TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

# Application of computer vision techniques for laser-based global localization of a mobile robot in a known, static environment

## BACHELOR THESIS

Performed with the purpose of obtaining the academic degree of

Bachelor of Science (BSc)

Under the supervision of

Paloma de la Puente Yusty

Submitted to

Vienna University of Technology
Faculty of Electrical Engineering and Information Technology
Institute of Automation and Control

by

Miguel Castillón Sánchez
Matriculation number: 1428044

<Vienna, June 19th, 2015>

Vision for Robotics Group
A-1040 Vienna, Gußhausstraße 27, Internet: http://www.acin.tuwien.ac.at

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

# DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. MIGUEL CASTILLÓN SÁNCHEZ,

con nº de DNI 73134466D en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo)

Application of Computer Vision techniques for laser-based global localization of a mobile robot in a known, static environment,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, a 19 de junio de 2015

Fdo: MIGUEL CASTILLÓN SÁNCHEZ

# Resumen

El problema de la localización global, es decir, la localización de un robot en un entorno conocido sin información sobre sus estados previos, ha sido ampliamente estudiado mediante diferentes métodos en la literatura. Un gran número de algoritmos han sido presentados y probados considerando varias condiciones.

El objetivo de este trabajo es comparar diferentes métodos de representación de ocupación de celdillas construidos a partir de datos provistos por un láser utilizando técnicas de visión por computador. El enfoque seguido usa comparación visual de un mapa global con uno local, creado gracias al sensor láser. Esta elección permite una fácil integración con sistemas robóticos existentes al mismo tiempo que evita problemas típicos de soluciones puramente visuales, como la influencia de cambios de iluminación.

Los dos algoritmos comparados—basados en área y en puntos característicos, respectivamente—han sido evaluados en un entorno interior simulado. Se ha asumido una situación estática, sin presencia de obstáculos móviles.

El trabajo ha sido realizado durante un programa de movilidad en el extranjero, por lo que el resto del documento se adapta en fondo y forma a las especificaciones de la Universidad de destino.

# Abstract

The problem of global localization, that is, the localization of a robot in a known environment without information of its previous states, has been vastly studied through different approaches in the literature. A large number of algorithms have been presented and tested considering various conditions.

The aim of this thesis is to compare different methods for occupancy-grid representations built from laser data using Computer Vision techniques. The approach followed uses visual matching of a global and a local map, created from data provided by a laser sensor. This choice allows an easy integration with existing robotic systems at the same time that avoids typical problems of purely visual solutions, such as the influence of lighting changes.

The two algorithms compared—area-based and feature-based, respectively—have been tested in a simulated indoor environment. A static situation has been assumed, without presence of mobile obstacles.

# Contents

# List of Figures

# 1 Description of the Problem

One of the most fundamental abilities required for the navigation of a mobile robot is that of self-localization, which consists on the recognition of the current position. A large number of methods concerning different approaches have been proposed and tested, such as [2] [3] [4].

The self-localization of a robot is usually divided into two different parts: position tracking and global localization, as stated in [5]. The first subproblem assumes that the initial location of the robot is known and tries to estimate the following ones incrementally, based on the last estimation and the data provided by the sensors. The second part, on the other hand, concerns the relocalization of the robot under global uncertainty. This means that the position of the robot in previous instants is completely unknown, and the current location may be anywhere inside the global map.

The aim of this thesis is to analyze a possible solution for the second subproblem of self-localization: the global localization of a robot. It is also defended in [5] that the ability to recognize places using landmarks usually plays an important role in obtaining better localization results, since high level place recognition can detect more distinctiveness. In [6] it is stated that the key issue in the so-called kidnapped-robot problem is that of properly matching sensor data provided by exteroceptive sensors to a world model. The reason is that the information given by proprioceptive sensors lacks of utility, since the state in previous instants is not known.

In the realization of this project the following hypotheses are assumed:

- The global environment is accurately modeled and available for the matching process—or can be a priori modeled. This is a reasonable premise, as it would not be possible for the robot to find its position if it is in an unknown area. However, it would not be appropriate for exploration purposes.
- The environment is static, which means that the obstacles that can be encountered do not change their position over time. It restricts its application in real-life situations, but it may be considerably useful in controlled environments with few users.
- The situation is two-dimensional, that is, the motion of the robot always takes place in the $x - y$ plane, as shown in Figure 1.1. It is not a very limiting condition, as the robot is supposed to remain always on the floor in a flat, indoor environment. The main advantage is that the computation costs are significantly smaller than in a 3-D case.
- The robot is equipped with a laser sensor—or similar—which provides the data needed for the creation of the maps. This feature can be found in

most of the robots used.

The solution of the problem consists on the calculation of the three variables for position ($\xi$) in a 2-D space referred to the global coordinate system, defined in equation (1.1).

$$\xi \equiv \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{1.1}$$

A local coordinate system is also defined as depicted in Figure 1.1. It is attached to the robot, corresponding the $x$ to the front direction. In both coordinate systems the $z$ axis is going up from the floor, and the robot can only change its orientation by rotating around it an angle $\theta$.

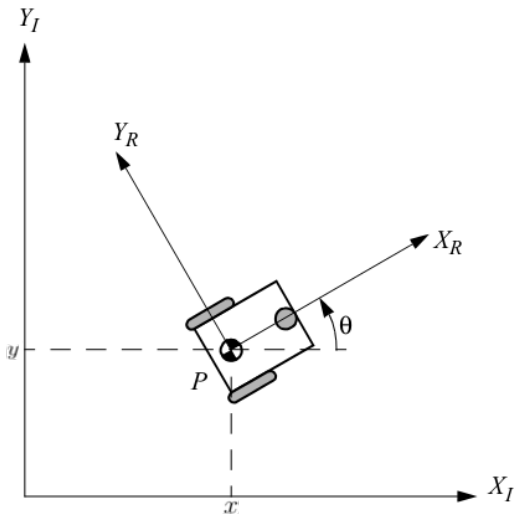


Figure 1.1: Position of the robot referred to the global coordinate system. Figure extracted from [1]

The next lines are a short explanation of the outline. In section 2 the approach followed is described. Moreover, the different tools and algorithms are named and explained. Section 3 discusses the integration of the method described in ROS environment. The experimental results are summed up in section 4, particularly focusing on the differences in the performance of the algorithms used, as well as on the influence of the parameters studied. Finally, sections 5 and 6 concern the final conclusions and the possible further work, respectively.

## 1.1 State of the Art

The problem of global localization has been largely treated through a number of different approaches, which are classifiable by the types of algorithms used.

Several projects address it using place recognition based on visual data. Its main advantage is the high distinctiveness of the different positions in the environment, which helps solve the problem of perceptual aliasing specially in places such as long corridors. However, this technique entails diverse disadvantages, like high sensitivity to illumination changes and long time required for initial setup.

Other methods employ particle filters like AMCL (Adaptive Monte Carlo Localization), which estimate the state of the robot while it is moving and sensing the environment. When solving the problem, the method starts with a uniform random distribution of particles—with each particle representing a possible state—since any point in the space is equally likely. As the robot moves, the information provided by the odometry is used to estimate the current state, which is correlated with the vision sensors to test the suitability of the prediction. Eventually, the particles should converge towards the actual position of the robot. The main downsides of this method are its slowness and the probability of large errors.

Moreover, there are some algorithms based on evolutionary computation concepts. As an example, a non linear evolutionary filter known as the Evolutionary Localization Filter (ELF) is presented in [7]. This algorithm searches for the solution stochastically using an evolutionary search technique, which avoids derivatives. Therefore, it is much more robust in dealing with signals with high noise ratio.

Relating to the matching process, there are many approaches different to the ones studied in this thesis. In [8] a radial sampling filter is presented. The experimental results show that it performs highly efficiently when matching two images, disregarding their rotation angle and translation. [9] introduces a point matching algorithm used for nonrigid shapes based on local neighborhood structures. It can successfully handle nonrigid deformation, noise in point locations, outliers, occlusions, and rotation.

In addition, there are several project that consider more general hypotheses in order to adapt to changing environments. Some of the topics concerned are feature learning and prediction of systematic changes. Moreover, some other groups are working on robots for long-term autonomy, using semantic information for long-term recognition.

# 2 Approach

The problem of global localization deals with the positioning of a robot under global uncertainty. The method presented in this thesis only uses laser measurements provided by the sensor, which are turned into an image to be matched with the global map. One of the main advantages of laser is the avoidance of typical problems of purely visual techniques, such as the influence of lighting changes.

In order to integrate the laser measurements with the usage of Computer Vision algorithm, the representation of the environment is carried out in occupancy-grid maps. These maps allow the transformation of the probability of existence of an obstacle into an image.

Other map types such as feature-based maps or topological maps were discarded. Topological maps represent the world as a network of nodes and arcs: the nodes are distinctive places in the environment and the arcs represent paths between places, as explained in [10], which makes them unsuitable for navigation purposes. Feature-based maps, on the other hand, build the representation of the environment based on landmarks, as described in [11], what constitutes a more complex approach.

The local map created only considers three different cases for every pixel—occupied, free or unknown—and they are represented with black, white or gray color, respectively. Moreover, it is assumed that the robot creates the local map being in its center and orientated as shown in Figure 2.1.

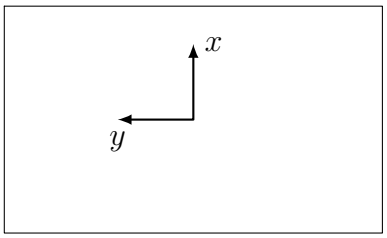


Figure 2.1: Local coordinate system of the robot in the local map

Once the local map has been created, it needs to be matched to the global map in order to find the most probable position of the robot at the given time. When the matching process is finished, the robot should be aware of its real position with a reliability as high as possible.

The flowchart in Figure 2.2 summarizes the process followed when the robot becomes lost.

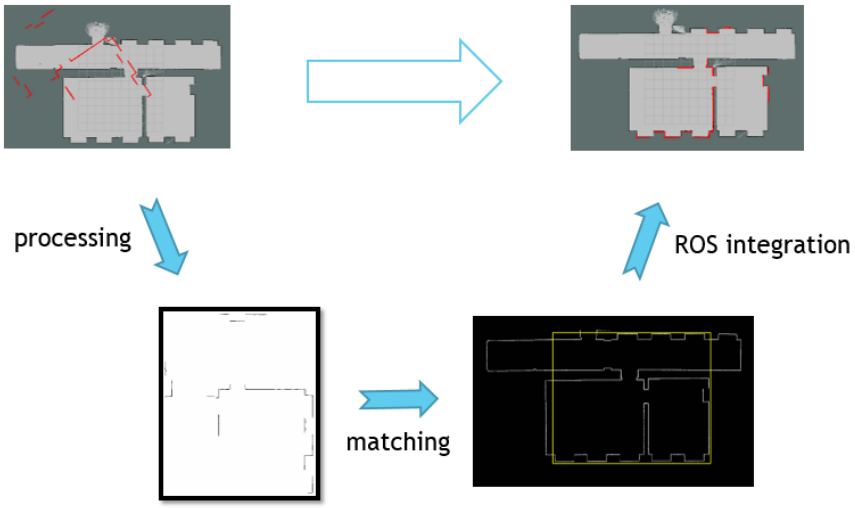


Figure 2.2: Global localization process

## 2.1 Difficulties

The problems with which this method has to deal may be summarized as the following:

- The measurements of the laser may not contain enough information, mainly because the range and field of vision of the laser are limited, offering more information for higher ranges and wider fields of vision.
- There might be a number of places in the environment similar enough to be confused. Therefore, the choice of the right position from a range of possible solutions may not be easy.
- The orientation of the robot is a variable to estimate, which means that the environment seen from the robot's point of view does not necessarily coincide with the global map stored in its memory.
- There can also be some occlusions: obstacles within the range and field of vision of the robot but hidden by other obstacles, making them completely unknown from the robot's perspective.
- The computational resources are limited. As a consequence, methods with smaller needs of computation would fit better for the purpose of global localization.

The difficulty introduced by occlusions is more serious than it may seem. In Figure 2.3, it can be seen that the local map created from laser measurements has an enormous white area outside the obstacles. This would complicate the matching process, since the algorithm would try to find a position in the global map that agreed with all that white space.

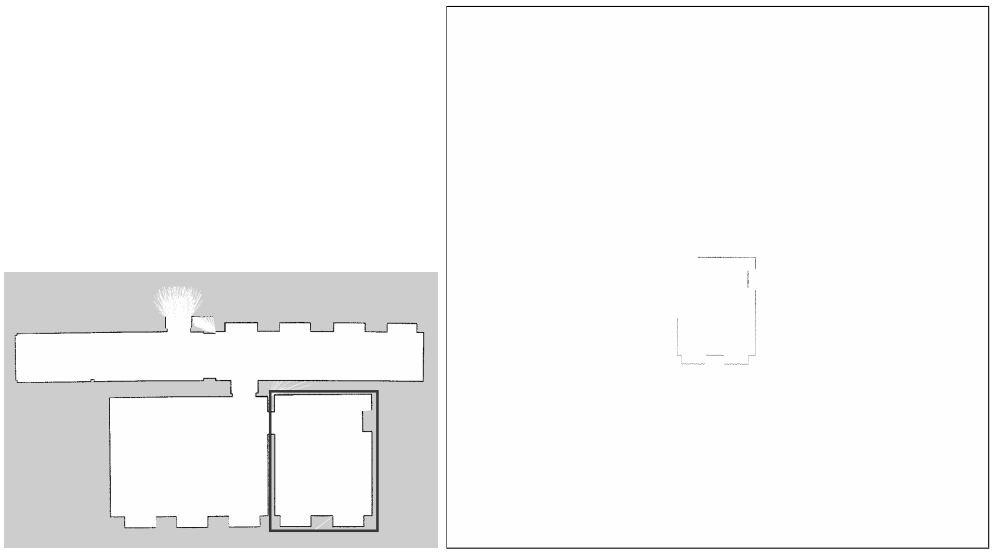

Figure 2.3: Global and local map

## 2.2 Tools used

Different tools have been used in the realization of this project. Most of them are related to the fields of computer vision or robotics, but there are others with diverse purposes. In this section, a brief introduction of the most important ones is made.

### 2.2.1 ROS

The Robot Operating System (ROS) is a flexible framework for writing robot software. It consists on a series of tools and libraries that aim at simplifying the task of developing robot systems across a wide variety of platforms.

The main reason to use ROS is that it is the most commonly used framework when it comes to robot software. Many enterprises and organizations work on ROS, including the Institute of Electrical and Electronics Engineers (IEEE). Moreover, a large number of algorithms are already implemented and available as packages, which facilitates the development of the projects. Another advantage is that it is a modular framework, that is, that nodes can be started independently. As a consequence, the implementations of vast systems is eased when working with other developers.

### 2.2.2 OpenCV

Open Source Computer Vision (OpenCV) is a library of programming functions mainly aimed at real-time computer vision algorithms with many different purposes, such as detecting and recognizing faces, identifying objects or following eye movements. It is used extensively in companies, research groups and by governmental bodies.

It has been chosen for this project because it eases the task of dealing with images. Many functions are already implemented, like the matching methods used, being their performance very high. An additional and important feature is it high speed, what makes it more useful for real-time applications. As a final advantage, it is open source (BSD license).

### 2.2.3 Gazebo

Gazebo is a simulator broadly used in robot software. It enables the rapid test of algorithms and of design of robots accurately and in an efficient way.

The main reason to use it in this project is that its complexity allows the simulation of a wide range of scenarios without complicating its usage. Also, it is compatible with ROS and can be perfectly used for real-time simulation. Moreover, it is open source and vastly used in the robotics environment. Other strengths of Gazebo are its realistic simulation of rigid bodies physics and its high-quality graphics.

### 2.2.4 RViz

RViz is a software program used to visualize different ROS topics. It is included as a ROS package and can be used along with gazebo as a complement for the simulation, representing on the map the data provided by the sensors.

It has been chosen for this project because it is the preferred visualizing tool when working with ROS. Moreover, an useful feature of RViz for this project is its convenient way of changing the robot's estimated pose into a false one, so that the robot has to locate itself again.

### 2.2.5 Biicode

Biicode is a dependency manager that allows the usage of functions without the need of writing them. It is open-source software and counts on a large number of packages.

Biicode has been a very important tool for this project, since it avoids the need of installing OpenCV, as its libraries are already uploaded. Moreover, it eases the task of checking missing dependencies and solving them. Another feature is the automatic building of the project, what saves much time by generating all the auxiliary files itself. Finally, the code can be published and the control of versions can be done by a very simple process, so that other users can take advantage of it.

In fact, the code of the matching process with the different methods studied is already uploaded to biicode[1] as open source.

## 2.3 Preprocessing of local maps

The matching process is the fundamental step in this approach, which means that the local map needs to be preprocessed to suit optimally its actual position in the global map. As a consequence, two characteristics of the local maps created must be faced: the existence of occlusions and the discretization of obstacles.

### 2.3.1 Dealing with occlusions

In order to solve the problem created by the occlusions, two processes can be implemented: alpha compositing and cropping.

**Alpha compositing** A possible solution for the problem created by the occlusions would be to ignore the outer white part by turning it into alpha channel. This process is called alpha compositing, and it consists on combining an image with a background to create the appearance transparency. However, because its application did not improve the results as expected, cropping became necessary.

**Cropping** The cropping process consists on cutting out the outer white parts of the local map, resulting Figure 2.4. This way, it is ensured that the mismatching

[1] `https://www.biicode.com/mcastillon/MapMatching`

caused by the outer blank space is solved. Finally, the space cropped at each side will have to be considered, as it will be shown in Section 2.5.

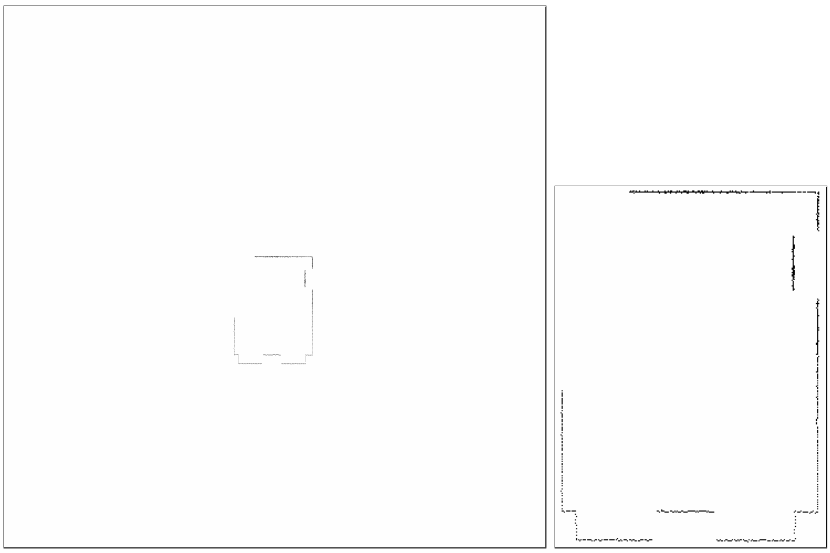

Figure 2.4: Local map before and after cropping

### 2.3.2 Opening

The local maps extracted from the simulation are not directly suitable for the matching. In their creation process, each occupied pixel detected by the laser is painted black in the local map, as explained in Section 3.1, which would create lines of obstacles if the measurements were continuous. However, the field of vision of the robot is discretized, meaning that some pixels that contain an obstacle may be left white. This is more likely to happen in wide spaces like long corridors or large rooms since the long distance to the obstacles increases the influence of the angular discretization.

Therefore, a morphological transformation called opening can be used as an improvement. This function is implemented in OpenCV, and it consists on the erosion of an image followed by a dilation. The result, shown in Figure 2.5, is the joining of points laid in a line despite the distance between them.

Nonetheless, there is a problem attached to opening. As can be seen in Figure 2.6, this process has to be carefully adjusted, since it can join points that are near to each other but do not belong to the same line—usually corners.

For those cases in which the process is difficult to adjust, another approach could consist on applying the opening to the global map as well. Consequently, some areas of the global map would also be excessively joined, so that the bad adjustment would not be such an important flaw. However, some parts of the global map would lose their uniqueness, which could lead to mismatching.

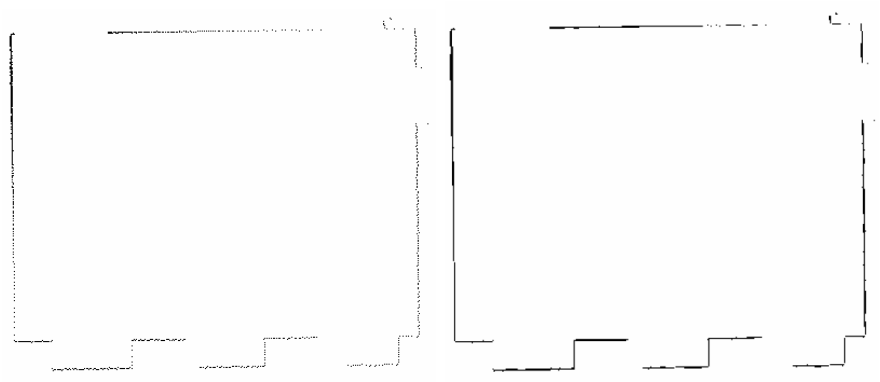

Figure 2.5: Local map before and after applying opening

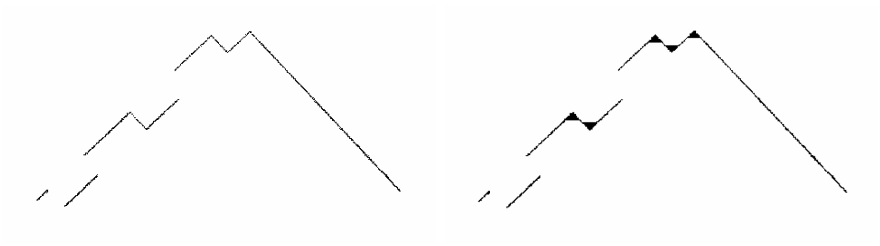

Figure 2.6: Opening process not properly adjusted

## 2.4 Matching methods

This project also aims at comparing two methods broadly used for global location, based on different algorithms. As presented in [12], the first of them is an area-based method called matchTemplate, while the second one is feature-based method named SURF.

Area-based methods perform better when the images have not many prominent details. Their two principal limitations are that they only suit the registration of images which differ only by a translation and that their computational complexity is very high. They are often used because of their easy hardware implementation.

On the other hand, feature-based methods allow to register images of completely different nature and can handle complex between-image distortions. Their main drawback is that the respective features might be hard to detect, specially when working with very simple images. As main characteristic of SURF, it is a specially fast method without any worsening of its performance. Moreover, it can handle scale and perspective changes between images, although it is not needed in this approach.

### 2.4.1 matchTemplate

The process followed by matchTemplate consists on calculating the probability attached to each pixel of the global map to be the solution for the location of the local map, and then chooses the highest one. Therefore, the number of positions checked is

$$N = (h_{global} - h_{local} + 1) \times (w_{global} - w_{local} + 1), \tag{2.1}$$

being $h_{global} \times w_{global}$ the size of the static map and $h_{local} \times w_{local}$ the size of the local map, with $h$ the number of rows and $w$ the number of columns.

The number of positions to be checked will influence the time that will be needed. Accordingly, the bigger the global map is and the smaller the local map is, the more positions have to be checked and more time will be used.

The main downside of matchTemplate is the fact that it can only handle images with a given orientation. Consequently, the rotation of the template has to be done separately, so that the matching process can be repeated for every rotated template. Now, the rotation process is discretized, and the number of positions checked is

$$N = \frac{360^\circ}{\Delta\alpha} \times (h_{global} - h_{local} + 1) \times (w_{global} - w_{local} + 1). \tag{2.2}$$

Then, the increment of the angle of rotation ($\Delta\alpha$) becomes a crucial factor. The smaller it is, the higher success rate the matching will have, since more angles will be checked. However, its decreasing will make the number of possible positions bigger, so it will use more time. A compromise between these two factors has to be found.

Moreover, matchTemplate can work with different correlation methods, and two of them will be compared: CCOEFF_NORMED and SQDIFF_NORMED.

The correlation used by SQDIFF_NORMED is

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'}(T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y'))^2}}. \tag{2.3}$$

On the other hand, the correlation used by CCOEFF_NORMED is

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))^2}{\sqrt{\sum_{x',y'}(T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y'))^2}}, \tag{2.4}$$

where

$$\begin{aligned} T'(x',y') &= T(x',y') - \frac{1}{w \cdot h} \cdot \sum_{x'',y''} T(x'',y'') \\ I'(x+x',y+y') &= I(x+x',y+y') - \frac{1}{w \cdot h} \cdot \sum_{x'',y''} I(x+x'',y+y''). \end{aligned} \tag{2.5}$$

An explanation of these correlations can be found in [13].

First, $R$ is the result matrix that stores the probability of each pixel in the global map to be the location of the upper-left corner of the local map. $T$ refers to the template (local map) and $I$ to image source (global map).

SQDIFF matches the squared difference. Therefore, the smaller the result value stored in $R$ is, the more probable is that pixel to be suitable for the matching, being 0 a perfect match.

On the other hand, CCOEFF matches a template relative to its mean against the image relative to its mean. Consequently, it has a higher computational cost than SQDIFF, but its performance is usually better, except for some concrete applications. Unlike SQDIFF, the most probable pixel stored in $R$ is the one with the highest value.

Finally, both of the methods are normed, which usually provides results with higher success rates.

### 2.4.2 SURF

SURF (Speeded-Up Robust Features) is an algorithm that detects interest points on images. Its main advantages are its computational speed and its robust performance. In [14] and [15], an explanation of the algorithm can be found. Once the features of the images are found, FLANN is used to match both images and RANSAC to find the rigid transformation between them.

SURF is based on the Hessian matrix, $H$, which is a square matrix of second-order partial derivatives. When it is applied to a function $f(x,y)$, the matrix is defined as in equation (2.6).

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \tag{2.6}$$

Using the determinant of this matrix it is easier to discover if a point is a local maximum or minimum. In order to translate it to the work with images, the values of the function $f(x, y)$ are replaced by the pixel intensities of the image $I(x, y)$. Moreover, the second derivative of the image is achieved by its convolution with kernels for the Gaussian derivatives in $x$, $y$ and combined $xy$ direction

Therefore, the new Hessian matrix $H$ is defined as in equation (2.7).

$$H(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix} \tag{2.7}$$

Where $L_{xx}(p, \sigma)$ denotes the convolution of the second order Gaussian derivative $\frac{\partial^2 g(\sigma)}{\partial x^2}$ of the image at point $p(x, y)$ and scale $\sigma$ in the $x$ direction, being $g(\sigma)$ the Gaussian distribution function. It applies similarly for $L_{yy}$ and $L_{xy}$.

There is an important parameter called `minHessian`, which is a threshold that defines which minimum value is needed for a point to become a feature. Therefore, the larger this parameter is, the less points will be chosen as features.

As mentioned before, FLANN-based matcher is used in order to match both images. FLANN (Fast Library for Approximate Nearest Neighbors) is a library that contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. A description of FLANN can be found in [16].

Finally, RANSAC (RANdom Sample And Consensus) is used to find the rotation and translation between two images processed as feature point clouds. RANSAC is broadly used for parameter estimation. Its main strength is that it can handle data affected by noise and corrupted by outliers—points which do not fit the model. It can deal with images that are more than 50 % of outlier contaminated.

## 2.5 Calculating the position of the robot

The position returned by these two methods is the one of the first pixel of the template referred to the first pixel of the image source, starting in the upper-left corner and defining $x$ and $y$ axes as in Figure 2.7. Consequently, the position of the robot has to be calculated, defined as in equation (2.8).

$$\overrightarrow{OR} = \begin{bmatrix} R_x \\ R_y \end{bmatrix} \tag{2.8}$$

The easiest case to calculate the position of the robot is when the rotation angle is 0° and no cropping has been made, as shown in Figure 2.7. Because the position of the robot is assumed to be in the center (see Figure 2.1), it can be calculated as in equation (2.9).

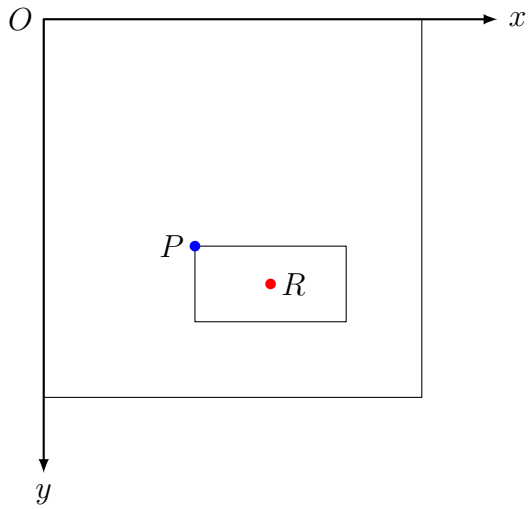


Figure 2.7: Position of the robot in the matched local map

$$\overrightarrow{OR} = \overrightarrow{OP} + \overrightarrow{PR} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} \frac{w_{local}}{2} \\ \frac{h_{local}}{2} \end{bmatrix}, \tag{2.9}$$

where P is the position of the upper left corner of the local map with respect to the reference system of the global map image.

However, if the cropping process explained before has taken place, the robot may not be located at the center of the cropped map, as the number of rows and columns may not be the same as each other. Therefore, the result would be as shown in Figure 2.8, and it can be calculated as in equation (2.10).

$$\overrightarrow{OR} = \overrightarrow{OV} + \overrightarrow{VR} = \overrightarrow{OP} + \overrightarrow{PV} + \overrightarrow{VR} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} -c_{left} \\ -c_{up} \end{bmatrix} + \begin{bmatrix} \frac{w_{local}}{2} \\ \frac{h_{local}}{2} \end{bmatrix}, \tag{2.10}$$

being $c_{up}$ and $c_{left}$ the number of rows or columns cropped from the upper and the left side, respectively.

Now, if the size of the cropped map is noticed, defined as in equation (2.11), equation (2.10) can be rewritten as equation (2.12).

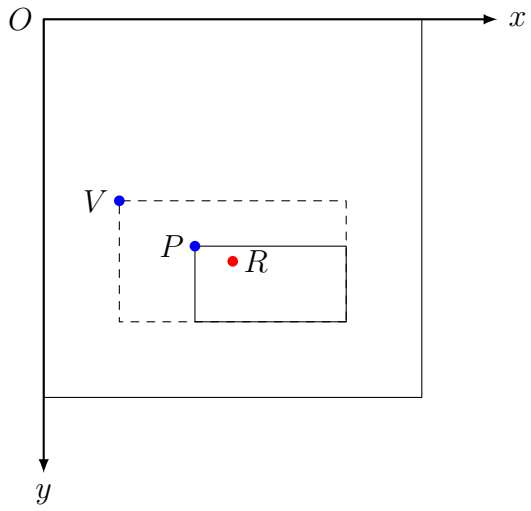


Figure 2.8: Cropped local map, original in dashed line

$$\begin{aligned} w_{cropped} &= w_{local} - c_{right} - c_{left} \\ h_{cropped} &= h_{local} - c_{up} - c_{down} \end{aligned} \tag{2.11}$$

$$\overrightarrow{OR} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} -c_{left} \\ -c_{up} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(w_{cropped} + c_{right} + c_{left}) \\ \frac{1}{2}(h_{cropped} + c_{up} + c_{down}) \end{bmatrix}, \tag{2.12}$$

being $c_{down}$ and $c_{right}$ the number of rows or columns cropped from the down and the right side, respectively.

Therefore,

$$\overrightarrow{OR} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(w_{cropped} + c_{right} - c_{left}) \\ \frac{1}{2}(h_{cropped} - c_{up} + c_{down}) \end{bmatrix}. \tag{2.13}$$

Equivalently, if the variation of the dimensions of the cropped map is used, with

$$\begin{aligned} \Delta w &= c_{right} - c_{left} \\ \Delta h &= c_{down} - c_{up}, \end{aligned} \tag{2.14}$$

equation (2.13) can be rewritten as

$$\overrightarrow{OR} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(w_{cropped} + \Delta w) \\ \frac{1}{2}(h_{cropped} + \Delta h) \end{bmatrix}. \tag{2.15}$$

Finally, if the cropping process has been made and the local map did not have the same orientation as the global map, the number of rows and columns cropped from each side will also rotate. Therefore, the angle of rotation has to be taken into account as shown in equation (2.16).

$$\overrightarrow{OR} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(w'_{cropped} + (\Delta w)') \\ \frac{1}{2}(h'_{cropped} + (\Delta h)') \end{bmatrix}, \tag{2.16}$$

where

$$\begin{aligned} w'_{cropped} &= w_{cropped} \cdot |\cos\alpha| + h_{cropped} \cdot |\sin\alpha| \\ h'_{cropped} &= h_{cropped} \cdot |\cos\alpha| + w_{cropped} \cdot |\sin\alpha| \\ (\Delta w)' &= (c_{right} - c_{left})\cos\alpha + (c_{down} - c_{up})\sin\alpha \\ (\Delta h)' &= (c_{down} - c_{up})\sin\alpha + (c_{left} - c_{right})\sin\alpha. \end{aligned} \tag{2.17}$$

Therefore, for the general case, the equation (2.18) can be applied.

$$\begin{bmatrix} R_x \\ R_y \end{bmatrix} = \begin{bmatrix} P_x + \frac{1}{2}[w_{cropped} \cdot |\cos\alpha| + h_{cropped} \cdot |\sin\alpha| + (c_{right} - c_{left})\cos\alpha + \\ +(c_{down} - c_{up})\sin\alpha] \\ P_y + \frac{1}{2}h_{cropped} \cdot |\cos\alpha| + w_{cropped} \cdot |\sin\alpha| + (c_{down} - c_{up})\sin\alpha + \\ +(c_{left} - c_{right}]\sin\alpha) \end{bmatrix} \tag{2.18}$$

# 3 Integration with ROS

The next step in this project is to allow the usage of the methods mentioned before in real-time simulation, which can be achieved using ROS. For doing so, a ROS node needs to be created, in which a subscriber gets the information sent by the sensor, and a publisher provides the calculated position once the whole process has succeeded.

Moreover, it is worth mentioning that the `actionlib` ROS package is used instead of services. Services are broadly used to send a request to a node to perform some task, and also receive a reply to the request. However, `actionlib` becomes useful when the service takes a long time to execute, since the user might want the ability to cancel the request during execution or get periodic feedback about how the request is progressing. Consequently, as the global location may be time consuming, it should have the capacity to be preempted if necessary.

Apart from the node, a client attached to it is used to initialize the global location process when needed. When the location process is used in real-time simulation, the different parts of ROS are connected as shown in Figure 3.1.

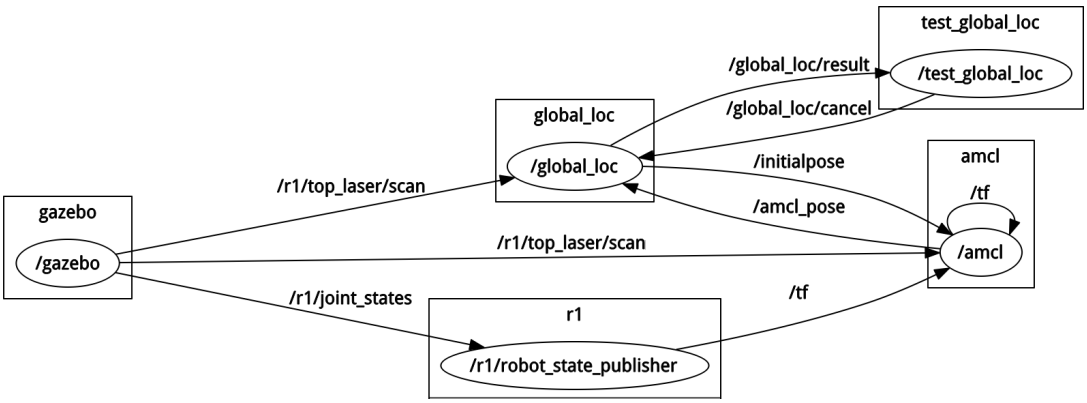


Figure 3.1: ROS linking

Gazebo publishes the laser data provided by the sensor in the simulation to the `amcl` node, along with the transforms of the physical model of the robot (as in Figure A.2), so that the position can be estimated in normal situations.

However, when the robot becomes lost, the node created (`global_loc`) is called by a client (`test_global_loc`) and the recovery process starts. The node receives the laser data and creates the local map in order to be matched with the global map. Once it is finished, the calculated position is published on the topic `initialpose`, to initialize the particle filter.

The maps used by ROS need to be stored in a pair of files: an image file and a `YAML` file. The image file encodes the occupancy data, usually in gray-scale, by giving darker colors to the pixels with higher probability to be occupied. The

thresholds in the `YAML` file are used to divide the three categories—free, occupied or unknown. The `YAML` file includes the map meta-data, namely the name of the map, its resolution, its origin and the thresholds.

## 3.1 ROS node

In the ROS node created, the local map is conceived as an occupancy grid map, which consists on a number of evenly divided cells, having each of them the probability of the presence of an obstacle attached, which are stored as an array.

The type of message used by occupancy grid belongs to navigation messages, a family used to interact with the navigation stack. Occupancy grid consists on three fields: `header`, `info` and `data`.

The `header` contains information used to communicate timestamped data in a particular coordinate frame. The field `info` carries the size of the map (`width` and `height`), its `resolution` and its `origin`. Finally, `data` contains the occupancy probabilities for each pixel in row-major order starting with $(0, 0)$, being in the range $[0, 100]$.

Furthermore, the package `occupancy_grid_utils` is used to deal with objects created as occupancy grids.

The local map has the same resolution as the global map and its origin is in the center, as shown in Figure 2.1. This information is taken from the `YAML` file attached to the image of the map, and stored in the corresponding fields mentioned above. Both the width and length are double of the maximum range of the sensor, so that any obstacle that may be detected can be represented on the map.

The node is subscribed to the measurements of the sensor, as stated before. The data provided consists on the discretization of the field of vision, so that each angle $\alpha_i$ has a distance attached $r_i$, as shown in Figure 3.2. The angles are defined as in the figure. Also, the number of measurements will be

$$N = \frac{\alpha_{max} - \alpha_{min}}{\Delta\alpha}. \tag{3.1}$$

With the purpose of transforming this data into their positions referred to the static coordinate system attached to the local map, the equation (3.2) is used.

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \cdot w_{local} - r_i \cdot \sin\alpha_i \\ \frac{1}{2} \cdot h_{local} + r_i \cdot \cos\alpha_i \end{bmatrix} \tag{3.2}$$

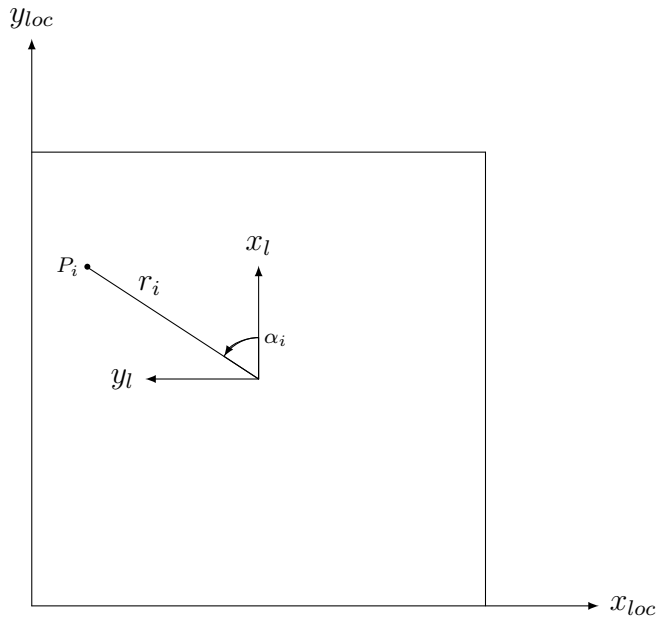


Figure 3.2: Laser measurements into the coordinate system attached to the local map

Afterwards, the positions of the obstacles, that are stored in a matrix with N elements, being

$$N = h_{local} \times w_{local} = 4 \cdot max_{range}, \tag{3.3}$$

need to be saved in an array of the same size, which can be used by the occupancy-grid map. This array is built for growing row number, so that the index of the array, $k$, is defined as

$$k = j \times w_{local} + i, \tag{3.4}$$

where $i$ is the column number and $j$, the row number.

Then, every index that contains an occupied pixel is given a value of `OCCUPIED` (100), being the rest `FREE` (0).

Once this process is completed, an OpenCV `Mat` image is created painting in black those pixels that correspond to an occupied position, leaving the rest white. The class `Mat` is broadly used in OpenCV to deal with images. Every image is represented as a matrix with a value associated to each pixel, which is often a vector with three elements—when RGB color scale is used—called layers. The values of these layers determine the color of the pixel.

After all these processes, the local map is matched with the global map, using the methods described before. The result is the position of the robot referred to the global coordinate system.

Moreover, a covariance matrix is associated to the position calculated. The element of this matrix in the $(i, j)$ position is the covariance between the $i^{th}$ and $j^{th}$ elements of the vector of variables, which indicates the strength of the correlation between them. As the covariance of the $i^{th}$ variable with the $j^{th}$ one is the same thing as the covariance of the $j^{th}$ variable with the $i^{th}$ one, the covariance matrix is symmetric. Moreover, each element of the principal diagonal is the variance of the respective variable. In this project, a fixed uncertainty was assigned to the variance of the variables, considering an absence of interdependence between them.

Conclusively, the position with the covariance stamped is published once the global localization has finished. The result of the relocalization is shown in Figures 3.3 and 3.4 for two different situations.

Figure 3.3: Relocalization of the robot in RViz. Red dots represent the laser data

Figure 3.4: Relocalization of the robot in RViz. Red dots represent the laser data

# 4 Results

The different methods have been tested and compared. For doing so, it is important to focus not only on the success of the position estimated, but also on the time that the relocation process takes.

The global map used for the experiments was created using SLAM Gmapping, which is implemented as a ROS node that permits the creation of a 2-D occupancy grid map from laser and pose data collected by a mobile robot. Further explanation of the algorithm can be found in B.

First, the algorithms have been tested for locals maps manually extracted from the global one. Afterwards, once the problems that arose were solved, local maps created from simulated scan measurements were used.

As it will be proved throughout this section, the maps edited by hand offer a much higher success rate, as they are exactly as a part of the global one. However, they cannot be used for the real goal of the thesis, only for academical purposes. On the other, the local maps created in simulation can be broadly used in real applications. Nonetheless, the results that can be obtained are not as good, since the two maps to be matched cannot coincide completely.

The tests were run on a computer provided with an Intel® Core™ i5-760 CPU of frequency 2.80 GHz.

Throughout this section, all the graphs that compare the three methods use the same color code:

CCOEFF_NORMED / SQDIFF_NORMED / SURF

## 4.1 Local maps extracted from the global map

The first tests run to compare these algorithms used parts of the global map as local maps. The factors to study were the success rate and the time needed by each method. Furthermore, there are some parameters that may also influence the success rate and the time needed, like the angle $\Delta\alpha$ used to rotate the local map when matching it, and the size of the local map once it has been cropped.

The tests were run on a series of 18 local maps, varied enough in order to cover as many cases as possible. They had different size and angle of rotation. It is more difficult to find the right solution when there is a number of positions that can satisfy the matching with similar probability. This usually happens in long corridors and corners, due to the lack of characteristic features.

A successful matching was considered when the area in the global map corresponding to the local map was properly represented in the result window.

As an example, Figures 4.1 and 4.2 show the matching process performed by matchTemplate with $\Delta\alpha = 15°$ for the same local map, but with different rotation angles.

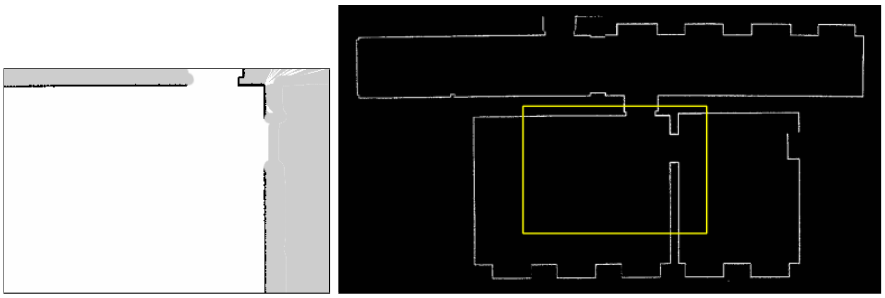

Figure 4.1: Local map rotated 0°

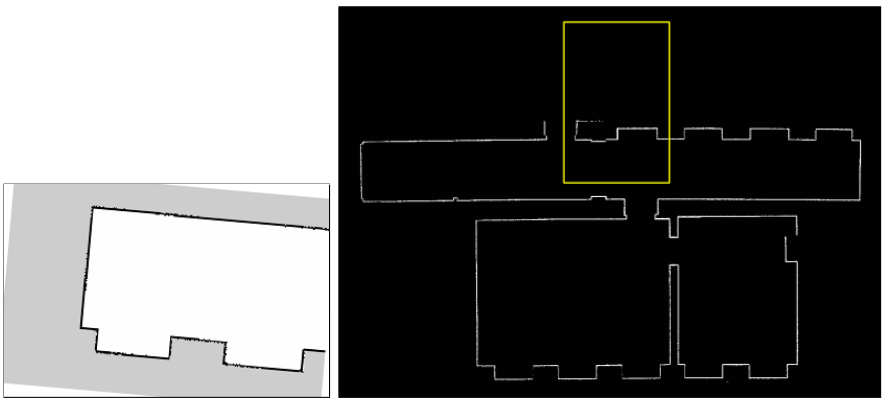

Figure 4.2: Local map rotated 175°

It can be seen that, as the rotation angle of the local map is not checked by matchTemplate, the matching process does not success.

Concerning SURF, Figures 4.3 and 4.4 show successful and unsuccessful matching processes, respectively.

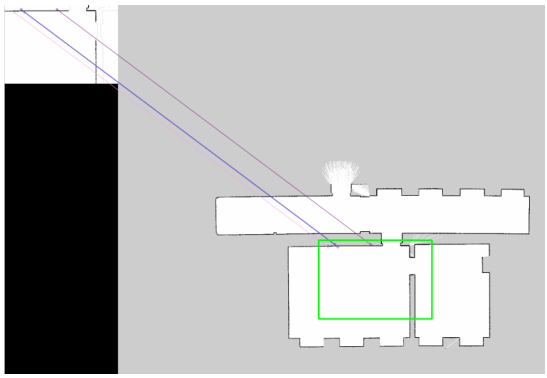

Figure 4.3: Successful matching by SURF

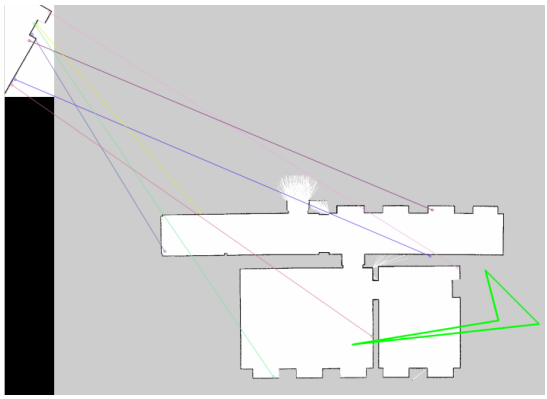

Figure 4.4: Unsuccessful matching by SURF

### 4.1.1 Performance by method

First, the three methods were tested for every local map, so that their performances could be compared. The success rate and the average time needed by each method is shown in Figure 4.5.

The results given by matchTemplate were much more successful than the ones from SURF. Moreover, a difference between the two correlations can be appreciated, as CCOEFF_NORMED had a slightly higher success rate than SQDIFF_NORMED.

On the other hand, SURF is proved to be much faster than matchTemplate, taking always less than 5 seconds to return the result. Both correlations of matchTemplate need between 40 and 65 seconds, averaging around 50 seconds, with SQDIFF_NORMED a little faster.

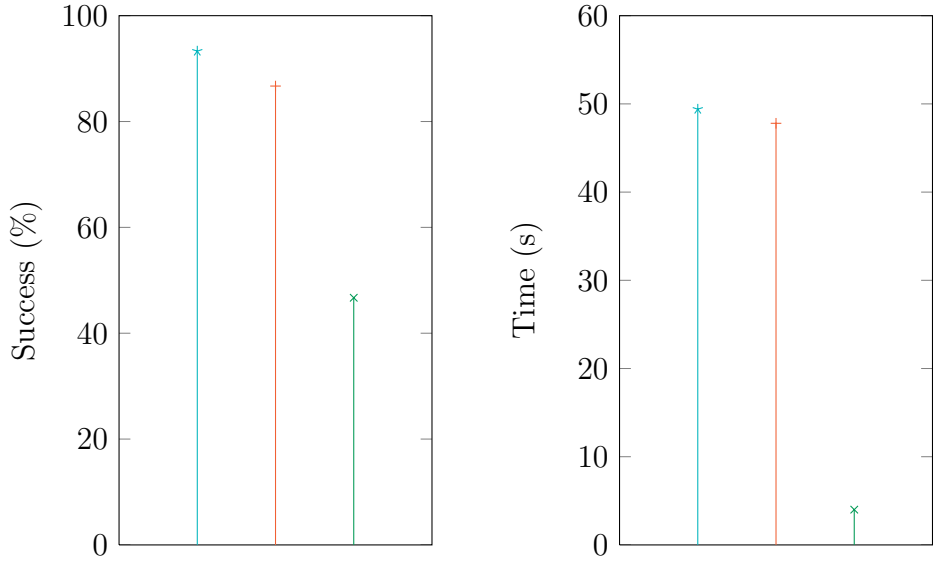


Figure 4.5: Success rate and time used by each method

### 4.1.2 Influence of the angle increment in matchTemplate

The success rate of each method and its average time are compared for two values of $\Delta\alpha$ in Figure 4.6.

The graph shows that both methods have a worse success rate for the largest increment ($\Delta\alpha = 15°$), as expected. This is because the position cannot be properly calculated for those local maps that are rotated an angle not multiple of 15. Of course, the worsening rate depends on the number of local maps that are rotated an angle not multiple of 15.

However, the time needed by matchTemplate when $\Delta\alpha = 15°$ is much smaller than for $\Delta\alpha = 5°$, no matter the correlation with averages of less than 20 seconds. It was also expected, since the slowest step in the algorithm—the rotation of the local map—is the one that determines the time used. Therefore, and given that the number of rotations is reduced three times, it is logical that the time needed also decreases to around one third.

### 4.1.3 Influence of the size of the local map

The last factor to study in this first part of the results is the size of the local map once it has been cropped. Figure 4.7 shows a scatter plot containing the time used by each method for each map—depending on its size. The respective trend lines of each method are also drawn.

This plot shows that the size of the local map influences the time used by

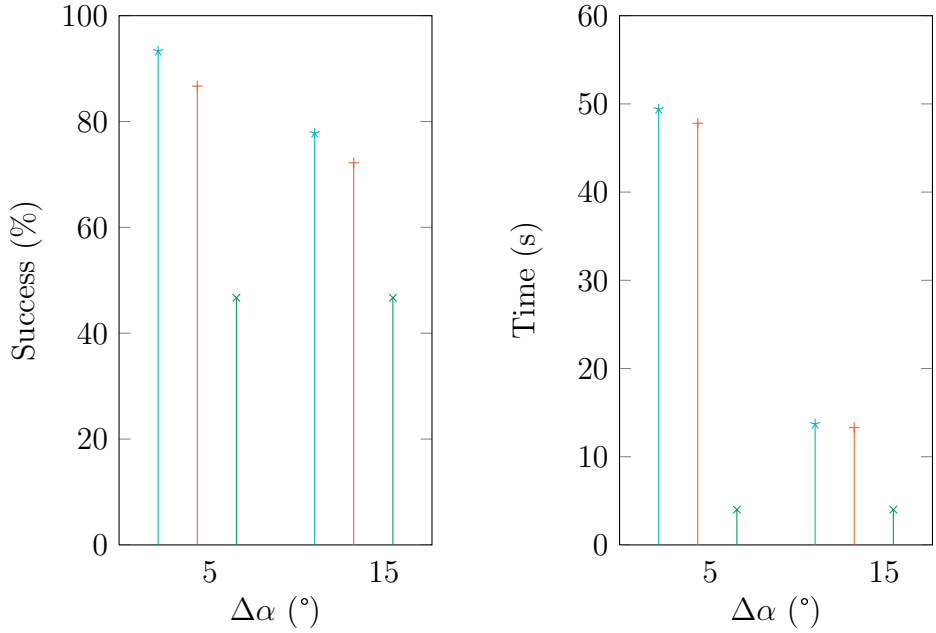


Figure 4.6: Success rate and average time for different $\Delta\alpha$

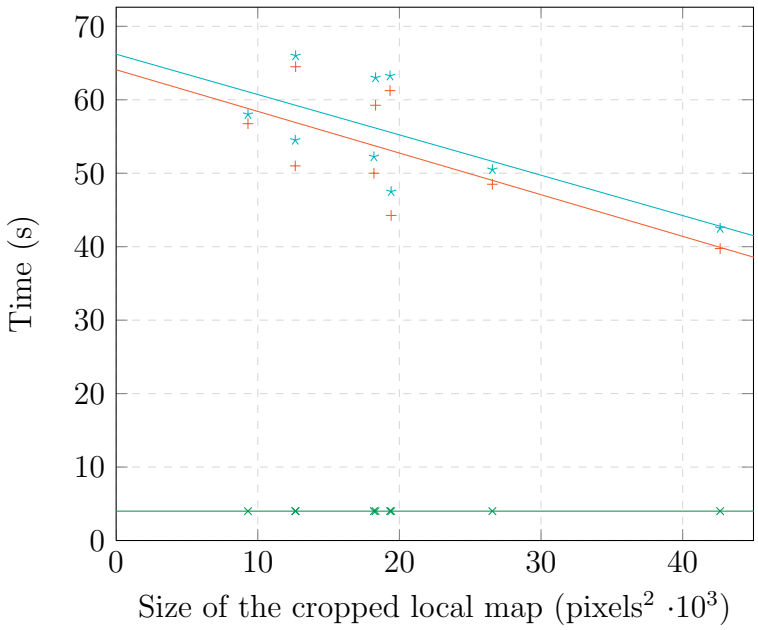


Figure 4.7: Influence of the size of the local map

matchTemplate in a slight way. For both correlations, the process is a little faster for larger maps. On the other hand, SURF is not affected by the size of the local map at all. This result could be predicted if noticing equation (2.1).

### 4.1.4 Conclusions

The conclusions concerning success rate that can be extracted from these results are that matchTemplate performs much better than SURF, without a big difference between both correlations. However, when it comes to time needed, matchTemplate is significantly slower than SURF.

Concerning the probability of success in the result given by matchTemplate, thresholds could be defined for CCOEFF_NORMED and SQDIFF_NORMED. That way, a result with a higher score than 0.70 for CCOEFF_NORMED or smaller than 0.60 for SQDIFF_NORMED could be considered successful with very low probability of false positives. Solutions with worse scores may also be successful, so there would be a higher number of false negatives. However, false negatives are preferred over false positives.

With regard to the angle increment in matchTemplate, it has been observed that larger $\Delta\alpha$ entails shorter times but also worse performance.

Relating to the size of the local map (not taking outer white area into account), it was proved that matchTemplate is a little faster for bigger maps, whereas SURF remains unchanged. The explanation of this dependency can be found when plotting the number of positions that have to be checked versus the size of the local map, as in Figure 4.8. It shows that for the range of sizes normally used in these tests (both length and width between 100 and 300 pixels), the dependency can be assumed to be linear with a coefficient of $R^2 \approx 0.93$.

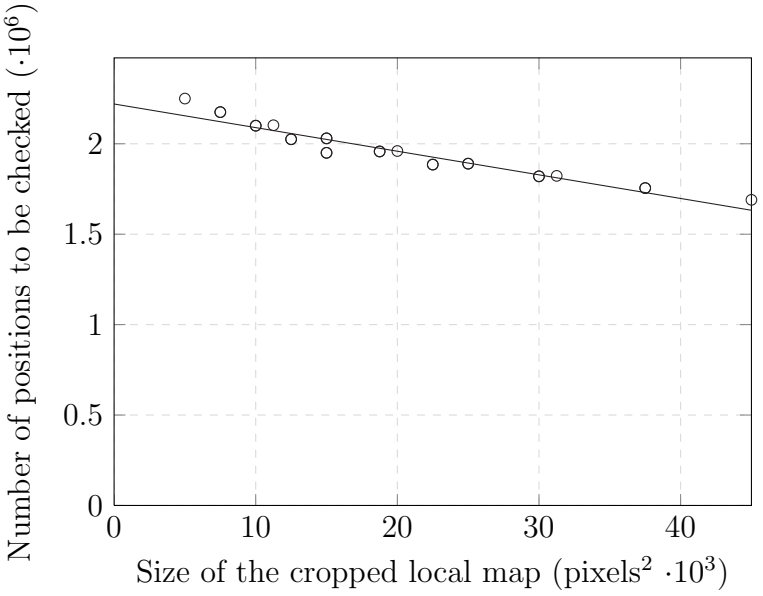


Figure 4.8: Number of positions checked in matchTemplate for different local map sizes

## 4.2 Simulation

The second part of the tests were run on a series of 23 local maps created in simulation and processed as explained in previous sections. The positions used for the experiments represent the different situations that the robot may face when solving its global localization, such as diversity of angles of rotation and lack of distinctive information.

In order to consider the result of the calculation as successful, the error in $x$ and $y$ directions had to be both smaller than 0.2 m, whereas the error in $\theta$ lesser than 5°.

With the purpose of studying the effect of the different parameters, they have been changed individually, remaining the rest as in the basic case (field of vision of 180°, range of 20 m and $\Delta\alpha = 5°$).

As an example, Figures 4.10 and 4.11 shows the local map (before and after preprocessing) created in the situation of Figure 4.9.

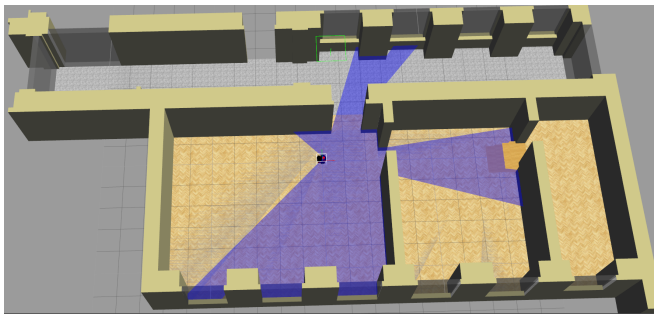

Figure 4.9: Robot calculating its position in real-time simulation

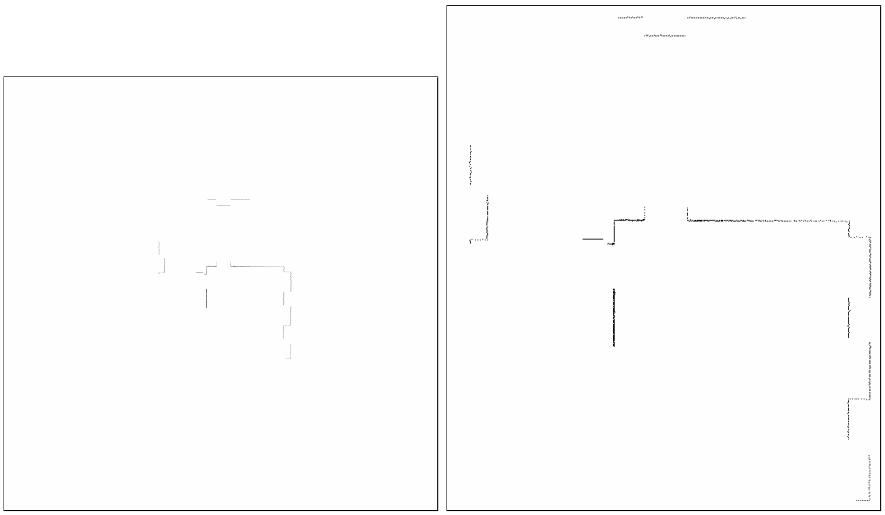

Figure 4.10: Local map created in simulation before and after cropping

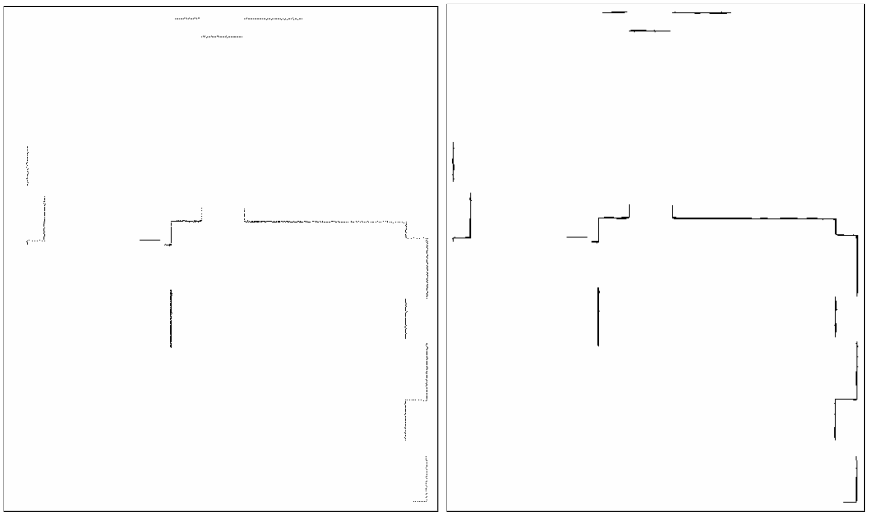

Figure 4.11: Local map created in simulation before and after opening

### 4.2.1 Performance by method

First, the results provided by each method were compared for a standard case: field of vision of 180°, range of 20 m and $\Delta\alpha = 5°$. Their success rate and the average time used is shown in Figure 4.12.

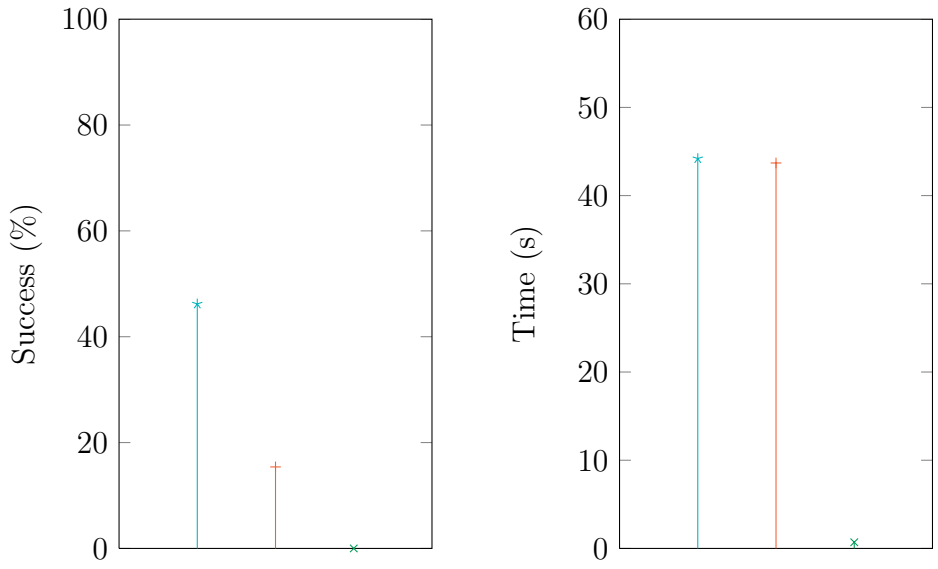


Figure 4.12: Success rate and average time used by each method

matchTemplate has a low success rate, particularly SQDIFF_NORMED, and they both perform much worse than in the previous tests. SURF did not succeed for any position.

Relating to the time needed, matchTemplate has a similar average to the first series of experiments, around 45 seconds. On the other hand, SURF averages

less than 1 second. The reason is that not enough feature points were found in most of the cases, so the matching process did not even start.

### 4.2.2 Influence of the characteristics of the laser sensor

**Field of vision** In relation to the field of vision of the sensor, two more values were tested (60° and 270°) in order to study its influence on the results. In Figure 4.13 and 4.14 the success rate and the average time, respectively, are shown.

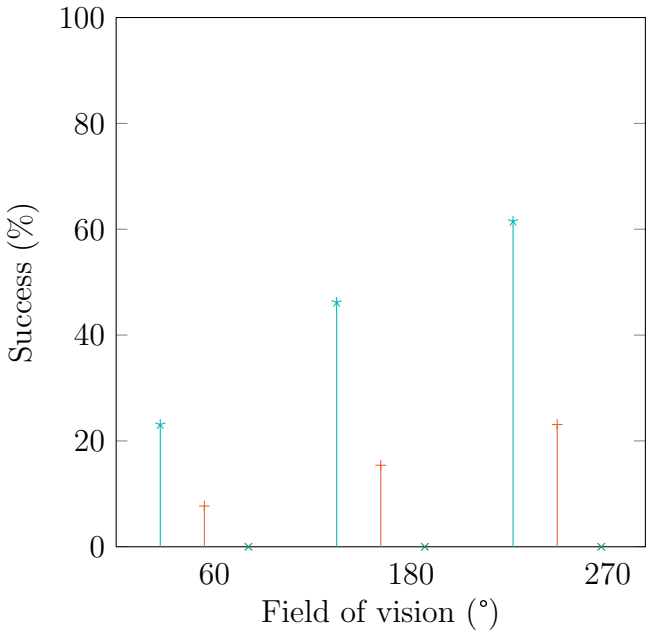


Figure 4.13: Success rate for different fields of vision

Concerning the success rate, matchTemplate improves its performance for wider fields of vision, as it was expected, being the rate of CCOEFF_NORMED higher than the one of SQDIFF_NORMED. SURF did not succeed for any position.

When it comes to the time used, little difference can be noticed. However, the wider the field of vision is, the larger the local map gets. Therefore, the time slightly decreases, as explained before. The average time of SURF is higher for 270° because most of the cases had enough feature points, so the matching process was done. However, its average is around 3 seconds, which is much faster than matchTemplate.

**Range** Relating to the range of the sensor, a comparison with a lower value was made to see how the results change. They are shown in Figure 4.15.

The success rate for CCOEFF_NORMED decreases for 4 m, as it was expected.

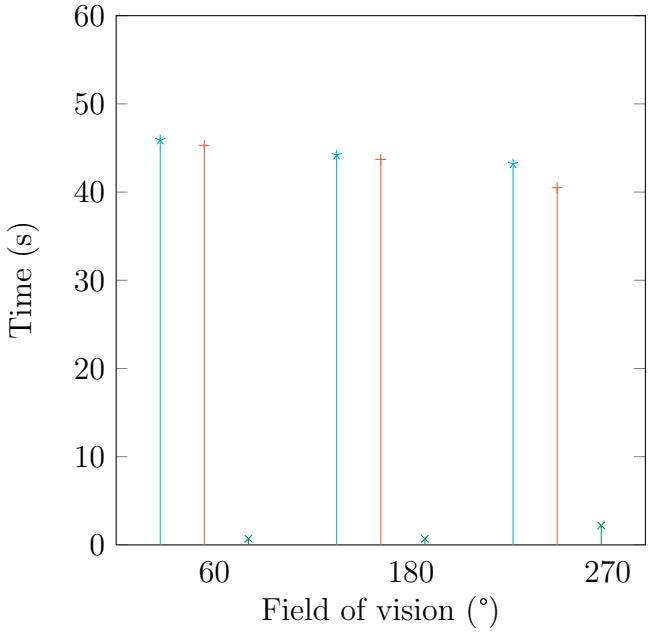


Figure 4.14: Average time used by different fields of vision

However, SQDIFF_NORMED remains with the same percentage. The average time used by matchTemplate increases for a range of 4 m. It was expected, as shorter ranges mean smaller local maps.

On the other hand, SURF cannot find enough features, what entails that no successful positioning is achieved.

### 4.2.3 Influence of the angle increment in matchTemplate

The success rate of each method and its average time are compared for two values of $\Delta\alpha$ in Figure 4.16

As it was expected, the success rates of matchTemplate get worse for larger $\Delta\alpha$, but the average time used is smaller. Precisely, the time for $\Delta\alpha = 15°$ is around three times shorter than for $\Delta\alpha = 5°$. Of course, the worsening rate depends on the number of positions that are rotated an angle not multiple of 15.

The performance of SURF is obviously not influenced by the change of $\Delta\alpha$.

### 4.2.4 Conclusions

The results of the test in simulation reveal that the suitability of this method depends largely on the purpose and the sensor used. If a commercial sensor were

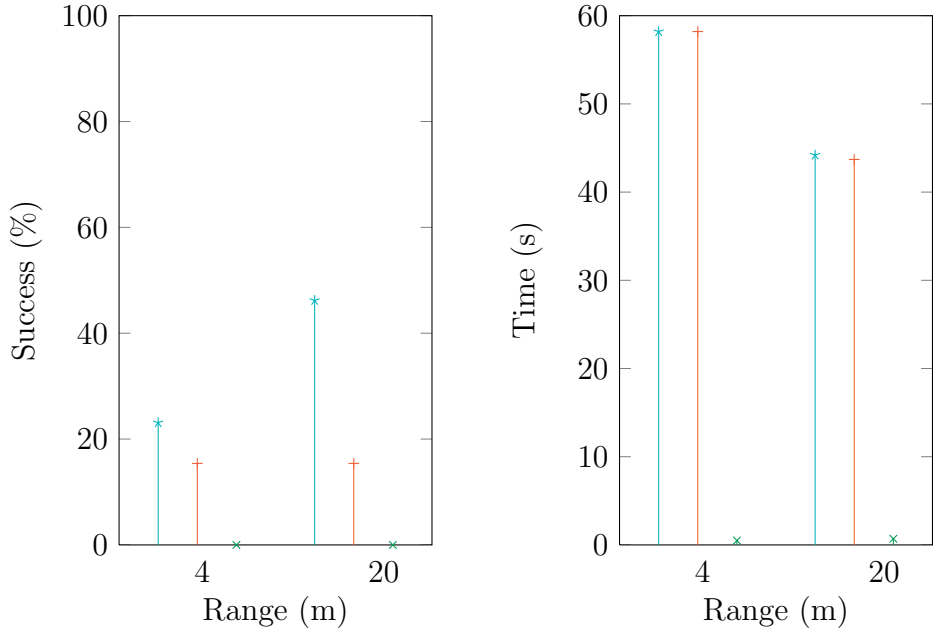


Figure 4.15: Success rate and average time for different ranges

used, such as Kinect, the characteristics would be similar to 60° and 4 m. With these parameters, the results provided may not be positive enough. However, if the sensor could provide 270° and 20 m, a success rate around 60% could be achieved.

Moreover, $\Delta\alpha$ could be adjusted to find an optimum between success rate and time used, which would vary for different applications.

The performance of SURF is extremely poor, which should be further investigated. If its low average time were required, some changes could be implemented on the creation of the local maps to try to increase its success rate.

For instance, Figure 4.17 shows an unsuccessful matching process, in which many feature points are found but not properly matched.

Concerning the probability of success in the result given by matchTemplate, thresholds could not be found in order to distinguish reliably between successful and unsuccessful solutions. This is a problematic fact, that entails further checking of the results before considering them proper.

As an example, the global localization process with matchTemplate using CCOEFF_NORMED for the positions shown in Figure 4.18. For the first position, the result was unsuccessful, with a score of 0.4888. On the other hand, the method succeeded for the second position, with a score of 0.0943.

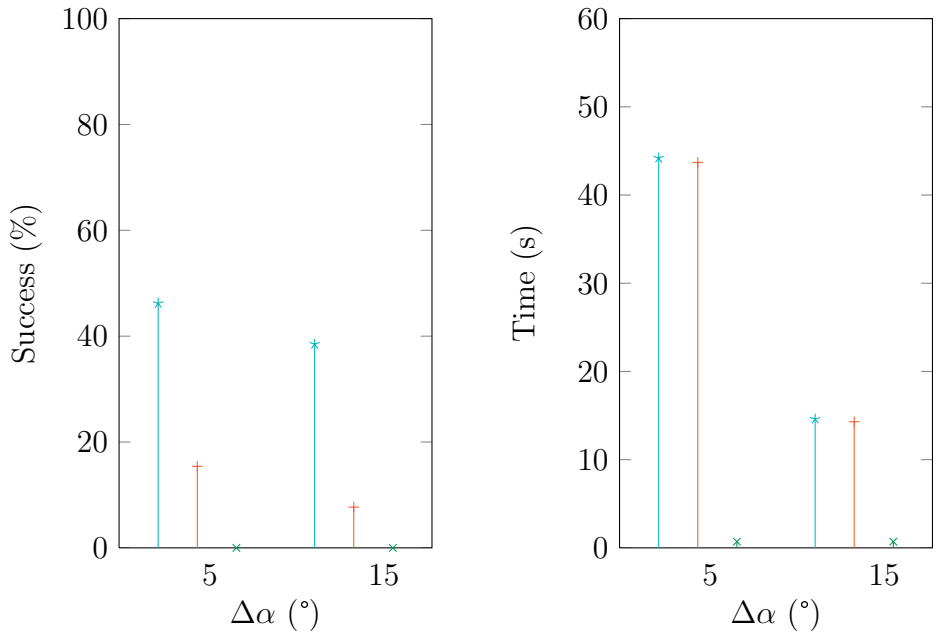


Figure 4.16: Success rate and average time for by different angle increments

Figure 4.17: Unsuccessful matching by SURF

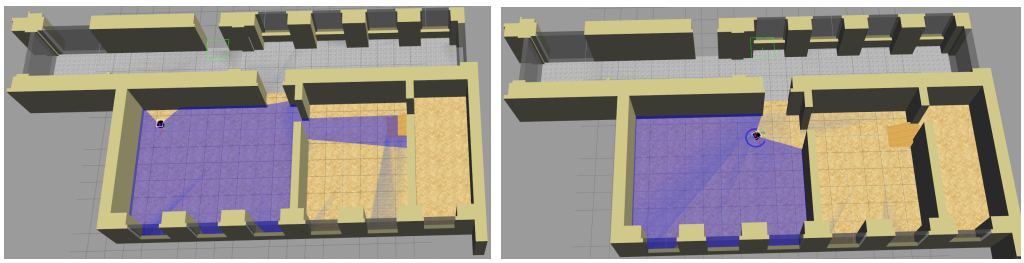

Figure 4.18: Two different situations simulated in gazebo

# 5 Conclusions

Two methods have been compared—matchTemplate, which is area-based, and SURF, feature-based. Moreover, two different correlation of matchTemplate have been tested: CCOEFF_NORMED and SQDIFF_NORMED.

Given the results obtained, the performance of matchTemplate has a medium success rate, but may be too slow for some real applications. On the other hand, SURF has not given good results in the tests performed, since more feature points are needed when using the raw local map images.

Regarding the failure of SURF, it is mainly due to its inability to find enough feature points and match them properly. A reason for it can be that the features of these maps at a local level are not characteristic enough. It is worth mentioning that it may not be the most suitable method for the images treated in this thesis, as they also lack of small obstacles, which are easier to match. Moreover, its main advantages—scale and perspective invariance—are not needed for this purpose.

However, feature-based methods count on positive characteristics, such as their fast performances. Therefore, in order to take advantage of them, some changes in the method and in the image processing could be implemented to try to raise its success rate. Moreover, other descriptors different to SURF could be implemented with the aim of finding a more suitable one.

In relation to the different correlations of matchTemplate studied, the average time needed by both of them are very similar. Nonetheless, when it comes to success rates, CCOEFF_NORMED performs usually better than SQDIFF_NORMED, sometimes significantly.

Concerning the characteristics of the laser sensor, which is crucial in this approach, the results are always better for wider fields of vision and longer ranges, as expected. When working with matchTemplate, $\Delta\alpha$ becomes a very important factor, which, depending on the application aimed, can be adjusted for a higher success rate or for shorter execution times.

Relating to the probability score given by matchTemplate to every location in the global map, a threshold cannot be found to distinguish between successful and unsuccessful matches. Therefore, the likelihood of errors when estimating the reliability of the solution is high enough to consider further checking methods.

Finally, it has also been observed in the realization of this project that a dependency manager saves time during the configuration and implementation phases.

# 6 Future Work

In applications using narrow fields of vision, the local maps created with the current method may not contain enough information. Therefore, an interesting process for creating it would be a 360° scan, that would provide further information, leading to better performances. Nonetheless, if the field of vision were wide enough, an easy improvement would be to locate the robot twice for every position—once with the initial orientation, and a second one after having turned 180°. Afterwards, if both processes returned the same position (with a 180° difference, considering an error margin), the probability of success would be significantly higher.

Moreover, a problem observed in the creation of the local maps that included long corridors was the large distance between points representing the same obstacles. A possible solution would be to apply a stronger opening process, but that might lead to the joining of near corners. Consequently, a cropping algorithm that considered the density of occupied positions could be implemented as further development.

Another possible approach would be the usage of the method analyzed in this project when the solution can be considered successful with very high reliability—a suitable method to assess this reliability will be necessary. For the rest of the cases, AMCL in global localization mode could be used. This way, the problem concerning the errors in the localization could be avoided.

It would also be useful the incorporation of the probability given by matchTemplate to each position of the global map to be the actual location of the robot. Therefore, the covariance matrix attached to every possible solution would depend on that score, what would be a more reliable result than only returning the highest-rated position, which could be used to initialize a particle filter.

As a final step, once the algorithm has been proved to perform properly, the new improvements can be added to the already released code.

# A Model of the Robot

The robot used for the simulations was the Pioneer 3-DX, which is vastly used for research purposes due to its versatility, reliability and durability.

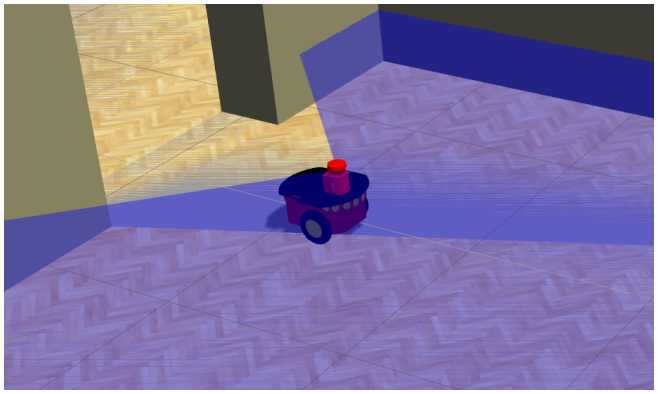

Figure A.1: Robot Pioneer 3-DX, used for the simulations

Moreover, the transformation tree of the robot is shown in Figure A.2. This information allows `amcl` to find the right relationships between the laser measurements provided by the sensor and the rest of the parts of the robot, so that it can be located properly in the environment.

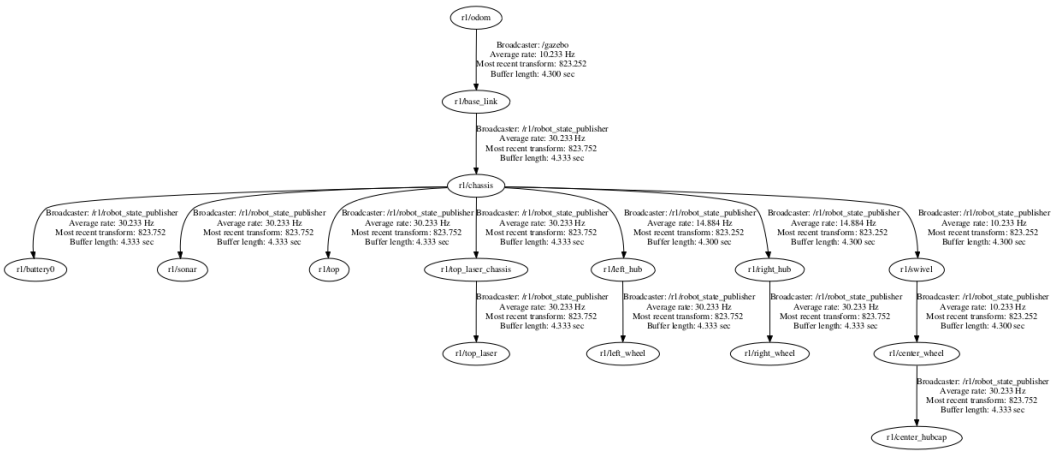


Figure A.2: Scheme of the physical model of the robot

# B SLAM

Simultaneous Localization and Mapping (SLAM) is one of the most widely researched topics in Robotics. It is useful for building and updating maps within unknown environments, while the robot keeps the information about its location. The main problems that SLAM faces are:

- Cumulative errors induced by proprioceptive sensors when estimating the movement of the robot.
- Difficulty on determining whether sensor measurements taken at different points in time correspond to the same object in the world.
- Existence of non-static worlds, whose features may change over time.

In [17] a comparison of the SLAM methods already implemented in ROS is developed. One of the best-known SLAM algorithms is Gmapping, which is also used in this project in the creation of the global map. It is a laser-based SLAM approach with a very good performance when there is a high number of particles available. Another algorithm is HectorSLAM, that combines a 2D SLAM system based on robust scan matching and 3D navigation technique using an inertial sensing system rather than odometric sensors. As a downside, it needs high scan rates in order to achieve good results. Moreover, KartoSLAM is a graph-based SLAM approach, in which each node represents a pose of the robot along its trajectory and a set of sensor measurements. Different nodes are connected by arcs, that represent the motion between successive poses.

When comparing these SLAM algorithms, the conclusion in [17] is that Gmapping is highly robust, with very low error and CPU load. On the other hand KartoSLAM and HectorSLAM also provided good results, but this last one may get better one if equipped with specific hardware.

# References


[1] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[2] Stephen Se, David G Lowe, and James J Little. Vision-based global localization and mapping for mobile robots. *Robotics, IEEE Transactions on*, 21(3):364–375, 2005.

[3] Patric Jensfelt and Steen Kristensen. Active global localization for a mobile robot using multiple hypothesis tracking. *Robotics and Automation, IEEE Transactions on*, 17(5):748–760, 2001.

[4] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999:343–349, 1999.

[5] Tom Duckett and Ulrich Nehmzow. Mobile robot self-localisation using occupancy histograms and a mixture of gaussian location hypotheses. *Robotics and Autonomous Systems*, 34(2):117–129, 2001.

[6] Tai-hoon Kim, Hojjat Adeli, Hyun-seob Cho, Osvaldo Gervasi, Stephen S Yau, Byeong-Ho Kang, and Javier Garcia Villalba. *Grid and Distributed Computing: International Conferences, GDC 2011, Held as Part of the Future Generation Information Technology Conference, FGIT 2011, Jeju Island, Korea, December 8-10, 2011. Proceedings*, volume 261. Springer Science & Business Media, 2011.

[7] Luis Moreno, Santiago Garrido, and Dolores Blanco. Mobile robot global localization using an evolutionary map filter. *Journal of Global Optimization*, 37(3):381–403, 2007.

[8] Hae Yong Kim and Sidnei Alves De Araújo. Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast. In *Advances in Image and Video Technology*, pages 100–113. Springer, 2007.

[9] Yefeng Zheng and David Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):643–649, 2006.

[10] David Kortenkamp and Terry Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *AAAI*, volume 94, pages 979–984, 1994.

[11] Alexander JB Trevor, John G Rogers, Carlos Nieto-Granda, and Henrik I Christensen. Feature-based mapping with grounded landmark and place labels. In *RSS Workshop on Grounding Human-Robot Dialog for Spatial Tasks (Los Angeles, CA*, 2011.

[12] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

[13] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[14] Christopher Evans. Notes on the opensurf library. *University of Bristol, Tech. Rep. CSTR-09-001, January*, 2009.

[15] Andras Majdik, Mircea Popa, Levente Tamas, Istvan Szoke, and Gheorghe Lazea. New approach in solving the kidnapped robot problem. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–6. VDE, 2010.

[16] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2, 2009.

[17] Joao Machado Santos, David Portugal, and Rui P Rocha. An evaluation of 2d slam techniques available in robot operating system. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–6. IEEE, 2013.