO's extractor was used. There were two main reasons behind the choice of FALKO over the developed algorithm: first, due to time constraints, since both the HectorSLAM and FALKO library were available in C++ code, unlike the developed algorithm which was implemented using Matlab; secondly, because the results obtained from both FALKO and the developed algorithm were very similar, as it will be seen in the next chapters.

The first approach done was to use FALKO to extract interest points. FALKO has also a preprocessing unit to convert polar coordinates into a 2D point cloud (cartesian coordinates), which is used in FALKO's keypoint extractor. This keypoint extractor was presented in Section 2.1.2 and is now summarized in Algorithm 7. Since sometimes there is not enough information on the extracted corners to perform a robust localization, a second approach was made using interest regions. The key idea behind interest regions is to use more information on each interest point extracted by using additional near points.

Algorithm 7: FALKO's keypoint extraction algorithm

Data: List of n laser scan points p_i

for $i = 0$ **to** n **do**
  `// Generate a neighbourhood region of oints` p_j `around` p_i
  neigh = getNeighPoints(p_i);
  for $j = 0$ **to** *length(neigh)* **do**
    `// Computate an orientation for each point` p_j
    $\phi_{j,L}$ =getPointOrientation[1]($p_{j,L}$);`// Left side of` p_i
    $\phi_{j,R}$ =getPointOrientation[1]($p_{j,R}$);`// Right side of` p_i
    `// Computate a score for each side of point` p_i
    $score_L(p_i)$ =getPointScore($\phi_{j,L}$);
    $score_R(p_i)$ =getPointScore($\phi_{j,R}$);
    $score(p_i) = score_L(p_i) + score_R(p_i)$;
  end
  `// Keypoints are then chosen as local minima of the score function with a non-maxima supression (NMS) procedure.`
end

[1]This function has a critical error which was corrected allowing better results in Matlab/MEX or ROS.

3.3.2 Mapping

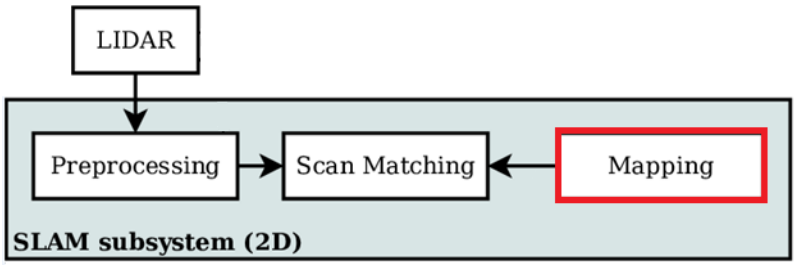


Figure 3.4: Main modules of the HectorSLAM: mapping module highlighted in red was modified to use an a priori map.

A SLAM method builds and updates a map of the environment and uses that map to perform localization simultaneously. As stated before in this work, there is no interest in the mapping feature of the HectorSLAM method and instead a reference map will be used. In order to understand how this reference map is applied in the implemented SLAM approach, it is important to first understand how the mapping process usually works in the HectorSLAM. After that, the modification done in the mapping unit (Fig. 3.4) will be presented and explained.

The HectorSLAM builds or updates the map every time the robot has an either linear or angular movement higher than a defined threshold. When this happens, the points scanned are mapped into a grid map, according to Bresenham's line algorithm [8], represented in Fig. 3.5.

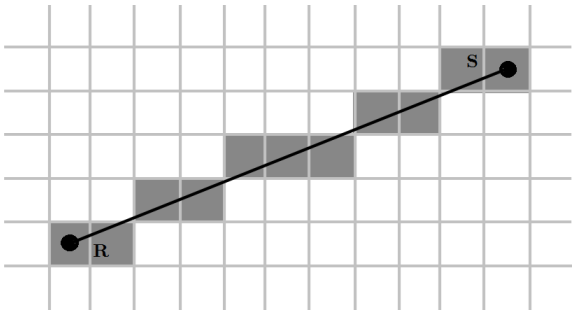


Figure 3.5: Bresenham's algorithm representation between the robot's position R and a scan endpoint S.

This algorithm is briefly summarized in Algorithm 8. The grid map generated is a logarithmic map, *i.e.* the probability of each cell being occupied is given in log odds, which makes the map update faster since multiplications are replaced by additions and the solution is numerically stable.

Algorithm 8: Bresenham Line Drawing Algorithm

```
1  (x_1, y_1): coordinates of one endpoint of the line
2  (x,y_2): coordinates of the other endpoint of the line
3  Δx = x_2 − x_1
4  Δy = y_2 − y_1
5  ε = 2 · Δy − Δx
6  for x = x_1 to x_2 do
7      SaveGridCell(x,y)
8      if ε > 0 then
9          y = y + 1
10         ε = ε − 2 · Δx
11     else
12         ε = ε + 2Δy
13     end
14 end
```

The probability $P(n|z_{1:t})$ that a cell n is occupied given the sensor measurements $z_{1:t}$ is estimated according to [17]:

$$P(n|z_{1:t}) = \left[1 + \frac{1 - P(n|z_t)}{P(n|z_t)} \frac{1 - P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1}$$

where $P(n)$ is a prior probability, $P(n|z_{1:t-1})$ the previous estimate and $P(n|z_t)$ denotes the probability that cell n is occupied given the measurement z_t. It is considered a probability $P(n|z_t) = 0.9$ to the cells containing the laser scan endpoints and a probability $P(n|z_t) = 0.3$ to the remaining free cells.

Even though the map update described previously was removed in the HectorSLAM integration to this dissertation, the mapping procedure to build a reference map for scan matching was quite similar. Since the interpolation scheme of the HectorSLAM does not allow the use of a feature map, a grid map has to be built based on the measurements acquired from the tests room. To build this grid map, a linear interpolation between the measured corners was done, *i.e.* lines were linearly drawn between consecutive points. These lines were projected into a grid map using Bresenham's line algorithm (Algorithm 8).

Chapter 4

Experimental Results

4.1 Validation Platform

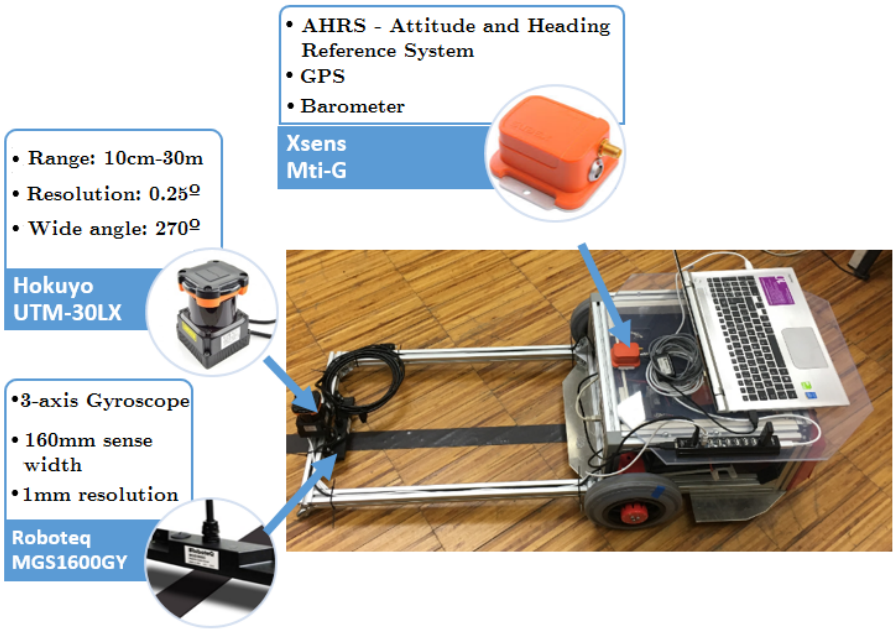


Figure 4.1: Robot structure used during tests.

Before starting to analyse the results obtained, it is important to know the laser characteristics and the platform's structure (Fig. 4.1). It is equipped with an Hokuyo UTM-30LX laser range-finder, a MGS1600GY sensor which is capable of detecting and reporting the position of a magnetic field along its horizontal axis (which was not used during this work) and a inertial measurement unit (IMU), which was also not used during this dissertation, since the main goal was to only use laser readings.

To analyse the results of tests performed, it is important to work with the robot's

base position, *i.e.* where the robot's motor, which commands the wheels movement, is located. As it can be seen in Fig. 4.1, to define the robot's base position, a transformation must be applied to the laser, along x-axis. The laser is located 0.7m along x-axis so the transformation applied is:

$$^{Robot}T_{Laser} = \begin{bmatrix} 1 & 0 & 0 & 0.7 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As it can be seen in Fig. 4.1. the laser used was Hokuyo's UTM-30LX. It uses laser source (λ = 870nm) to scan a 270º semicircular field. Table 4.1 shows the laser's main specifications. Sensor's measurement data along with the correspondent angle are transmitted through a communication channel.

Supply Voltage	12VDC +- 10%	Angular Resolution	0.25º
Guaranteed Range	0.1 $\sim$ 30m	Measurement Step	1080
Maximum Range	0.1 $\sim$ 60m	Scan Speed	25ms
Scan Angle	270º	Measurement Resolution	1mm

Table 4.1: Hokuyo's UTM-30LX laser main specifications.

This communication is made using *LaserScan* messages, included in *sensor_msgs* ROS package. Besides other attributes, this type of message contains an ordered array of 1081 range measurements from the minimum to the maximum angle. Based on these values and on the angular resolution, which is also an attribute of *LaserScan* message, it is possible to convert this measurements into cartesian coordinates as already presented before and as it i shown in Algorithm 9 shows how this conversion was done for all laser scan points. This is useful since most algorithms developed work with this type of coordinates rather than polar coordinates.

Fig. 4.2 summarizes the validation platform architecture using ROS framework. As already stated before, the communication between laser scanner and the platform is done using *LaserScan* messages, which are later converted. A more in-depth explanation in how this process works is presented on Appendix A. The robot's

Algorithm 9: Laser scan data conversion to cartesian coordinates

Data: Set of n laser scans ($ranges$), each with 1081 measurements

```
for i=1 to n do
    for j=1 to 1081 do
        x(i,j) = ranges(i,j) · cos(min_angle + (j − 1) · angle_increment);
        y(i,j) = ranges(i,j) · sin(min_angle + (j − 1) · angle_increment);
    end
end
```

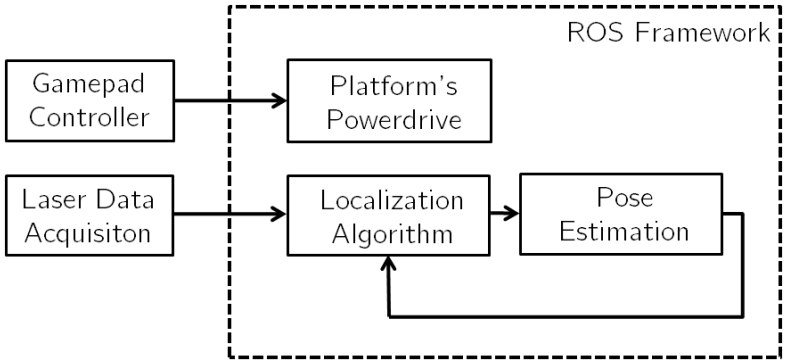


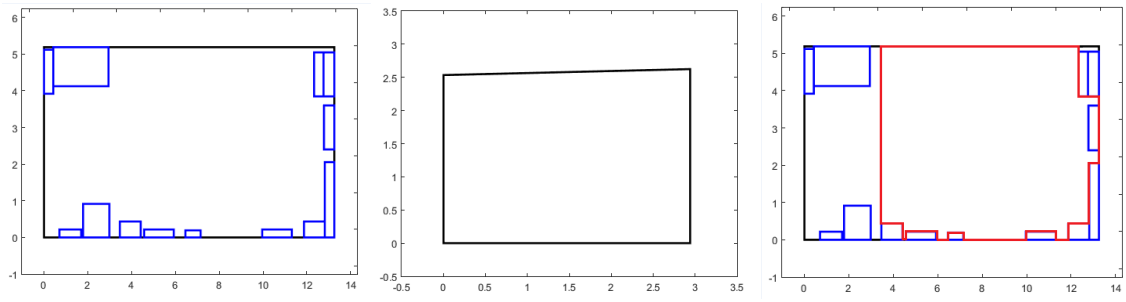
Figure 4.2: Overview of actions performed on the validation platform.

movement is controlled using a gamepad controller, which sends instructions to the platform's powerdrive.

4.2 Workspace Description

To evaluate the performance of the developed algorithms, they were tested in the same environment, presented in Fig. 4.3. According to the tests performed, this room had some changes in its topology that will be described next. As stated before, it is

Figure 4.3: Panoramic photo of the test room used during experiments.



(a) Full room with objects in it.

(b) Smaller scenario within the room.

(c) Test room used for last tests performed.

Figure 4.4: Two different scenarios used during localization tests.

crucial for scan matching that the reference map is as close to the real environment as possible. To do so, the room where tests were performed was manually measured with measuring tape and afterwards digitally constructed using those measures. Figure 4.4 shows both scenarios tested during ICP implementation. Figure 4.4a is the full room with objects inside (closets, tables and smaller objects) where most of the tests were performed. It can be seen that this scenario is slightly different from the one in Fig. 4.3, as some objects were re-arranged during tests. During ICP implementation, and with the purpose of validate the algorithm developed, a smaller scenario (Fig. 4.4b) was created to minimize the error coming from small objects detection.

The scenarios presented above were used to evaluate the algorithms' behaviour using static positions around the scenarios. To evaluate the behaviour of algorithms in later tests where the robot performed an elliptical path around the room, a different room configuration was used, as it can be seen in Fig. 4.4c, which is the scenario presented above in Fig. 4.3. This new scenario was built with the purpose of reducing the noise coming from chairs and tables that are within the part of the room that was closed, allowing a better evaluation of the algorithms developed.

4.3 Primitive Extraction

Too have a better evaluation of segmentation and primitive extraction, the full room scenario was used. This is because the smaller scenario is only composed of four lines, which are relatively simple to extract. The fact that the room has objects

in it gives the possibility of a better analysis of the algorithms implemented.

Segmentation algorithm was the first one tested. The key to get good results with the segmentation algorithm is the regulation of distance threshold D_{thd}. As it can be seen in Fig. 4.5, increasing the value of D_{thd} results in less segments, each one containing more points. As it can be seen, using a low value of D_{thd} (0.01) results in no segmentation (1081 segments/points) and using a high value of it (0.5) will result in only one segment generated (with 1081 points on it).

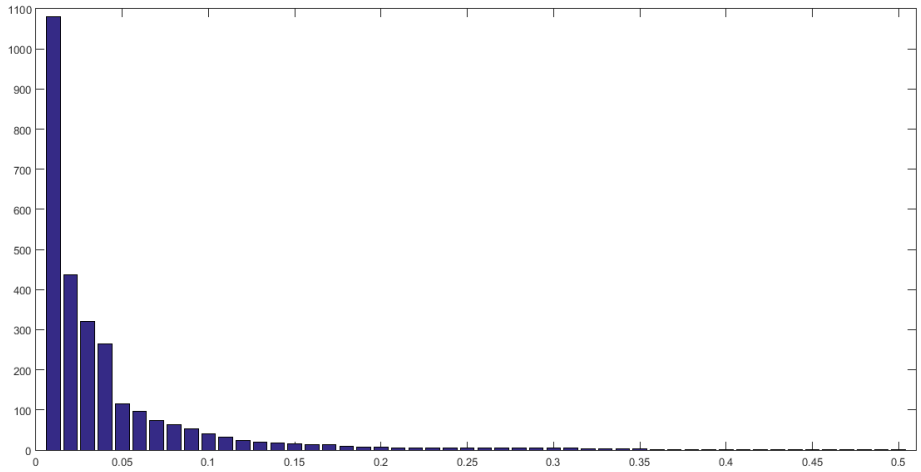
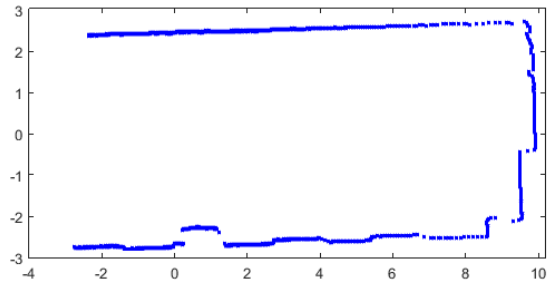


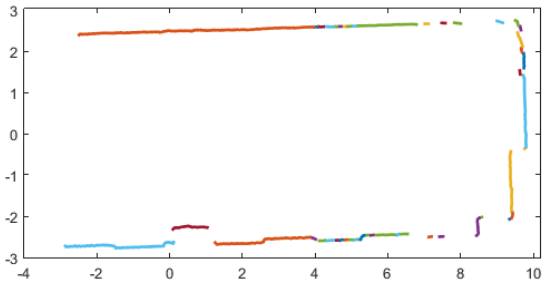
Figure 4.5: Number of segments generated in terms of distance threshold D_{thd}.

This distance threshold may be regulated to best fit the set of points used, *i.e.* it has a great dependency on the environment. However, as it can be seen in Fig. 4.6, the results of this segmentation method are not yet appropriate for line fitting and corner extraction because there are segments that can still be split. To do so, a Douglas-Peucker algorithm was applied on top of this results so that a line fitting algorithm can be used without compromising the corner extraction results.

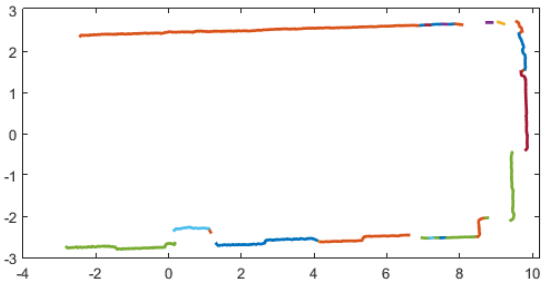
Just like the segmentation algorithm, Douglas-Peucker algorithm is also very reliable on the regulation of threshold ϵ. Figure 4.7 shows the results of Douglas-Peucker algorithm for different values of threshold ϵ. For lower values of ϵ, more segments are generated which might result in an increase error in some parts of the scenario. For higher values of ϵ, the result is closer to input from initial segmentation with some segments split, leaving however some segments that could be important untouched. This regulation will depend not just on the environment, but also on what this results are for. Fig. 4.8 shows the number of segments generated from different



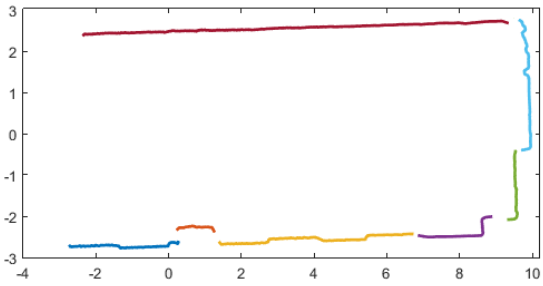
(a) Laser scan data.



(b) $D_{thd} = 0.05$.

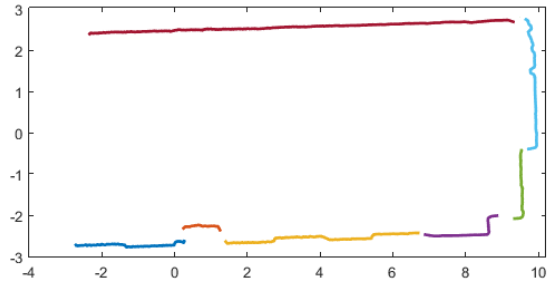


(c) $D_{thd} = 0.10$.

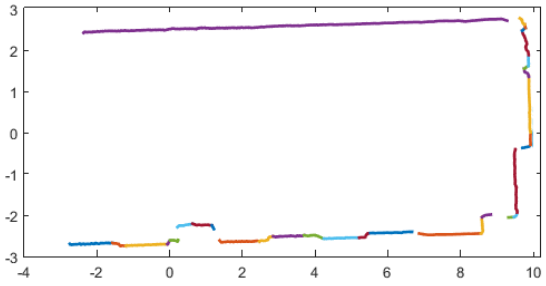


(d) $D_{thd} = 0.20$.

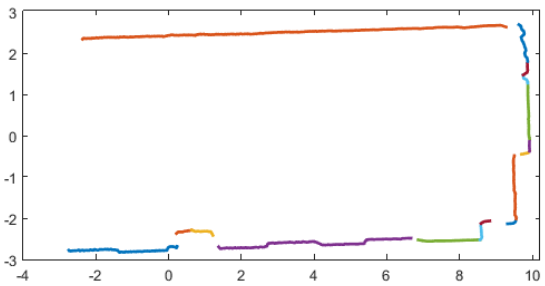
Figure 4.6: Laser scan data segmented with different values of D_{thd}.



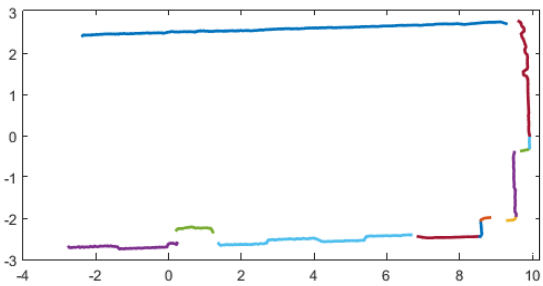
(a) Initial Segmentation ($D_{thd} = 0.20$).



(b) DP with $\epsilon = 0.05$.



(c) DP with $\epsilon = 0.10$.



(d) DP with $\epsilon = 0.15$.

Figure 4.7: Douglas-Peucker algorithm results for different values of threshold ϵ.

values of threshold ϵ. As it can be seen, this values are far more consistent than the ones observed in Fig. 4.5, which allows the implementation to be less sensitive to environment changes.

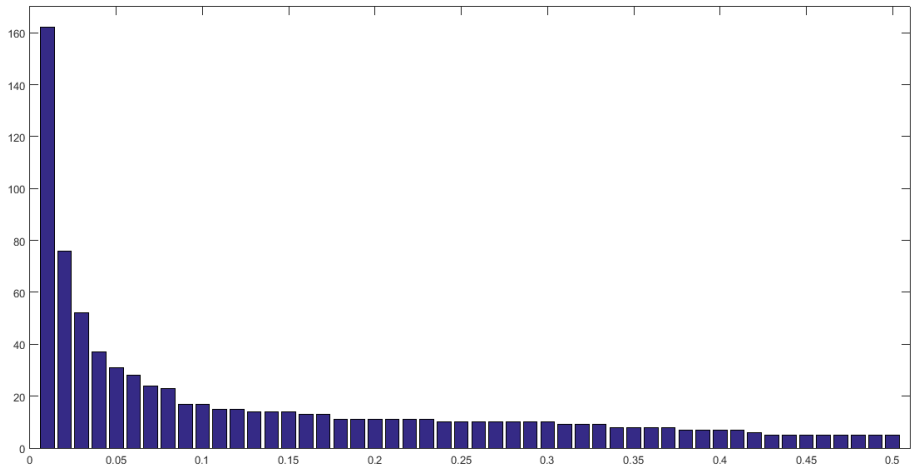


Figure 4.8: Number of segments generated in terms of threshold ϵ.

For the purpose of this dissertation, low values of ϵ compromise the results of a line fitting algorithm and so values of ϵ between 0.15 and 0.20 are the ones that give better results overall.

The initial approach followed this procedure, *i.e.* an initial segmentation applied to the laser scan data and afterwards a Douglas-Peucker algorithm to split some of segments. Because the results of DP algorithm were very promising, a second approach was made with no initial segmentation done, only applying Douglas-Peucker on the laser scan points. Even though the results were similar, the computation times of both approaches were slightly different, as shown in Table 4.2.

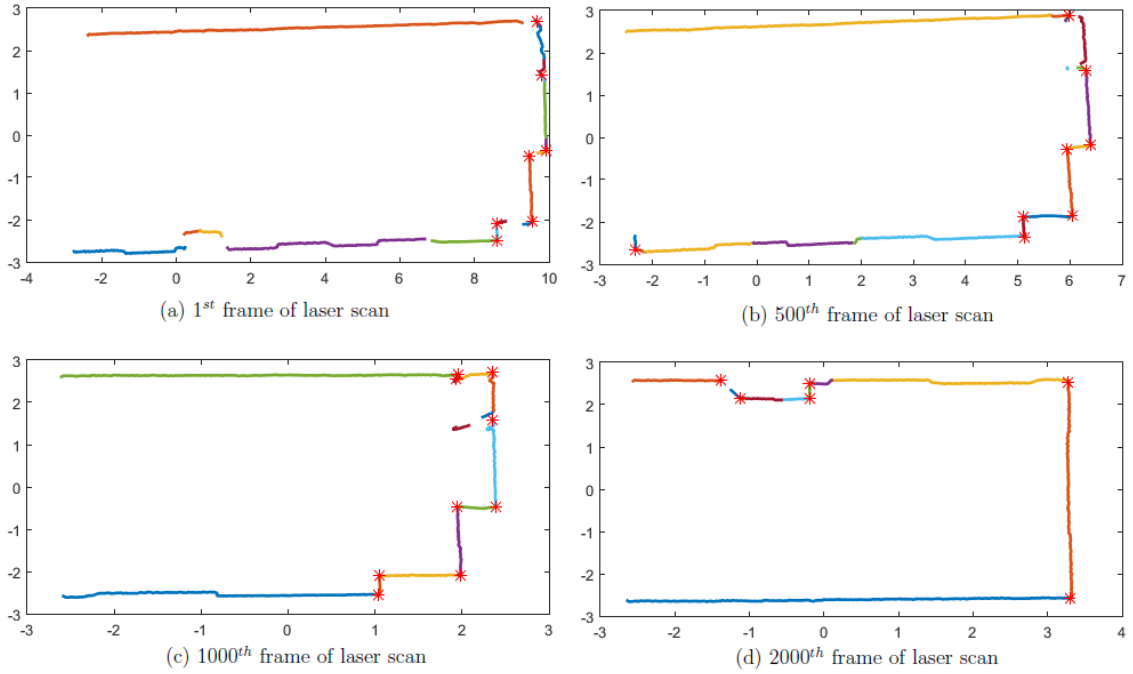
		ϵ=0.10	ϵ=0.15	ϵ=0.20
Segmentation & Douglas-Peucker	D_{thd}=0.05	0.0267	0.0257	0.0250
	D_{thd}=0.10	0.0249	0.0245	0.0244
	D_{thd}=0.20	0.0280	0.0256	0.0253
Douglas-Peucker without initial segmentation		0.0499	0.0474	0.0452

Table 4.2: Computation times with different approaches of segmentation.

As it can be seen, even though computation times are overall small, segmentation using only Douglas-Peucker algorithm without any initial segmentation is relatively

slower than segmentation following the first approach presented. This is because DP algorithm is a fairly heavy algorithm when used on a large set of points due to its recursiveness. If an initial segmentation is previously done, DP will only be used on particular segments and the computation effort will be much lighter.

Following the segmentation and Douglas-Peucker implementation, Fig. 4.9 shows the results of this implementation from random selected frames of laser scan. From



(a) 1^{st} frame of laser scan

(b) 500^{th} frame of laser scan

(c) 1000^{th} frame of laser scan

(d) 2000^{th} frame of laser scan

Figure 4.9: Corner detection after applying LSF and line intersection.

Fig. 4.9, it can be seen that, for this scenario, results are quite promising. Even though laser scan data is not optimal, because there are some important missing points that could help detect some more corners, the results still show a good corner extraction. However, segmentation and regression methods always input some error into the system, which affect ICP algorithm results aswell.

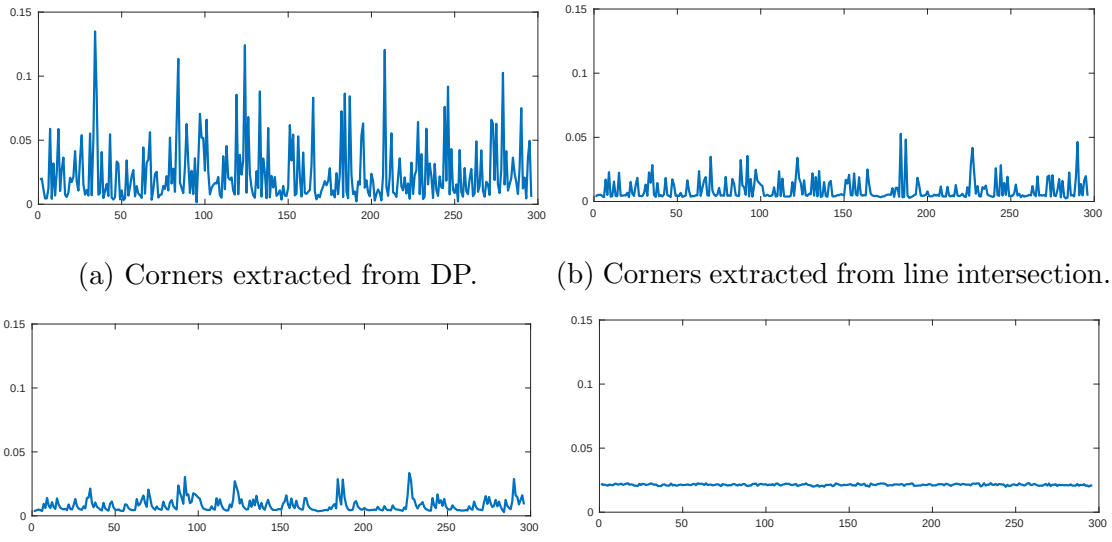
4.4 ICP Algorithm

As already stated before, the first tests were made in a smaller scenario, which only has four walls and no objects in it. Because it is such a simple scenario, corners can easily be extracted and no outliers will be detected. For that reason, when using the re-weight ICP approach, all criterion functions had exactly the same results and so Huber's criterion function was chosen for these initial tests.

Even though when the robot has no movement it can easily extract two or more corners, when a movement command is imposed, it may happen that only one corner is detected or a previous detected corner suddenly stops being detected, which may lead into pose estimation errors. To solve this problem, a feature tracking (FT) technique [3] was used, in which the detected features were kept in memory. This allows that for a few iterations after a feature stops being detected, the algorithm still consider it has being currently detected. The results obtained for this small scenario will be analysed and discussed in Section 4.4.1 and Section 4.4.2.

4.4.1 Small scenario with no movement

A first test was made with no movement applied to the robot. Figure 4.10 shows the error of robot's position applying different algorithms. As it can be seen, using only Douglas-Peucker algorithm to extract corners produces the worst results, as it would be expected. Even though using no corners on ICP algorithm gives a less variable position error, the best results are obtained when corners are extracted from line intersection (either with or without FT applied).



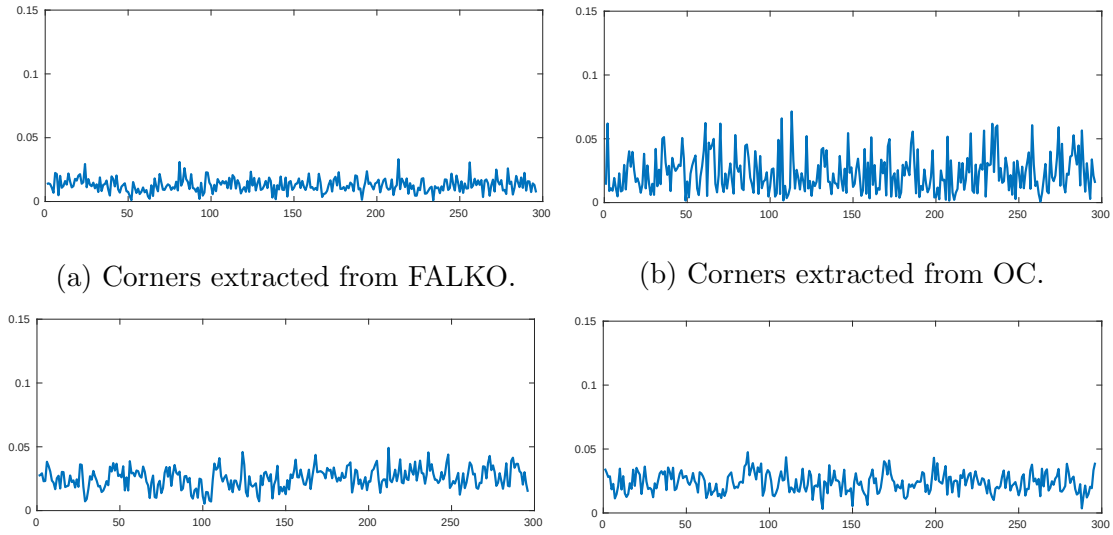
(a) Corners extracted from DP.

(b) Corners extracted from line intersection.

(c) Corners extracted and application of FT.

(d) ICP using full laser scan without corners.

Figure 4.10: Robot's position error for small scenario with no movement (1).



(a) Corners extracted from FALKO.

(b) Corners extracted from OC.

(c) Corners extracted from Normal Edge Detector.

(d) Corners extracted from Normal Blob Detector.

Figure 4.11: Robot's position error for small scenario with no movement (2).

After using the developed algorithms, four more algorithms from FLIRT and FALKO libraries (described before) were used as comparison with the previous obtained results. Both libraries are provided as C++ repositories [1][2]so it was necessary to adapt the existing code into Matlab MEX functions.

As it can be seen in Fig. 4.11, FALKO's detector is the one with the better results, even though it still has a considerable amount of deviation between error measurements.

Table 4.3 shows the Mean Error, Root-Mean-Square Error (RMSE), Standard Deviation (SD), maximum and minimum values of error for all the algorithms evaluated. The first conclusion that can be done is that, for this specific scenario and with no movement applied to the robot, all algorithms are viable because their mean error goes between 0.5-2.4cm. However, extracting corners with the developed algorithm or FALKO does give better overall results.

It is important to evaluate both position and orientation of the robot. In order to evaluate its orientation, Table 4.4 shows the angular error for the robot's orientation. Even though all error measurements are acceptable (with corners and using the full laser scan), as it can be seen, using extracted corners gives better overall results.

[1]FALKO - `https://svn.openslam.org/data/svn/falkolib/`

[2]FLIRT - `https://svn.openslam.org/data/svn/flirtlib/`

Method	Mean Error	RMSE	SD	Max.	Min.
ICP w/o Corners	0.0214	0.0214	0.0006	0.0228	0.0198
ICP w/ Corners	0.0239	0.0334	0.0233	0.1349	0.0015
ICP w/ Corners (LSF)	0.0056	0.0068	0.0038	0.0284	0.0024
ICP w/ Corners(LSF+FT)	**0.0055**	**0.0061**	**0.0027**	**0.0215**	**0.0012**
OC	0.0231	0.0277	0.0153	0.0714	0.0008
FALKO	**0.0128**	**0.0139**	**0.0052**	**0.0331**	**0.0009**
NormalBlobDetector	0.0239	0.0251	0.0077	0.0476	0.0031
NormalEdgeDetector	0.0256	0.0269	0.0082	0.0626	0.0054

Table 4.3: Error analysis for small scenario with no movement.

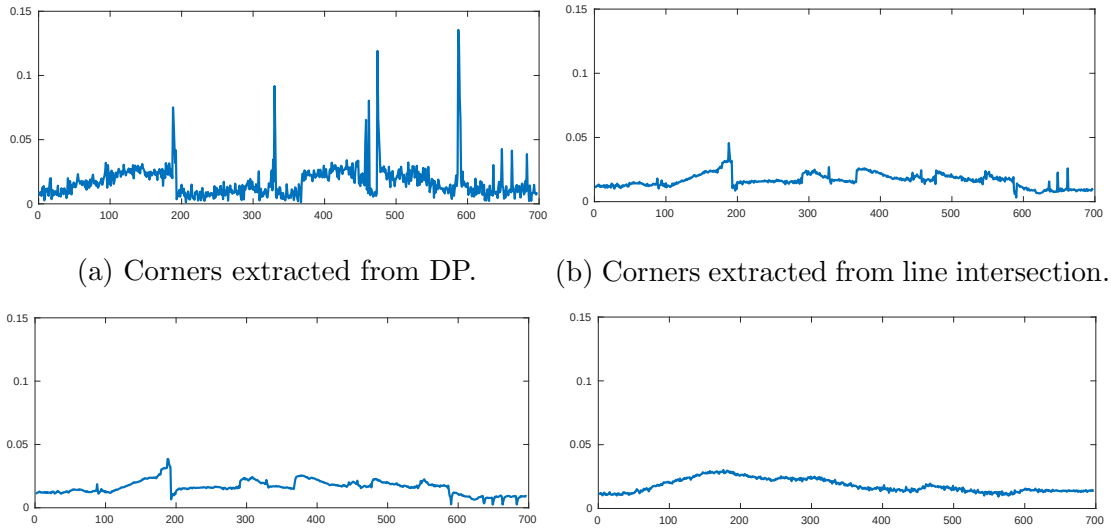
Method	Mean Error	RMSE	SD	Max	Min
ICP w/o Corners	0.6846	0.6854	0.0342	0.7717	0.5726
ICP w/ Corners (LSF)	0.1833	0.2197	0.1211	0.8335	0.0278
ICP w/ Corners(LSF+FT)	0.1833	0.1958	0.0687	0.5936	0.0875
FALKO	0.4647	0.5322	0.2594	1.2251	0.0018

Table 4.4: Angular error analysis for smal scenario with no movement.

4.4.2 Small scenario with rotation movement

Because this was the simplest test scenario possible, another similar was made, but with a rotation movement applied to the robot. The robot was placed on the centre of the scenario and a rotation of 360º was applied to him. The purpose of this test is to evaluate how well the algorithms do when new corners are detected and previous ones start to become unseen by laser range-finder. The exact same procedure was done here, with all 8 algorithms being tested.

Figure 4.12 shows the error evolution of the methods previous presented. As it can be seen, error evolution with rotation movement applied is very different from the one seen before with no movement. A few conclusions can be done from this results. First, error is higher every time a new feature (corner) is detected or a previous one is no longer detected. This is substantially notable in Fig. 4.12a, where error measurements can go up to almost 14cm, which is not acceptable. When extracting corners using line intersection (with and without FT) this error peaks still exist, however they have much lower values. Figure 4.13 shows the results for the remaining algorithms applied. It can be seen that FALKO gives the best results, just like observed before. However, just like before, it is noticeable that introducing



(a) Corners extracted from DP.

(b) Corners extracted from line intersection.

(c) Corners extracted and application of FT.

(d) ICP using full laser scan without corners.

Figure 4.12: Robot's position error for small scenario with rotation movement (1).

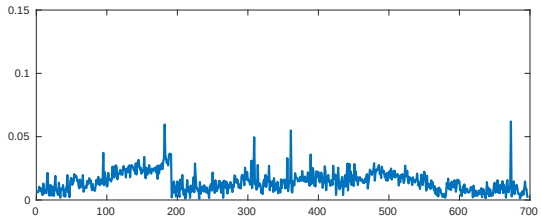
a rotation movement on the robot produces worst results and less robust position measurement.

Table 4.5 shows the results of all algorithms applied during this test with rotation movement. Like already seen before, both algorithms that extract corners using LSF (with or without FT application) have similar results, with the application of FT reducing the value of RMSE.

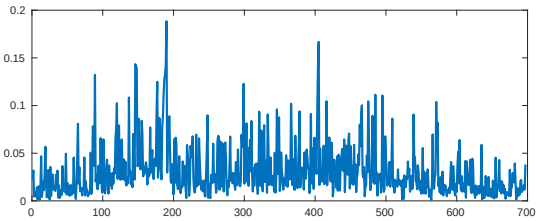
Method	Mean Error	RMSE	SD	Max	Min
ICP w/o Corners	0.0173	0.0184	0.0064	0.0296	0.0053
ICP w/ Corners	0.0171	0.0213	0.0127	0.1355	0.0012
ICP w/ Corners (LSF)	0.0160	0.0172	0.0062	0.0338	0.0028
ICP w/ Corners(LSF+FT)	**0.0160**	**0.0168**	**0.0051**	**0.0334**	**0.0013**
OC	0.0311	0.0412	0.0269	0.1884	0.0009
FALKO	**0.0143**	**0.0164**	**0.0079**	**0.0620**	**0.0009**
NormalBlobDetector	0.0223	0.0245	0.0100	0.0574	0.0012
NormalEdgeDetector	0.0217	0.0239	0.0100	0.0581	0.0002

Table 4.5: Error analysis for small scenario with rotation movement.

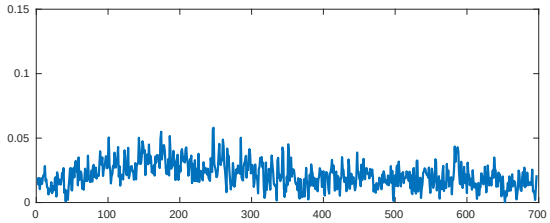
Comparing these results with FALKO's results possibilitates the drawing of one important conclusion: even though FALKO algorithm gives the best overall mean error and RMSE, it is important to notice that standard deviation is lower for the algorithm developed, which means that the error measurements are more constant for



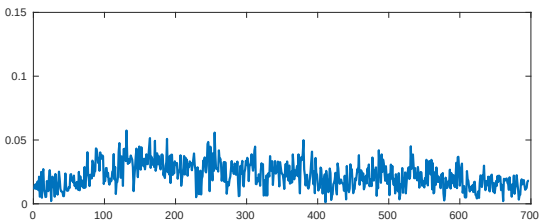
(a) Corners extracted from FALKO.



(b) Corners extracted from OC.



(c) Corners extracted from Normal Edge Detector.



(d) Corners extracted from Normal Blob Detector.

Figure 4.13: Robot's position error for small scenario with rotation movement (1).

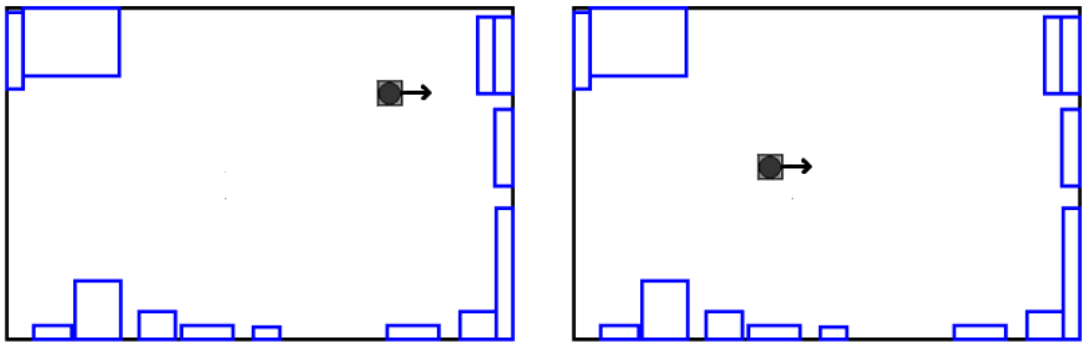
this algorithm. This aspect is even more important when looking at maximum error of both algorithms, with FALKO having more than 6cm, while the algorithm developed has, approximately, 3.34cm. This consistency shows that this last algorithm is more robust than FALKO's algorithm for the scenarios tested.

Because these algorithms rely so much on environment features, it is important to analyse these algorithms for a bigger and more complex scenario, which will be done in Section 4.4.3.

4.4.3 Full room scenario

After testing the localization algorithm in the small scenario, it is necessary to evaluate it in a bigger one. To do so, two kind of tests were performed: the first one was a static test, i.e not giving any movement to the robot (neither translational or rotational), in three different positions around the room - it is a quantitative test since the main purpose is to compare the position error for the different algorithms; in the second test, the robot had both translational and rotational movement, performing a path around the room returning to the start point - it is a qualitative test, in which the main goal is to evaluate if both starting position and end position match.

The first position tested is shown in Fig. 4.14a. Table 4.6 presents the results obtained for the position error for this position. Only the algorithms that shown the best results in the previous tests were used, which are ICP without the use of corners, ICP with the use of corners extracted (both with the application of Kalman Filter and not) and FALKO's keypoint extractor.



(a) First static position used to evaluate algorithms' results in full room scenario.

(b) Second position used to evaluate algorithms' results in full room scenario.

Figure 4.14: Elliptical path performed without obstacles in it using both extracted features and full laser scan approaches.

As it can be seen from Table 4.6, all the results are similar, with the algorithm of ICP without corners performing slightly better. This happens because the other three algorithms depend on feature extraction, which might not be done correctly if the laser range-finder is as close to an object as it is in Fig. 4.14a.

Method	Mean Error	RMSE	SD	Max	Min
ICP w/o Corners	0.0200	0.0200	0.0006	0.0215	0.0187
ICP w/ Corners (LSF)	0.0310	0.0312	0.0033	0.0410	0.0222
ICP w/ Corners(LSF+FT)	0.0301	0.0301	0.0015	0.0337	0.0259
FALKO	0.0316	0.0320	0.0051	0.0455	0.0163

Table 4.6: Error analysis for full room (position 1).

Table 4.7 show the angular error regarding the laser orientation. As it can be seen, the results from using ICP with corners are slightly better than with the full laser scan, like it was seen in the smaller scenario.

The second position tested was close to the centre of the room, as shown in Fig. 4.14b, where the laser scan can gather a wider amount of data and, consequently,

Method	Mean Error	RMSE	SD	Max	Min
ICP w/o Corners	0.4165	0.4174	0.0272	0.4651	0.2834
ICP w/ Corners (LSF)	0.2015	0.2059	0.0427	0.3215	0.0658
ICP w/ Corners(LSF+FT)	0.2032	0.2055	0.0307	0.2725	0.1336
FALKO	0.2748	0.3032	0.1281	0.6181	0.0128

Table 4.7: Angular error analysis for full room (position 1).

more amount of features. This is particularly helpful when using ICP algorithms that rely on feature extracted, like the ones being tested.

As it can be seen in Table 4.8, there is not much difference here between the algorithms tested, with the use of corners performing just slightly better than the ICP with the full laser scan.

Method	Mean Error	RMSE	SD	Max	Min
ICP w/o Corners	0.0297	0.0297	0.0017	0.0345	0.0260
ICP w/ Corners (LSF)	0.0298	0.0299	0.0025	0.0392	0.0198
ICP w/ Corners(LSF+FT)	0.0291	0.0291	0.0017	0.0373	0.0221
FALKO	0.0245	0.0252	0.0060	0.0434	0.0110

Table 4.8: Error analysis for full room (position 2).

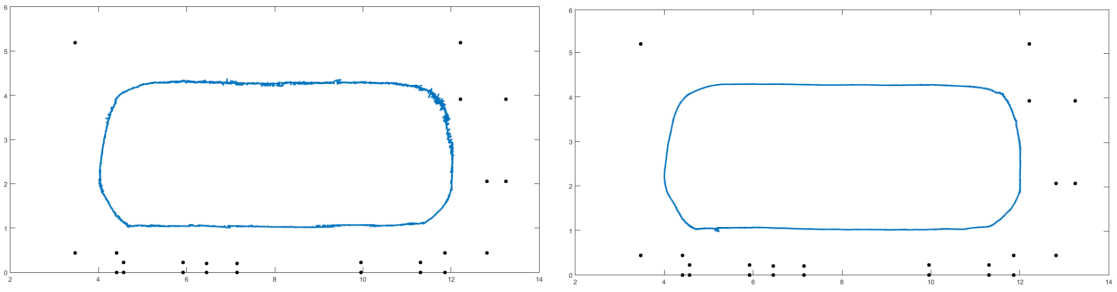
For this position, angular error is higher, specially for algorithms that use extracted corners. However, the results are overall good, without the error being higher than 1^{o}.

Method	Mean Error	RMSE	SD	Max	Min
ICP w/o Corners	0.4819	0.4822	0.0164	0.5265	0.4350
ICP w/ Corners (LSF)	0.5563	0.5577	0.0391	0.7483	0.4573
ICP w/ Corners(LSF+FT)	0.5512	0.5517	0.0246	0.6516	0.4983
FALKO	0.5689	0.5811	0.1182	0.9577	0.2535

Table 4.9: Angular error analysis for full room (position 2).

As stated before, there was also done a test performing an elliptical path inside the latest scenario presented (Fig. 4.4c), where two different scenarios were tested. The normal scenario has no obstacles in it, *i.e.* objects that are not present on the reference map used for scan matching. The dynamic scenario contains dynamic obstacles that appear along the path, in this case it is a person crossing the robot's path multiples times during the whole dataset recording.

Figure 4.15 shows the results obtained from the elliptical path using both extracted features and full laser scan in the normal scenario (where the blue line is the robot's position and the black dots are the natural features existent on the environment). As it can be seen in Fig. 4.15a, even though there is some noise during the path (specially near right upper corner), the path is closed, which means that the robot returns to his initial position as it was done during the dataset recording. In Fig. 4.15b the path has overall less error, which indicates that for more complex scenarios, feature extraction methods might be compromised when features can not be clearly extracted. This problem will later be mitigated with the integration of the HectorSLAM, as it will be seen ahead.



(a) Extracted corners (normal scenario). (b) Full laser scan (normal scenario).

Figure 4.15: Elliptical path performed without obstacles in it using both extracted features and full laser scan approaches.

After the first tests performed with no obstacles, a dynamic scenario was done with the purpose of evaluating the performance of the algorithms explored when dynamic obstacles (people crossing th robot's path) were introduced. Algorithms using extracted features did not perform well in this scenario, because when obstacles came into the path, multiple features were not able to be seen causing the algorithms to have high pose estimation errors. Figure 4.16 shows the results of the dynamic scenario using the full laser scan for scan matching. Although the results from the normal scenario were quite good, the presence of obstacles introduced many position estimation errors during this path. This shows that even scan matching algorithms that use the full laser scan as input do not have the robustness against obstacles needed for non-test environments (*e.g.*, industrial environments).

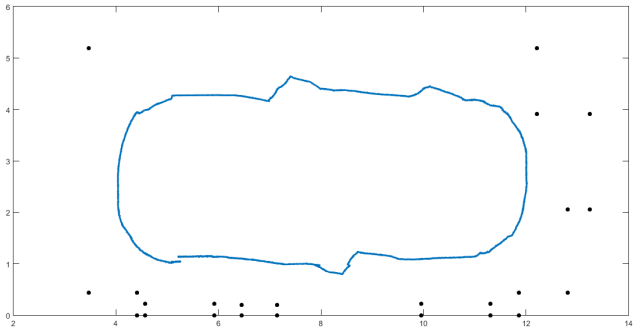


Figure 4.16: Elliptical path performed with dynamic obstacles in it using full laser scan.

4.5 Applying Corner Features to the HectorSLAM

The main goal behind the HectorSLAM integration on the developed algorithms is to increase the robustness of the localization process, without increasing the pose estimation error (or if possible, trying decrease its value). In order to do so, a few approaches were done and will now be presented and discussed.

Three approaches were done concerning the HectorSLAM integration on the algorithms explored and developed. Even though these approaches use the HectorSLAM on the same way, they differ in how the data is introduced into this SLAM method. Also, these approaches try to mimic the behaviour of a SLAM technique in an industrial environment, *i.e.* its given an *a priori* map of the environment to the HectorSLAM and this method uses that map to perform localization without generating or updating the map. The first approach uses the full laser scan as input, *i.e.* this is the standard the HectorSLAM and is mainly used to compare with other algorithms' results. A second approach was then done using a map of natural features extracted as input, instead of all laser data. The third and last approach was done using regions of interest as input. Features extracted by FALKO are considered points of interest and so it was interesting to consider a neighbourhood around each interest point extracted (region of interest). Table 4.10 shows the initialization parameters used in FALKO during the tests performed.

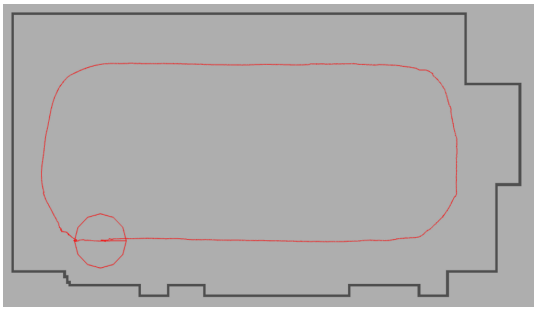
Figure 4.17 show the results for the elliptical path performed in the latest scenario presented (Fig. 4.4c) for the three approaches described above. As it can be seen in Fig. 4.17, the overall results with the integration of the HectorSLAM are quite good.

Minimum Score Threshold	50	Neighbourhood Parameter b	0.05
Minimum Extraction Range	0.01	Neighbourhoot Minimum Point	2
Maximum Extraction Range	30	β Ratio	5.5
NMS Radius	0.03	Number of Grid Sectors	16
Neighbourhood Parameter a	0.1		

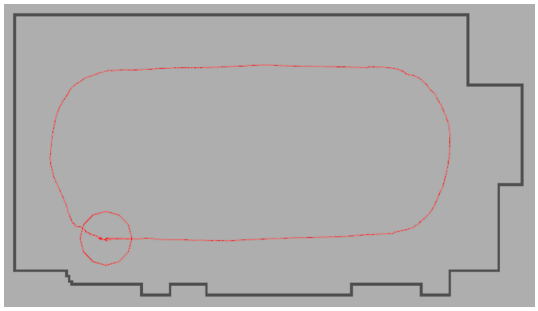
Table 4.10: Initialization parameters used in FALKO's extractor.

Results using the full laser scan (standard the HectorSLAM) are good as expected in both scenarios. As it can be seen, using the HectorSLAM with extracted features (keypoints) had some issues around the room, specially on the left down corner. This happens due to the fact that in the area there are many close objects which makes the corners hard to extract in some of the laser scans. The lack of information about a certain area of the map due to the fact that corners are not easily extracted compromises the results of the localization, which is solved using regions of interest. Using additional information around each extracted interest point will bring more scan matching correspondences and so better overall results.

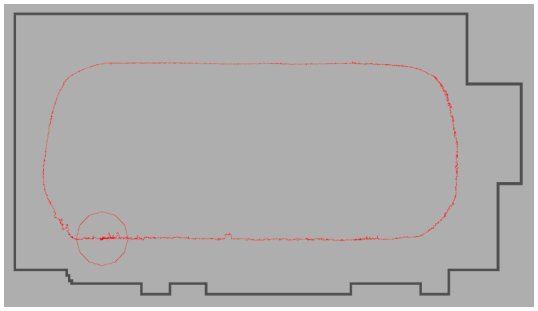
As it can be seen in Fig. 4.18a, the area mentioned above gathered by laser scan does not match with precision the real map of the environment. This can mislead the behaviour of the algorithm, specially when using features since the scan presented does not allow the extraction of important corners that should be visible. In Fig. 4.18b there is another example of what can be a problematic scenario, in which the presence of an obstacle occludes some of the main features of the environment. However, it is notable that on either of these two scenarios, the methods developed had good results overall, which shows that the integration of the HectorSLAM really brought more robustness and flexibility to the system.



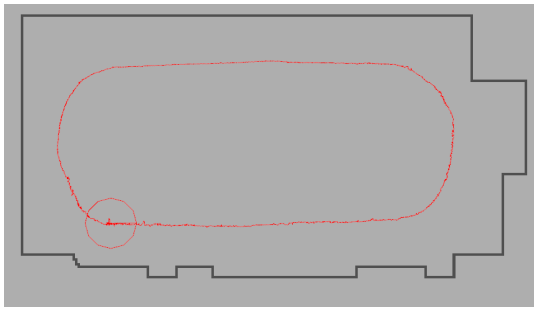
(a) Full laser (normal scenario).



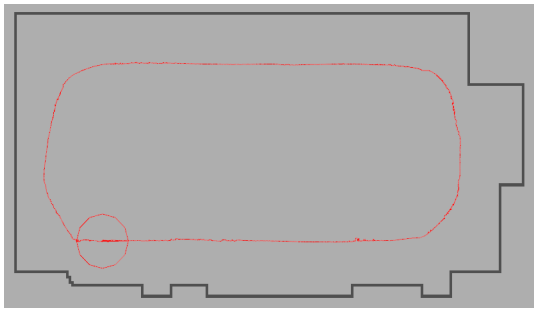
(b) Full laser (dynamic scenario).



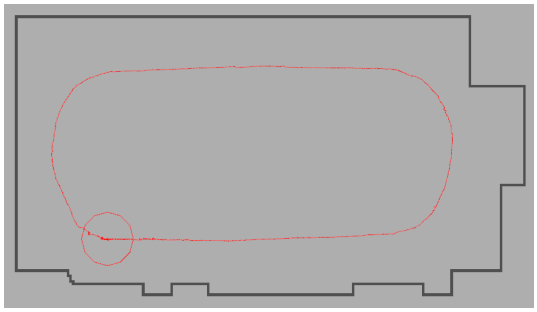
(c) Extracted corners (normal scenario).



(d) Extracted corners (dynamic scenario).



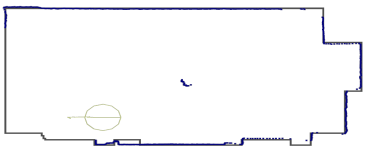
(e) Regions of interest (normal scenario).



(f) Regions of interest (dynamic scenario).

Figure 4.17: Position estimation for elliptical path in both normal and dynamic scenarios.

(a) Comparison between real map and laser scan.



(b) Laser scan with an obstacle present.

Figure 4.18: Two possible situations that can mislead localization algorithm.

Chapter 5

Conclusion and Future Work

This dissertation focused on the research of localization algorithms for mobile robots (more specifically an AGV) using only laser data. Even though localization methods are still very conservative, there has been an evolution over the years to try to simplify these methods, without inputting more any more error into it. In that way, some important conclusions can be drawn from the work developed in this dissertation.

Regarding feature extraction and scan matching with ICP algorithm the results were good overall. However, the different scenarios used had indeed repercussions on the behaviour of the algorithms developed. On one hand, using a small scenario with no more than four walls (no objects inside introducing noise), allowed the algorithms developed using feature extraction to have the best results amongst others tested. On the other hand, when a bigger scenario was used, the results with and without feature extraction were similar, since the error from extracting these features was higher. It is however important to refer that even on feature extraction algorithms worst results, the mean error obtained was around 2-3cm, which is completely acceptable for the scenarios tested.

The HectorSLAM brought robustness to the localization algorithms evaluated before. Even though with a simple scenario all algorithms could do a good pose estimation, when a dynamic scenario was used, all of them had some problems. With the integration of the HectorSLAM these problems were mitigated, with this method showing some promising results, that can be explored in the future.

The work developed in this dissertation showed that when environment natural

features can be easily extracted, a scan matching algorithm using those features has overall better results than when using scan matching using the full laser scan. The main drawback of this method is when features can not be extracted so easily, *i.e.* when there is a great amount of noise on the laser scan data coming from obstacles that are not in the map used as reference for scan matching. Even though this was not so problematic with the integration of the HectorSLAM, it is important to highlight that there is still work to be done concerning feature extraction. Since this is the core of the algorithm developed, enhancing the feature extraction process will also improve the performance of the localization algorithm overall.

Appendix A

Background and Datasets

Before starting to analyse the algorithms developed and their results, it is important to have a background on how the tests were performed. This chapter focus not only on that, but also on giving an overview of the system used for that purpose: Robot Operating System (ROS). A brief analysis over this system will be done in order to understand how the datasets were taken and the nomenclature used to describe it.

A.1 Robot Operating System

The work presented in this dissertation was developed in Robot Operating System (ROS) , so a brief overview of this system is extremely important. ROS is the most popular robotics framework nowadays. It is a modular, tools-based software, providing libraries and drivers in order to help the development of robotic applications, enabling researchers to perform simulations and real world experiments in a simple and reliable way.

A.1.1 System Design

As described in [29], ROS can be summarized in five important characteristics: peer-to-peer service, tools-based, multi-lingual, thin, free and open-source.

- *Peer-to-peer*: ROS has a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using its communication infrastructure. A peer-to-peer architecture coupled to a buffering system and

a lookup mechanism (called *master* in ROS), allows each component to communicate directly with any other;

- *Tools-based*: Instead of a monolithic runtime environment, ROS adopted a micro-kernel design, where a large number of small tools are used to build and run the various system components. The advantage of this solution is that a problem with one tool (executable) won't affect the others, which makes the system more robust and flexible than a system based on a centralised design;

- *Multi-lingual*: The choice of a programming language can change based on personal preference and because of that, ROS is designed to be language-neutral. The main libraries are written in C++, Python and LISP, however it can support many other languages, such as Octave, Matlab or Java. Peer-to-peer connections are negotiated in XML-RPC, which exists in a great number of languages;

- *Thin*: ROS developers intend for all drivers and algorithms to be contained in standalone executables that have no dependencies on ROS. This allows code reusing and, above all, keeps its size down. Unit testing is also easier when the code is split into libraries, as standalone programs can be used to exercise different library features;

- *Free and Open-Source*: ROS source code is publicly available. Because of the characteristics exposed before, it is extremely easy to exchange code and develop it. This was particularly useful in the development of this work, as it allowed the use of already written code.

A.1.2 File System

ROS resources are organized hierarchically on disk. There are two important concepts to retain about ROS file system: a **package** is a directory containing nodes (explained bellow), libraries, data and most importantly a **manifest** file (pack-

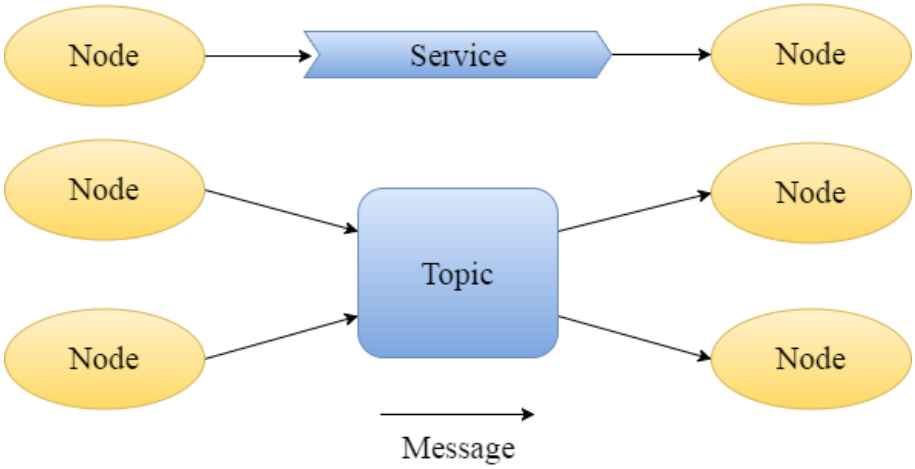


Figure A.1: ROS basic communication structure.

age.xml, for example), describing the package and its dependencies on other packages; a collection of packages is usually known as a **stack** and its primary goal is to simplify the process of code *sharing*. A collection of stacks is called **distribution**. During this work, **ROS Kinetic Kame distribution** was used.

A.1.3 Nomenclature

To understand some of the notation used in this dissertation, it is important to know how ROS works and what is the nomenclature used in it [1]. Figure A.1 shows the basic ROS components and how they communicate between themselves.

A **node** is a process that performs computation. These nodes are meant to work together in a robot control system. For example, for a differential robot like the one used on this work, one node controls a laser range-finder, one node performs localization, one node performs path planning, and so on.

Nodes communicate with each other by passing **messages**. A message is a data structure, comprising a combination of primitive types (integer, float, boolean, etc.) and arrays of primitive types and constants.

Messages are sent between nodes using **topics**. A node can either publish data into a topic or subscribe a topic (read data from it). There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. Data is exchanged asynchronously by means of a topic.

For synchronous transactions, a **service** is the most appropriate type of communication between nodes. Services are defined by a pair of messages structures: one

for the request and one for the reply.

Another important ROS concept is the **bag**. Bags are formats for storing and playing back data, very useful to store data that can be difficult to collect and use it to develop and test algorithms, as if it was real time data. During this dissertation, all the tests performed were done on bag files (datasets) and because of that it is important to have a careful look at it.

A.2 Dataset Extraction

Most of the tests performed were done using Matlab, in order to evaluate results in an easier and better way. To analyse the data gathered from laser range-finder, it has to be saved in a bag file and then transformed into a text file, so it can be worked out in Matlab (Fig. A.2).

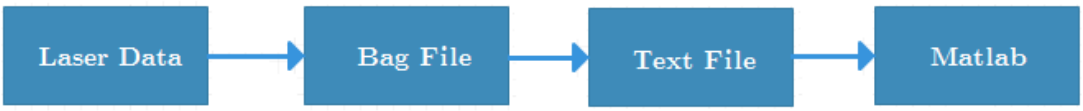


Figure A.2: Dataset extraction scheme.

The validation platform has many sensors on it and so there are many topics published at the same time, such as odometry, inertial measurement unit (IMU) or laser scan data, which is the one wanted. To record this topic into a bag file, the following command has to be typed:

$$\$ \text{ rosbag record /scan },$$

where /scan is the topic associated with laser scan data. Because Matlab itself does not handle bag files (without the use of extra toolboxes), it is required that the bag file created is converted in one file that can be read by Matlab. The bag file is then written into a text file using the command:

$$\$ \text{ rostopic echo /scan} > \text{dataset.txt}$$

where dataset.txt is the desired converted bag file.

After having a text file with the laser scan data, it is important to retrieve the desired information. A *LaserScan* message is a predefined ROS message, transversal

to most robots with laser scanners which makes it easier to work with different types of robots with laser range-finders. The format of this type of messages is presented in Fig. A.3. Most of the information contained in these messages is about

```
#
# Laser scans angles are measured counter clockwise, with 0 facing forward
# (along the x-axis) of the device frame
#

Header header
float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]
float32 time_increment   # time between measurements [seconds]
float32 scan_time        # time between scans [seconds]
float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]
float32[] ranges         # range data [m] (Note: values < range_min or > range_max should be dis
carded)
float32[] intensities    # intensity data [device-specific units]
```

Figure A.3: *LaserScan* message format [1].

laser specifications, such as its angle width, its range or its scan time. The most useful information is on the array *ranges*, where all the distances between the laser and the objects detected are stored. Since it stored in a text file, this array can be easily extracted to Matlab and worked from there. This range measurements alongside with the correspondent angle form the polar coordinates, which can later be converted to cartesian coordinates as it will be seen ahead.

Bibliography

[1] Ros Wiki. `http://wiki.ros.org/`. Accessed: Mar, 2017.

[2] Tim Bailey and Hugh Durrant-Whyte. Simultaneous Localization and Mapping SLAM: Part II. In *IEEE Robotics & Automation Magazine, Vol. 13, pp 108-117*, 2006.

[3] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[4] Per Bergström and Ove Edlund. Robust registration of point sets using iteratively reweighted least squares. In *Computational Optimization and Applications*, 2014.

[5] Per Bergström and Ove Edlund. Robust registration of surfaces using a refined iterative closest point algorithm with a trust region approach. In *Numerical Algorithms*, 2017.

[6] Paul J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Feb 1992.

[7] G.A. Borges and M.-J. Aldon. Line Extraction in 2D Range Images for Mobile Robotics. In *Journal of Intelligent & Robotic Systems*, 2004.

[8] J. E. Bresenham. Algorithm for Computer Control of a Digital Plotter. *IBM Syst. J.*, 4(1):25–30, March 1965.

[9] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, Dec 2016.

[10] Andrea Censi. An ICP variant using a point-to-line metric. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[11] K. C. J. Dietmayer, J. Sparbert, and D. Streller. Model based object classification and object tracking in traffic scenes from range images. In *IEEE Intelligent Vehicles Symposium*, 2001.

[12] Albert Diosi and Lindsay Kleeman. Fast Laser Scan Matching using Polar Coordinates. In *The International Journal of Robotics Research*, Oct 2007.

[13] David H. Douglas and Thomas K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. In *The International Journal for Geographic Information and Geovisualization, Vol. 10, No. 2 (pp. 112-122)*, 1973.

[14] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, Inc, NY, 1973.

[15] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Communications of the ACM, Vol. 24, pp 381-395*, 1981.

[16] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. In *IEEE Transactions on Robotics, Vol. 23, pp 34-46*, 2007.

[17] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Auton. Robots*, 34(3):189–206, April 2013.

[18] Fabjan Kallasi, Dario Lodi Rizzini, and Stefano Caselli. Fast Keypoint Features From Laser Scanner for Robot Localization and Mapping. In *IEEE Robotics and Automation Letters*, 2016.

[19] Stefan Kohlbrecher, Oscar von Stryk, Johannes Meyer, and Uwe Klingauf. A Flexible and Scalable SLAM System with Full 3D Motion Estimation. In *IEEE*

International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2011.

[20] M. Lauer, S. Lange, and M. Riedmiller. Calculating the Perfect Match: an Efficient and Accurate Approach for Robot Self-Localization. In *Robocup 2005: Robot soccer world cup IX, Springer, pp. 142-153*, 2006.

[21] John J. Leonard and Hugh Durrant-Whyte. Mobile Robot Localization by Tracking Geometric Beacons. In *IEEE Transactions on Robotics and Automation, Vol. 7, pp 376-382*, 1991.

[22] Jiayuan Li, Ruofei Zhong, Qingwu Hu, and Mingyao Ai. Feature-based laser scan matching and its application for indoor mapping. *Sensors*, 16(8):1265, aug 2016.

[23] David G Lowe. Object Recognition from Local Scale-Invariant Features. In *IEEE International Conference on Computer Vision*, 1999.

[24] Feng Lu and Evangelos Milios. Robot Pose Eestimation in Unknwon Environments by Matching 2D Range Scans. In *Journal of Intelligent and Robotic Systems*, Mar 1997.

[25] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Eighteenth national national conference on Artificial Intelligence, pp 593-298*, 2002.

[26] E. Pedrosa, A. Pereira, and N. Lau. Efficient localization based on scan matching with a continuous likelihood field. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 61–66, April 2017.

[27] Cristiano Premebida. Detecção e Classificação de Objectos em Ambiente Exterior para Veículos Autónomos. Master Thesis, Departamento de Engenharia Eletrotécnica e de Computadores, Universidade de Coimbra, Portugal, 2007.

[28] Cristiano Premebida and Urbano Nunes. Segmentation and Geometric Primitives Extraction from 2D Laser Range Data for Mobile Robot Applications. In

5rd National Festival of Robotics Scientific Meeting (ROBOTICA), Coimbra, Portugal, 2005.

[29] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: An open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[30] Urs Ramer. An Iterative Procedure for the Approximation of Plane Curves. In *Computer Graphics and Image Processing, Vol. 1, No. 3 (pp. 244-256)*, 1972.

[31] Szymon Rusinkiewicz and Marc Levoy. Efficient Variants of the ICP Algorithm. In *Third International Conference on 3D Digital Imaging and Modeling*, 2001.

[32] J. M. Santos, D. Portugal, and Rui P. Rocha. An Evaluation of 2D SLAM Techniques Available in Robot Operating System. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Linköping, Sweden, Oct 2013.

[33] S. Santos, J. Faria, F. Soares, R. Araujo, and U. Nunes. Tracking of Multi-Obstacles with Laser Range Data for Autonomous Vehicles. In *3rd National Festival of Robotics Scientific Meeting (ROBOTICA)*, 2003.

[34] Pranab Kumar Sen. Estimates of the regression coefficient based on kendall's tau. In *Journal of the American Statistical Association, 63(324), (pp. 1379–1389)*, 1968.

[35] Cyrill Stachniss, Udo Frese, and Giorgio Grisetti. OpenSLAM. `https://openslam.org/`. Accessed: May, 2017.

[36] Henri Theil. A Rank-Invariant Method of Linear and Polynomial Regression Analysis. In *Advanced Studies in Theoretical and Applied Econometrics (pp. 345–381), Springer Netherlands*, 1992.

[37] Sebastian Thrun and Arno Bücken. Learning maps for indoor mobile robot navigation. Technical Report CMU-CS-96-121, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 1996.

[38] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[39] Gian Diego Tipaldi and Kai O. Arras. FLIRT - Interest Regions for 2D Range Data. In *IEEE International Conference on Robotics and Automation*, 2010.

[40] J. Vandorpe, H. Van Brussel, and H. Xu. Exact Dynamic Map Building for a Mobile Robot using Geometrical Primitives Produced by a 2D Range Finder. In *IEEE International Conference on Robotics and Automation*, 1996.