
André Francisco da Conceição

# Interbot Mobile Robot: Navigation Modules

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Electrical and Computer Engineering

September, 2016

UNIVERSIDADE DE COIMBRA

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

# INTERBOT Mobile Robot: Navigation Modules

**André Francisco da Conceição**

Coimbra, September 2016


# INTERBOT Mobile Robot: Navigation Modules

**Supervisor:**
Professor Doutor Urbano José Carreira Nunes

**Co-Supervisor:**
Professora Doutora Ana Cristina Barata Pires Lopes

**Jury:**
Professor Doutor António Paulo Mendes Breda Dias Coimbra
Professor Doutor Rui Alexandre de Matos Araújo
Professor Doutor Urbano José Carreira Nunes

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, September 2016

# Agradecimentos

Quero começar por agradecer aos orientadores desta dissertação de mestrado, Professor Doutor Urbano Nunes e Professora Doutora Ana Lopes, por me oferecerem todo o material e meios necessários para o trabalho ter sido possível, bem como a sua orientação em todos os momentos da elaboração do trabalho.

Aos meus colegas de laboratório, pela sua ajuda e por proporcionarem bons momentos de descontracção. Um especial agradecimento ao Jorge Perdigão, por me ter dado o impulso necessário para o começo do trabalho e por se mostrar disponível para me assistir, sempre que necessário.

Ao André Lopes, Daniel Almeida e Tiago Barros, pela sua paciência e disponibilidade para a ajudar, na resolução de alguns dos problemas encontrados ao longo desta etapa.

Ao ISR, pelas condições e recursos proporcionados, que permitiram que eu concluísse todos os objectivos a que me comprometi. Este trabalho foi suportado pela Fundação para a Ciência e Tecnologia (FCT), programas COMPETE e QREN, ao abrigo do projecto "AMS-HMI12 - *Assisted Mobility Supported by shared control and advanced Human Machine Interfaces*", com a referência RECI/EEI-AUT/0181/2012.

Aos meus colegas e amigos, pelo o vosso auxílio e pelos bons momentos passados ao longo desta jornada. Por esta vida académica que agora termina, mas que me ficou marcada e que irei levar comigo para o futuro.

Ao João, à Joana e à Mariana, pela força, carinho e amizade que foram bastante precisos ao longo deste projecto. Agradeço a vossa preciosa ajuda, os vossos conselhos, e todos os momentos inesquecíveis e experiências trocadas, que se tornaram bastante importantes, principalmente nos momentos mais difíceis.

Por fim, um especial agradecimento aos meus pais e ao Nuno. Obrigado pelo vosso esforço, protecção, suporte emocional e por estarem lá sempre que necessário. Fizeram de mim tudo o que eu sou até hoje, sem vocês isto tudo não seria possível... Um grande Obrigado!

# Resumo

Existe atualmente uma necessidade acrescida de recorrer a robôs autónomos para desempenhar algumas tarefas indispensáveis e que possam ser facilmente substituídas, sem causar nenhum transtorno à sociedade. Para isso, surgem os robôs de navegação centrados no humano, que são desenhados com o intuito de ajudar as pessoas a ultrapassarem algumas das suas dificuldades. Desde robôs guias, a robôs de interacção e de serviço, são muitas as possibilidades que existem no mundo da robótica, capazes de desempenhar este tipo de tarefas. Os robôs móveis que utilizam estratégias de navegação são, habitualmente, desenvolvidos para serem usados em instalações industriais, para o transporte de materiais ou outros componentes, em instalações hospitalares, para o acompanhamento e interacção com pacientes, em ambientes de escritório, onde podem desempenhar funções de auxílio a pessoas, aumentando a sua produtividade, ou mesmo em museus, servindo de guia para os visitantes. Para que estes robôs possam navegar, é necessário que se consigam localizar no ambiente em que operam, planear e seguir trajectórias, evitando qualquer tipo de obstáculos e adaptando-se às condições dinâmicas que possam surgir, tais como pessoas ou diversos obstáculos dinâmicos em circulação no ambiente, dos quais o robô não tem conhecimento *a priori*. Para isso, são necessários diversos módulos de *software* que integrem métodos de mapeamento, localização e planeamento de caminhos, para que, em conjunto, formem um sistema de navegação que seja eficiente e robusto.

Esta dissertação de mestrado teve como objectivo propor o desenvolvimento de um sistema integrado de navegação, aplicado num robô móvel desenvolvido no Instituto de Sistemas e Robótica (ISR), denominado "*Interbot - Social Robot*", usando dados sensoriais provenientes de um *laser range finder* (Hokuyo UTM-30lX). Foram desenvolvidas várias arquitecturas de *software* para navegação desta plataforma, a fim de se identificarem quais as vantagens e desvantagens dos métodos usados por cada arquitectura, após a realização de testes experimentais. Assim, pode-se apurar qual a melhor arquitectura para ser usada pelo *Interbot* para navegação em *long-term* dentro de ambientes de escritório, sem ter em conta, para já, pessoas e obstáculos dinâmicos. Como resultado final dos trabalhos realizados conducentes a esta dissertação, foi possível deixar o robô *Interbot* pronto a navegar no piso 0 do ISR, para que, no futuro, sejam realizadas melhorias e novas aplicações com o *Interbot*, designadamente em tarefas de interacção com seres humanos e tarefas envolvendo múltiplos robôs. Assim, como potenciais aplicações a curto prazo, estes robôs poderão ser usados pela comunidade de investigadores do ISR ou também como guias de visitantes do ISR ou do Departamento de Engenharia Eletrotécnica e de Computadores (DEEC).

**Palavras chave:** robôs autónomos, navegação, trajectórias, obstáculos, método de localização, método de mapeamento, método de planeamento de caminhos, *Interbot*

# Abstract

Nowadays, there is a growing need to use autonomous robots to perform some necessary tasks that can be easily replaced, without causing any disturb in the society. For that purpose, human-centered navigation robots were developed with the intent of helping people to overcome some of their difficulties. Guide robots, interaction and service robots, represent some of the existing possibilities in robotics capable of performing these kind of tasks. The mobile robots that use navigation strategies are usually developed with the purpose of being in industrial facilities, to carry resources or other components, in hospital facilities, for monitoring and helping patients, in office environments, where they can perform messenger tasks, to improve employee's productivity, or even at museums, to perform guided tours. In order to these robots being capable of navigating, it is necessary that they can localize themselves in the environment where they operate, plan and follow trajectories, avoiding any type of obstacles and adapting to the possible dynamic conditions, such as people or several dynamic obstacles in its surroundings, of which the robot has no *a priori* knowledge. To perform these tasks, several software modules that integrate mapping, localization and path planning methods are necessary, therefore creating an efficient and robust navigation system.

This master's dissertation had the aim of purposing the development of an integrated navigation system, applied in a mobile robot developed at Institute of Robotics and System (ISR), named "Interbot - Social Robot", using sensorial data from laser range finder (Hokuyo UTM-30lX). Several software architectures for the navigation of this platform were developed, in order to identify the advantages and disadvantages of the methods used by each architecture, after performing several experimental tests. Therefore, it can be identified which is the best architecture to be used by Interbot for a long-term navigation in indoor office environments, without taking into account, for now, people and dynamic obstacles. As final result of the works conducted by this dissertation, it will be possible to deploy the Interbot robot in the ISR ground floor, autonomously navigating, so that, in the future, some improvements can be applied and new applications using Interbot can be developed, such as interaction tasks with humans and tasks involving multiple robots. Thus, as potential short-term applications, these robots can be used by the ISR researchers community, or as guides for potential visitors of the ISR or Electrical and Computers Engineering Department (DEEC).

**Key words:** autonomous robots, navigation, trajectories, obstacles, localization method, mapping method, planning method, Interbot

*"Everything is theoretically impossible, until it is done."*

— Robert A. Heinlein

# Contents

# List of Acronyms

| | |
|---|---|
| **AMCL** | Augmented Monte Carlo Localization |
| **BMS** | Battery Management System |
| **CMU** | Carnegie Mellon University |
| **D-DWA** | Double-Dynamic Window Approach |
| **DEEC** | Departamento de Engenheria Electrotécnica e de Computadores |
| **DOF** | Degree Of Freedom |
| **DWA** | Dynamic Window Approach |
| **EKF** | Extended Kalman Filter |
| **FSPF** | Fast Sampling Plane Filtering |
| **HMP** | Hybrid Motion Planner |
| **IMU** | Inertial Measurement Unit |
| **ISR** | Institute of Robotic and Systems |
| **KIST** | Korea Institute of Science and Technology |
| **MCL** | Monte Carlo Localization |
| **RBPF** | Rao-Blackwellized Particle Filters |
| **RPM** | Revolutions Per Minute |
| **SLAM** | Simultaneous Localization and Mapping |
| **USB** | Universal Serial Bus |

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This chapter summarizes motivations that led to this dissertation, the main goals and the key contributions, as well as a resume of each chapter to contextualize the reader.

## 1.1 Motivation and Context

In the recent past, robotics had a great advance in terms of robotic navigation. Several methods and mobile robots were developed with a constant expansion and evolution. These mobile robots have to be capable of autonomously navigating in the environment were they are deployed and perform some activities and tasks which can be, at some level, helpful for humans. Taking this into account, these robots have to be human-centered platforms, so they have to know how to behave in environments where it can exist interaction with humans, in order to be socially accepted. Nowadays, this factor is still a problem that mobile robots developers have to face, because society is not used to this kind of technology, so there is the need to find strategies to make it more appealing. Mobile robots can be used in several environments, for instance, in industrial facilities, where they can carry resources and components, in medical institutions, for helping and interacting with patients, and in office environments or research institutions, to assist people and improve heir productivity. To this end, these robots have to own a navigation system with a software architecture composed by different modules. Each of these modules should be easily replaced once they integrate methods that are frequently updated. The most important modules and those that are worth to mention are the Perception module, where localization and mapping are part of, allowing the robot to know its surroundings and where it is at, and Path Planning module, which includes methods for computing trajectories for the robot to follow between several locations, always considering information obtained from the Perception module.

This dissertation work relies on applying a navigation system to a mobile robot named "Interbot-Social Robot". This robot was developed in ISR thanks to the work of several researchers, in particular, André Lopes, Daniel Almeida and Tiago Barros. The main challenge of this work was to develop a navigation system for the Interbot, based on laser scan readings, in order to enable a safe navigation, considering its surroundings and taking the shortest possible path, adapting to the several possible changes in the environment where it is inserted. Therefore, several software architectures using different methods for Perception and Path Planning were studied, implemented and tested using the Robot Operating System (ROS) framework, so that the advantages and limitations of each of these architectures were analyzed, for determining the methods to be integrated in Interbot, for indoor environment navigation purposes. In this first stage, dynamic environments were not taken into account. This research project can be the start of new and more ambitious ones

involving Interbot platforms. The Interbot can be replicated and used in ISR for other researching purposes, as well as it can be useful for performing simple tasks, like guide visitors, and complex ones, involving the cooperation of multiple robots.

## 1.2 Goals

The purpose of this work was to integrate several software architectures in a navigation system and determine which one need to be used in Interbot for a long-term navigation, considering the advantages and limitations of each method used by the Perception and Path Planning modules.

The main goals were:

1. Study different methods for mapping, localization and path planning;
2. Adapt the Hybrid Motion Planner (HMP) framework integrated in Collabnav node [1] into Interbot;
3. Integrate several architecture systems for navigation, taking into account the hardware available at Interbot platform;
4. Implementation of an integrated navigation architecture, in ROS environment, composed by the selected methods, with the purpose of achieving a long-term navigation.

## 1.3 Implementations and key contributions

The implementation of several software architectures for building a system on ROS framework, capable of performing a long-term navigation, was the major key contributions of this dissertation.

Figure 1.1 shows the main modules composing the navigation system. The chapters that compose this dissertation are presented and resumed below:

**Interbot Platform (Chapter 4)**

- A system overview about the Interbot low-level part containing a brief description about the mechanical and electronic components.
- Interbot's system architecture, describing all the components that are part of the hardware architecture, as well as a brief description about the main modules that compose the software architecture.

**Perception Module (Chapter 5)**

- Description of the methods that were used for localization (AMCL), mapping (Costmaps Building) and methods for Simultaneous Localization and Mapping (SLAM), for instance, Hector_SLAM and Gmapping.

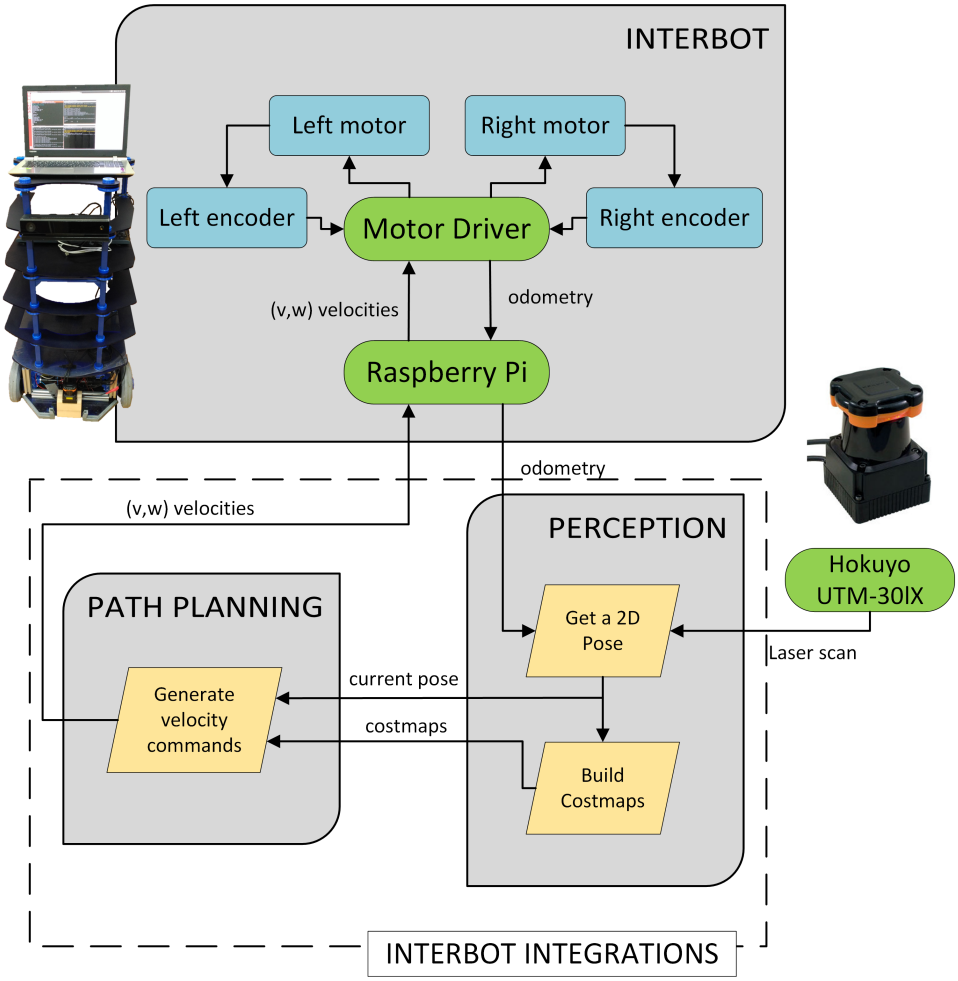



Figure 1.1: Navigation System main modules.

**Path Planner Module (Chapter 5)**

- Description of the methods that were used for path planning, in particular, move_base and HMP.
- Implemented global planning algorithm for being used along with HMP.

**Interbot Integrations (Chapter 5)**

- Five different software architecture, implemented in ROS, for the navigation system integrated in Interbot.

The Chapter 6 describes all the scenarios used to test the implemented architectures and their results are presented and discussed.

# Chapter 2

# State of the art

## 2.1 Indoor Mobile Robots

Robot navigation takes a fundamental role in mobile robotics. Research and development on this topic started many years ago and nowadays there are approaches and methods that are constantly evolving and are applied in many projects. Like a human being, for a safe and efficient navigation, a robot has to perceive its surroundings and decide a way to move from a point to another, avoiding obstacles and taking an appropriate path to a target location. For this simple task it is necessary modules for perception, including localization and mapping, and path planning. In other words, the robot uses sensor information so it can localize itself using a map of the environment and decide which direction and velocity are better to accomplish its target location, not forgetting its surroundings and possible new static and moving obstacles.

The following sections describe these modules and examples of algorithms that can be used in navigation systems. Figure 2.1 represents the navigation modules explored in this chapter.

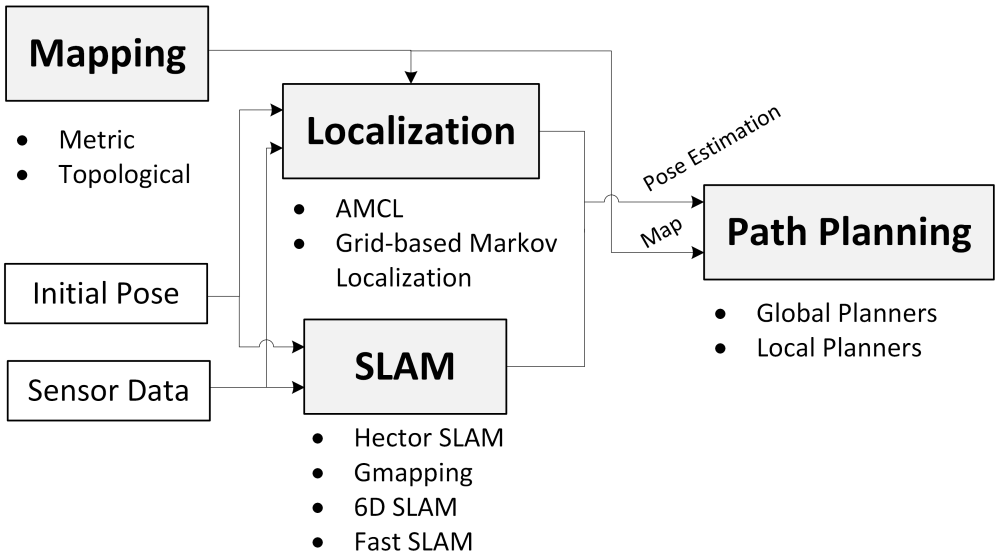



Figure 2.1: Typical modules for indoor mobile robots: mapping, localization, SLAM and Path Planning.

## 2.2 Mapping

Mapping corresponds to the ability for an autonomous robot to build a floor plan map that represents walls and other significant environment features using idiothetic (e.g.:encoders) and allothetic (e.g.:

lidar or sonar) sources. Therefore, these maps are often used by localization algorithms so that the robot can localize itself, and also by navigation algorithms in order to generate global and local paths from a point to another.

In [2], the two major map types for indoor environments are presented: metric and topological maps.

### 2.2.1 Metric Maps

Metric maps are also known as grid-based maps, and consist of a two-dimensional representation of the environment involving the robot by the mean of a discrete occupancy grid with a certain resolution. Examples of methods for obtaining these type of maps were proposed by Elfes, in [3], and Moravec, in [4].

Each grid cell has an associated occupation value which represents the probability of that cell being occupied by an obstacle. Basically, this cell occupation value increments in each iteration if the sensor beam indicates that cell as occupied and decrements it if the sensor beam indicates that cell as empty. These maps are easy to build and to maintain and their resolution is independent of the environment complexity because the resolution is determined by the cells size: a decrease on the cell size means an increase on the map resolution.

An example of a metric map is represented in Figure 2.2, where the black areas correspond occupied cells and the gray areas are the empty cells.

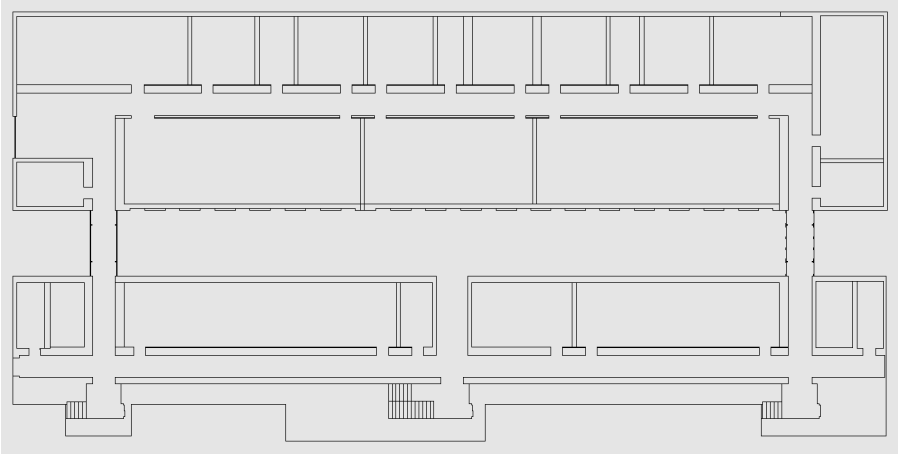


Figure 2.2: Metric map example.

### 2.2.2 Topological Maps

Topological maps map the environment using graphs with nodes and arcs, so it can be considered a high-level representation of the environment. Each node represents a specific place or landmark (eg. doorway) and they are connected using arcs, corresponding to paths between them.

The topological map resolution is determined by the environment complexity, so, generally, they are more compact than to the grid-based maps. This can become more intuitive for the user and, internally, the path planning is simplified and faster since the used computational resources are reduced.

This kind of map is build using a metric map. The free-space of the metric map is divided into several regions corresponding to rooms or corridors, separated by lines which represent narrow passages or doorways. For each region it is created a node and, as mentioned above, each node will be connected to another node using a arcs representing a path. Figure 2.3 presents an example of this type of map. As it can be seen, the nodes correspond to specific points of the metric map that can be critical for the navigation, such as door entrances or corridor corners, and the arcs are the paths between those nodes.

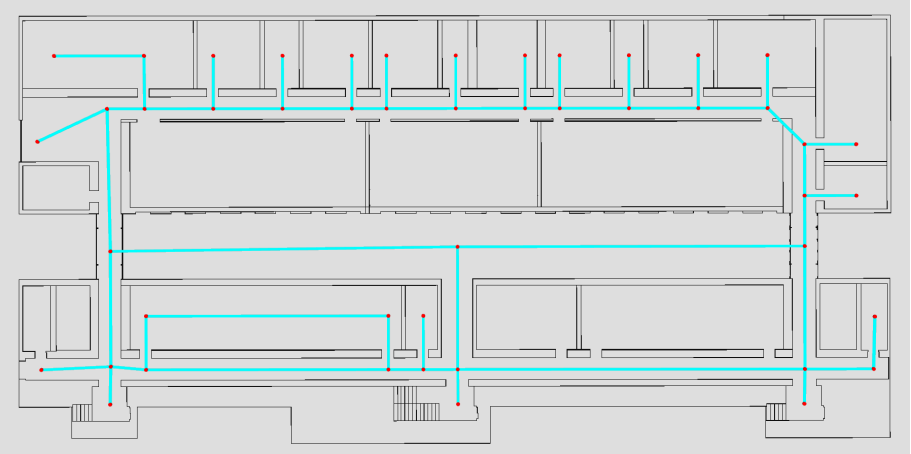


Figure 2.3: Topological map example. The map is represented by the light blue lines (paths) and the red spots (nodes).

Table 2.1 resumes the advantages and disadvantages of these types of maps described above. These types of maps are often combined to improve accuracy and efficiency for a reliable navigation.

Table 2.1: Comparison of metric and topological maps (adapted from [2])

| **Metric approaches** | **Topological approaches** |
|---|---|
| ⊕ easy to build, represent, and maintain | ⊕ allows global efficient planning, low space complexity (resolution depends on the complexity of the environment) |
| ⊕ recognition of places (based on geometry) is non-ambiguous | ⊕ does not require accurate determination of the robot's position |
| ⊕ facilitates computation of shortest paths | ⊖ difficult to construct and maintain in larger environments |
| ⊖ global planning inefficient, space consuming (resolution does not depend on the complexity of the environment) | ⊖ recognition of places (based on landmarks) often ambiguous |
| ⊖ requires accurate determination of the robot's position | ⊖ may yield suboptimal paths |

## 2.3 Localization

For an autonomous navigation, the robot has to keep track of its own pose, relatively to the environment where it is inserted. There are several localization techniques, the majority of them based on probabilistic localization.

According to [5] and [6], a belief is represented using a conditional probability distribution and corresponds to the robot's knowledge about the state of the environment involving it. This belief is the posterior probability distribution over a state variable, taking into account preceding measurements and it is given by

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}), \tag{2.1}$$

where $x_t$ is the state at time $t$, $z_{1:t}$ are the measurements obtained until $t$ and $u_{1:t}$ are the past commands.

At time $t$, the posterior continuous probability distribution can be computed by applying the Bayes Filter in a recursive function, using the belief at time $t-1$ and the most recent command and measurement data. The resulting algorithm can be divided into two steps: a motion update, where a belief $\bar{bel}(x_t)$ is calculated using the prior belief over state $x_{t-1}$, $bel(x_{t-1})$, and the command data $u_t$; a measurement update, where $\bar{bel}(x_t)$ is multiplied by $p(z_t|x_t)$, which is the sensor measurement model, and by $\eta$, which is a normalization factor. Methods that use continuous distributions in the belief equation 2.1 are the Kalman filter ([7]) and Multi-Hypothesis Tracking (MHT) ([8]), for instance.

Discrete distributions can also be used in Bayes Filter, although the posterior distribution $p(x_t|z_{1:t}, u_{1:t})$ has to be approximated by a finite number of values, each one corresponding to a region in space. For the purpose of this work, only methods that use this type of discrete distribution are important, so examples of this methods, for instance, grid-based Markov Localization and AMCL, are described below.

### 2.3.1 Grid-based Markov Localization

In accordance with [5], in Grid-based Markov Localization, the state space is partitioned in adjacent grid cells, each one of them with a corresponding belief value. this method is accurate and robust against sensor noise although it is dependent on the grid resolution in a way that a better resolution results in a higher accuracy but takes much computational effort.

Initially, the belief values are uniformly distributed over the pose state space. In each cycle iteration, the belief values increase in cells around the actual robot's pose and decrease as we move away from that location. This calculation is done recurring to the current scan and the latter state space. After some iterations, the probability mass is centered around the robot's actual pose and the robot is now localized.

This method can solve kidnapping problems after some iterations and does not need an initial pose if a global grid is used, although this compromises the computational complexity.

### 2.3.2 Augmented Monte Carlo Localization

According to [5], Monte Carlo Localization (MCL) is an algorithm applicable to both local and global localization problems and it represents the robot's belief about position by a set of $M$ particles with a certain importance weight.

The initial global uncertainty is obtained through a set of particles drawn random and uniformly over the pose space and as the robot senses something different over time, importance factors are

assigned to each particle. After re-sampling and incorporating the robot motion, the particle set has uniform importance weights and the most relevant places have a greater particle number associated. After several iterations, the location that has the most particles associated is the most likely location and correspond to the approximated robot's position.

MCL was not robust enough to recover from problems like global localization failures or kidnapping because, over time, the particles associated to locations other than the most likely poses gradually disappear and the algorithm is unable to recover in some cases. A second version of this algorithm named Augmented Monte Carlo Localization (AMCL) solves those possible problems by adding random particles based on some estimation of the location accuracy at each iteration. Short and long-term average of the measurement likelihood are tracked and if the short-term likelihood is better or equal to the long-term likelihood, none random particle is added. In the other hand, if the short-term likelihood is worse than the long-term likelihood, a number of random particles proportional to the quotient of these values is added. For instance, if the robot is kidnapped there is a sudden decay in measurement likelihood, so the number of random particles increases and after some iterations it is able to relocalize itself.

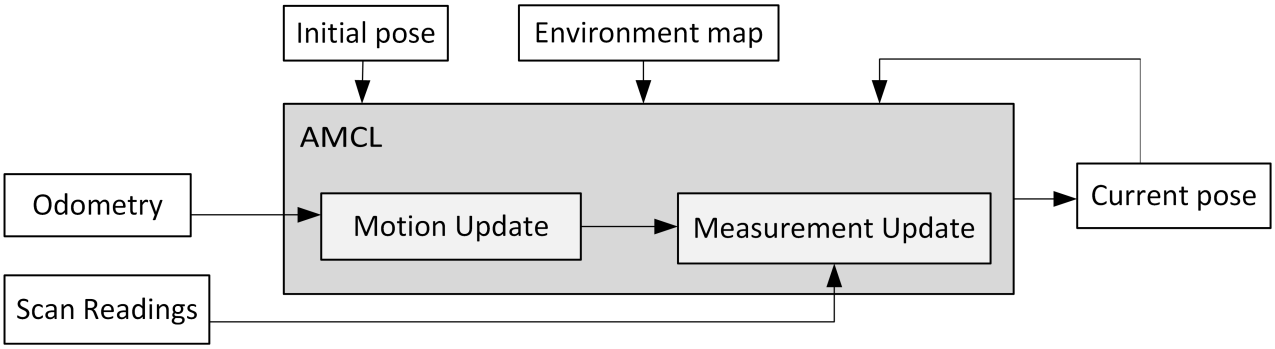



Figure 2.4: AMCL inputs and outputs.

## 2.4 Simultaneous Localization and Mapping

A robust navigation is only possible if an efficient localization and a map building algorithm is used which can reliably represent the robot's surrounding environment. There are algorithms that can perform both localization and mapping, known as SLAM algorithms .

Those algorithms are able to map the environment, localize the robot and they need to be capable of working in real-time. Among this algorithms, those which are worth to mention are, for instance, Hector SLAM [9], Gmapping [10],[11], 6D SLAM [12] or Fast SLAM [13].

For the propose of this work the last two algorithms were dismissed from the beginning because 6D SLAM needed the acquisition of 3D laser scans of the environment and Visual SLAM was not robust enough and caused high load on the CPU. So the algorithms that are worth mentioning due to the possible applications at the platform taking into account the available material are Hector SLAM and Gmapping which are mentioned below.

### 2.4.1 Hector SLAM

Using laserscan data, any navigation robot can use Hector SLAM ([9]) for grid map building and localization. In comparison with other grip-mapping techniques, hector slam is a good choice for systems that use high accuracy laser rangefinders but can't rely on wheel odometry and for systems that have low-power, low-weight and low-cost processors because it consumes low computational resources. This can be an advantage relatively to systems like Gmapping that uses accurate odometry and does not benefit so much from LIDAR systems.

This approach is based on the alignment optimization of the laser scan's endpoints with the grid map that is being published or has been obtained before (*a priori* map). There is no need for an association between the laser scans or a pose search which could overcharge the processor, because as those laser scans are aligned with the grid map, the matching process takes into account the preceding scans. The scan-matching is processed using a Gaussian-Newton equation, which finds the rigid transformation $\xi = (p_x, p_y, \psi)^T$ that fits the laser beams with the map.

If there are new static obstacles that do not exist in the map obtained so far, the occupancy grids are updated so that the robot can avoid them. It should be taken into consideration that further experiments with hector can confirm that this algorithm needs some iterations to make sure that a certain dynamic obstacle is not a static one. For instance, if a person is moving at the front of the laser, this person is not updated in the map.

One disadvantage of Hector SLAM, is that it is not a good option for large world scenarios because it does not provide loop closure, although results presented in the literature [9] show that this algorithm can accurately close the loops in many real world scenarios.


Figure 2.5: Hector SLAM inputs and outputs.

### 2.4.2 Gmapping

Gmapping is based on Rao-Blackwellized Particle Filters (RBPF) [11] and the main goal is to estimate the map and robot's localization using odometry and sensor's information. Basically, RBPF divides the process into two steps: first, using the odometry data and sensor's information, the robot's trajectory can be estimated; finally, given the trajectory estimated in the first step and using the sensor's observations, the map is computed.

Using a particle filter, RBPF estimates the possible posterior trajectory poses and since the map is dependent on the trajectory, a map is also computed for each trajectory. The chosen trajectory corresponds to the one that has the highest probability and the associated map is the output of the algorithm.

Gmapping improves the RBPF-SLAM performance by considering not only the robot's movement but also the most recent observation. With this improvement, the posterior poses take into

account previous odometry from the encoders and the most recent sensor's data information. Additionally, with this map and pose estimations, a scan-matcher is used to match up the observations with the current map, which gives the most likely robot's pose.

## 2.5 Path Planning

Path planning takes an important part in a robot's navigation system, once it is responsible for the decision of the best trajectory to follow considering the current velocity and collision avoidance, requiring sensor information, robust localization and costmaps to represent the environment being navigated by the robot.

Methods for path planning are divided into global planning, local planning and hybrid planning. Since the first two have advantages and disadvantages, hybrid planners were developed to combine the advantages of global deliberative planning and local reactive planning. These three methods are specified bellow.

### 2.5.1 Global Planners

This type of planning approaches have the advantage of offline computing a complete trajectory from the starting point to the end point, even if the end point is not reachable by the robot's sensors when the robot is at the start point. This can be a disadvantage as global planning do not consider unexpected changes in the environment involving the robot.

Another disadvantage is if the global world is inaccurate or if it is not available for some reason, for instance, in a populated indoor environment. This algorithms are relatively slow owning to the fact that this robot motion planning is really complex due to the repeated adjustments of the global planning.

These kind of approaches can also be divided into sampling-based and search-based algorithms. Sampling-based algorithms, such as probabilistic roadmaps [14], randomized path planners [15] and rapidly-exploring random trees [16], use a randomization of the configuration space. Search-based algorithms, for instance, A*[17] and D* [18], generate a graph representation of the planning problem.

### 2.5.2 Local Planners

Local path planning approaches cannot produce optimal solutions because they use small parts of the world model to generate robot control, and therefore rely on local environment information. For this reason, these algorithms require low computation, which represents an advantage for keeping up with the high rates of sensor information. They only consider a small subset of obstacles near the robot, allowing a fast re-adaptation to unforeseen obstacles and dynamic changes in the environment.

Examples of these approaches are: potential field methods [19], where obstacles impose negative forces on the robot and the goal location imposes a positive force; the vector field histogram [20], which uses an occupancy grid of the environment to construct an histogram representing the free space around the robot and it is used to compute velocities; Dynamic Window Approach (DWA), which is specified in section 3.1.

### 2.5.3 Hybrid Planners

As described in [1], hybrid planners gather the advantages of the global and local approaches and combine them into a single method. As result, the planned path will be the collision free shortest-distance path, taking into account a dynamic environment and providing a safe navigation. The majority of these algorithms uses the DWA's local path-planning approach.

Examples of these approaches are, for instance, D* combined with DWA [21], global DWA for holonomic robots [22] or for non-holonomic robots [23] and Hybrid Motion Planner (HMP), described in section 3.3.

## 2.6 Guide and interaction robots

Robots that provide tour guides and interact with people or perform some services like a night security or a messenger always have to put safety first and be interactive so that they can be trustful and somehow not intimidating. This required safety is associated with secure trajectory plans, safe obstacle avoidance and velocities that are suitable for navigation in populated indoor environments.

One of the main challenges presented over the years was to execute a safe and reliable navigation through crowds, which implies a navigation at approximate walking speed and at the same time not colliding with people and static or moving objects. Obstacles that are invisible to sensors (ex: glass cages) or the frequent modification of the environment are two of the most difficult problems to overcome. Along with this safe navigation, it is required an intuitive and appealing user interface in a way that these robots do not cause indifference or intimidate people. For this purpose, the design of these robots must be intuitive and user-friendly and must provide easy-to-use interfaces.

Several human-centered robots developed for guidance, interaction and service are presented in the next tables. Table 2.2 summarizes methods used for localization and mapping and Table 2.3 presents methods used for global and local planning. These tables can show how these methods have been evolving over the years, from the first autonomous robot deployed in a populated environment, to a robot that is able to navigate in an office environment 24 hours a day.

Table 2.2: Description, localization and mapping of interaction indoor mobile robots.

| Robot | Description/Sensors | Localization | Mapping |
|---|---|---|---|
| Rhino (1997) [24],[25] | Deployed during 6 days in "Deutsches Museum Bonn" , it was designed to guide people during museum tours. It could be controlled using a web interface. It was equipped with sonars proximity sensors, infrared sensors, laser range finders and a dual-color camera system mounted on a pan/tilt unit. | Markov algorithm was implemented along with an "entropy filter". This filter discards, for instance, sensor readings corresponding to people obstructing robot's sonars. | A metric map of the environment is provided. When the robot reaches a goal, it resets it's map to the initial one, ensuring that if a passage is once blocked it won't be avoided indefinitely. |

| Robot | Description/Sensors | Localization | Mapping |
|---|---|---|---|
| Minerva (1998) [26] | Is the second version of Rhino with new implementations that improved its performance. It was first deployed in the "Smithsonian's National Museum of American History" during two weeks of testing.<br>It had laser range finders, sonars, cameras, a pan/tilt unit and a touch-sensitive display. | It uses the same strategy than Rhino for separating corrupted sensor readings from authentic ones.<br>It is used a texture map of the ceiling for orientation as well, in order to solve the problem of high density of people surrounding the robot. | An occupancy map is obtained using laser range finders and sonars.<br>A texture map of the ceiling is learned using images from a camera pointed to the ceiling. |
| Robox (2001) [27] | Designed to guide people at the Swiss National Exhibition expo.02. A group of ten of these robots guided nearly a million people during 5 months, seven days a week, eleven hours a day.<br>Its hardware included laser scanners, a camera looking at the ceiling and tactile sensors. | An adaptation of the localization is applied [28]. | It is used a topological map.<br>It contains *ghost points* which act like invisible barriers or virtual readings so that the robot cannot reach forbidden areas that are not detected by its sensors like doors or staircases, for instance. |
| Jinny (2004) [29],[30] | Designed at Korea Institute of Science and Technology (KIST), it has been tested in office buildings and exhibits of Hyundai heavy industries. It is designed to interact with people and to perform some tasks.<br>It has two laser range finders set up in front and rear sides, two infrared sensors at different heights and an optical fiber gyroscope for localization improvement. | Its localization method is based on MCL. | The robot uses 3 types of maps: a topological map (global path planning), grid maps (map-matching) and active maps (local path planning). |
| PR2 (2010) [31] | Designed by Willow Garage, this is an autonomous indoor navigation robot for real office environments. It is regularly left running unattended at the office, during the night .<br>It is equipped with an Hokuyo UTM-30lx laser scanner at the base, another hokuyo on a tilting platform for 3D view, a camera mounted on a pan/tilt platform, two stereo camera pairs and an Inertial Measurement Unit (IMU). | If there is no *a priori* map, the robot's pose is estimated by integrating odometry merged with data received from an IMU.<br>If there is an *a priori* map the robot is localized using AMCL. | |
| Cobot (2011) [32] | Developed at Carnegie Mellon University (CMU), the Collaborative Robots perform autonomous tasks in populated office environments. From September 2011 to January 2013, a group of these robots have traveled 130 Km during 182 hours.<br>This robot has an Hokuyo URG-04lx short laser range finder at the drive base, a Microsoft Kinect depth camera sensor and a pan/tilt camera mounted on top. | Planar points and points observed by the laser range finder sensor are matched with expected features on a previously obtained map.<br>These planar points are obtained by using depth image points from a depth camera along with a filtering algorithm named Fast Sampling Plane Filtering (FSPF) described in [33]. | It is used a vector map, which represents the obstacles by a set of line segments.<br>The procedure for obtaining these maps is described in [32]. |

Table 2.3: Global and local path planning of interaction mobile robots.

| **Robot** | **Global Planning** | **Local Planning/Collision Avoidance** |
|---|---|---|
| Rhino (1997) [24] | Based on value iteration (described in the corresponding section of [24]). | $\mu$DWA algorithm [34] was implemented. It considers *hard constraints*, which ensure the robot's safety and express dynamic constraints, and *soft constraints*, which merge the robot's desire of moving to its goal and its desire to avoid obstacles that are in the way. |
| Minerva (1998) [26] | The path planning considers the path length and the information content (the amount of information expected to be observed at different environment's locations). | |
| Robox (2001) [27] | An optimum path is selected through several nodes of the topological map, depending on the current position and a target location. | Follows DWA's principle with some differences: the calculation of the trading off speed, heading and clearance, used for the objective function, is made in the actuator space ($v_r$,$v_l$); time to collision is used as a clearance measure instead of distance to collision; ghost points should be avoided. |
| Jinny (2004) [29],[30] | | First, the planner gets the robot position, the target location and the obstacle set computed using the active map.<br>Next, it checks for error states such as blocked paths or target goals occupied by obstacles.<br>Finally, the path is computed by moving in the direction of the navigation function gradient, suffering a smoothing process for performance improvement. |
| PR2 (2010) [31] | It uses an A* algorithm and plans directly in the configuration space. | It uses DWA's algorithm. |
| Cobot (2011) [32] | Using a topological map, the chosen path is the shortest one, taking into account the initial and final nodes along with distances between nodes that are part of that path. | Obstacles are avoided by computing the open path lengths which are available for the robot in different angular directions. |

# Chapter 3

# Background Material

## 3.1 Dynamic Window Approach

Synchrodrive mobile robots have some constraints related to their dynamics that can make a certain movement impossible to perform because they have limitations in velocities and accelerations given by the fact that there are velocities that can't be reached in short time. This limited accelerations imposed by the wheels's motors need to be considered by the robot so that the motion planning does not waste time in unnecessary calculations and improve computing time which is really important for a safe and efficient navigation. These dynamics cause constraints that can approximate the trajectory of synchrodrive robots by a sequence of circular arcs determined by the velocity vector $(v_i,\omega_i)$, which represent instant translational and rotational velocities.

According to [35], DWA is a local path planning algorithm that prevents platforms from colliding with obstacles, producing optimal local solutions, as it operates only in a small portion of the world model to generate robot control. This can be an advantage considering the high rates of the sensor information and the low computation complexity that is needed for this method. The algorithm consists in two different parts: (1) reducing the search space of the translational and rotational velocities for values that are possible to perform considering the dynamics of the robot and the actual velocity and (2) choose those the translational and rotational velocity that maximizes an objective function that is going to be specified later on.

### 3.1.1 Reducing the search space

The velocity search space needs to be reduced to those velocities which are possible to perform in short time due to dynamic constraints and to velocities that do not cause any robot's collision to an obstacle. This reduced part of the algorithm is done in three steps for discarding these velocities:

- Circular trajectories $(V_s)$

  Velocities $(v_i,\omega_i)$ have to be determined for each $n$ time interval between $t_0$ and $t_n$, considering that the trajectory which is generated does not intersect any obstacle. For this approach, only the first time interval is considered, assuming that the remaining $n-1$ time intervals are constant because the search is made after each time interval and the velocities for those $n-1$ time intervals are constant if there is not any new command.

- Admissible Velocities $(V_a)$

  A velocity is considered admissible when the robot is able to stop before it hits the closest obstacle. Considering $\dot{v}_b$ and $\dot{\omega}_b$ the translational and rotational accelerations for breakage,

respectively, and $dist(v, \omega)$ the distance to the closest obstacle in the robot's trajectory, the set of admissible velocities $V_a$, which allow the robot to stop before it collides with an obstacle, is given by

$$V_a = \{(v, \omega) | v \leq \sqrt{2 \cdot dist(v, \omega) \cdot \dot{v_b}} \wedge \omega \leq \sqrt{2 \cdot dist(v, \omega) \cdot \dot{\omega_b}}\} \tag{3.1}$$

- Dynamic window ($V_d$)

  Considering the limited accelerations imposed by the wheels motors, the dynamic window corresponds to those velocities that can be reached within the next time interval. So, the velocities that are present in the dynamic window $V_d$ are defined by

$$V_d = \{(v, \omega) | v \epsilon [v_a - \dot{v} \cdot t, v_a + \dot{v} \cdot t] \wedge \omega \epsilon [\omega_a - \dot{\omega} \cdot t, \omega_a + \dot{\omega} \cdot t]\}, \tag{3.2}$$

  where $t$ is the time interval during which $\dot{v}$ and $\dot{w}$ will be applied and $(v_a, \omega_a)$ is the actual velocity.

  From (3.2), it is noticed that the dynamic window is centered around the actual velocity and its limits depend on the accelerations that the motors can execute in the next time interval. All the velocities that stay outside of these limits are not considered for the obstacle avoidance.

- Resulting search space ($V_r$)

  Combining all the previous restrictions, the following intersection represents the reduced two-dimensional velocity search space that is going to be used for computing the objective function:

$$V_r = V_s \cap V_a \cap V_d \tag{3.3}$$

### 3.1.2 Maximizing the objective function

After determining the search space for the velocities, $V_r$ corresponds to the maximum of the objective function, incorporating the criteria (a) target heading, (b) clearance and (c) velocity. So, this objective function is represented by

$$G(v, \omega) = \sigma(\alpha \cdot \text{heading}(v, \omega) + \beta \cdot \text{dist}(v, \omega)) + \gamma \cdot \text{velocity}(v, \omega)), \tag{3.4}$$

where $\sigma$ is a function responsible for smoothing, and $\alpha$, $\beta$ and $\gamma$ are weight parameters.

The following items describe each function that is used for the calculation of this objective function.

- Target heading

  The function heading$(v, \omega)$ represents the alignment of the robot relatively to the goal direction. It is simply given by heading$(v, \omega) = 180 - \theta$, where $\theta$ corresponds to the target point angle, relatively to the robot's heading direction.

- Clearance

  The function dist$(v, \omega)$ corresponds to the distance to the nearest obstacle intersecting with the curvature that results from the selected velocity and if there is no obstacle intersecting this curvature this function is represented by a high constant.

- Velocity

  The function velocity$(v, \omega)$ is just the projection of the current robot's translational velocity.

- Smoothing

  The function $\sigma$ is used for smoothing the sum of the previous three components.

At the end, the maximum of the objective function is then chosen and the resulting velocity is then applied to the robot, ensuring that the movement that will be executed corresponds to a safe trajectory towards the target point of the robot.

## 3.2 Move_base

Using the robot's current pose, odometry information, an environment grid map, sensor's observation data and a specific goal, this algorithm is able to output the required trajectory to move the robot from a point to another. Figure 3.1 represents a high-level view of move_base and how it interacts with its inputs with the purpose of producing those velocity commands.

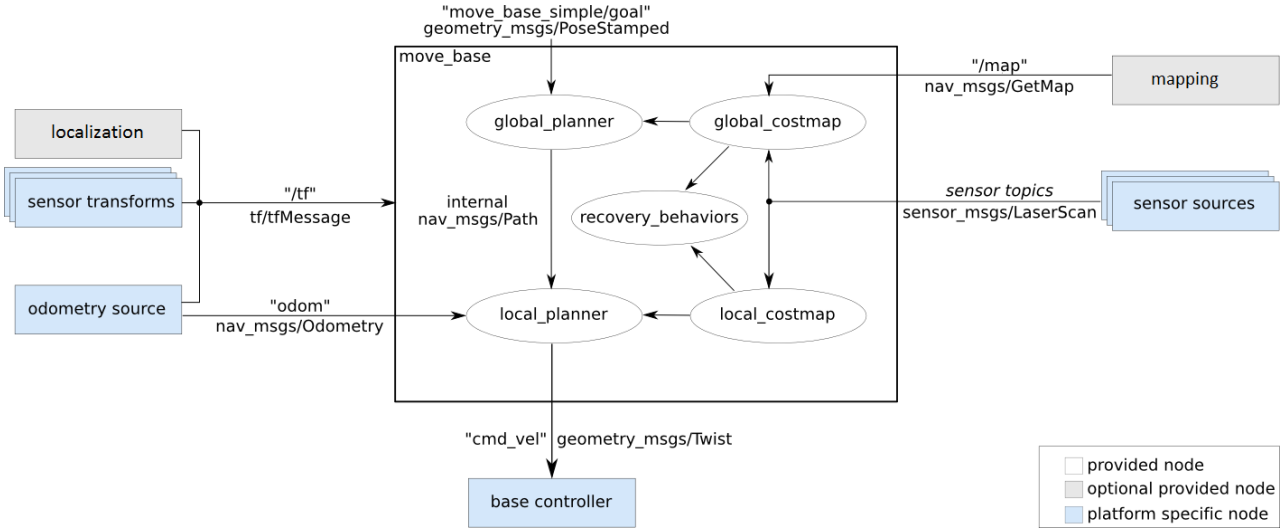



Figure 3.1: Move_base algorithm (adapted from `http://wiki.ros.org/move_base`).

### 3.2.1 Global Planner

Using a global costmap based on the occupancy grid map built dynamically as the map changes, along with the robot's current pose and a specific goal, the global planner creates a high-level plan for reaching that target location.

It assumes a circular robot and uses Dijkstra's algorithm, planning directly in the configuration space. In this case, dynamics and kinematics are not considered or, in other words, it just outputs two-dimensional points of the trajectory.

### 3.2.2 Local Planner

Taking into account the path created by the global planner, a local path is computed, which considers dynamics and kinematics of the robot.

Based on DWA's technique, the local planner uses a local costmap, built using the sensor's data, to produce the velocity commands necessary for a safe and robust navigation.

## 3.3 Hybrid Motion Planner

Developed at the ISR in University of Coimbra, HMP is a method integrated in a framework named CollabNav. This framework was primarily developed to be applied to a brain-actuated robotic wheelchair and it was designed to ensure a robust navigation in constrained environments, taking into account the needs of a single user. As this platform was designed to help severe motor-impaired users, this method needs to ensure safe navigation using smooth trajectories and avoiding collisions at all costs. The focus will be on the planner although there are other modules, in particular, perception, Human-Machine Interface and Collaborative Controller that together allow the system's navigation taking into account user's commands.

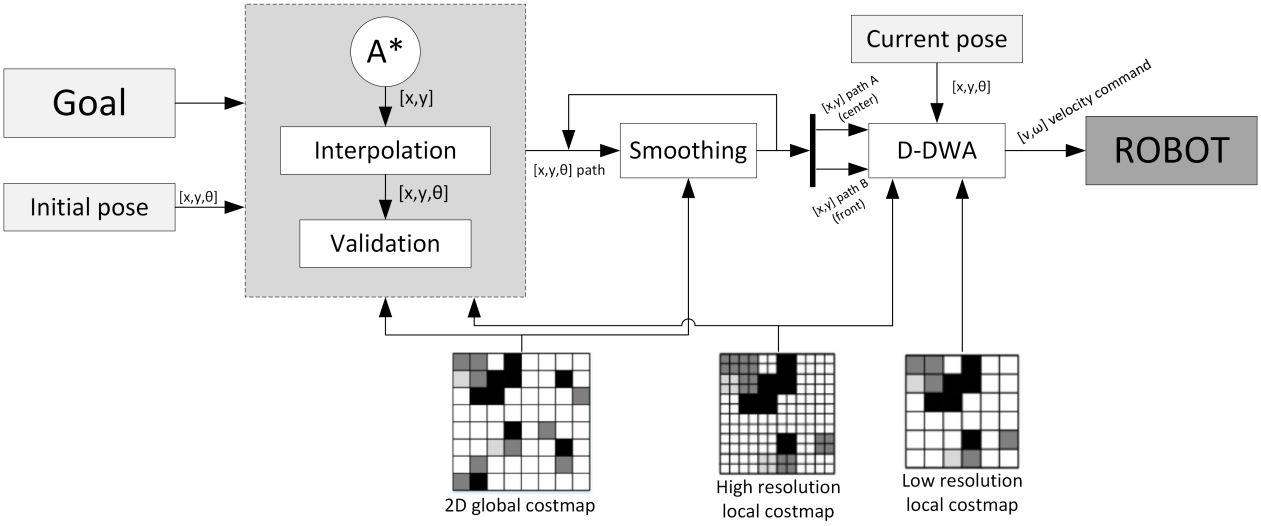



Figure 3.2: Hybrid Motion Planner (adapted from [1]).

Figure 3.2 represents the way all the modules that compose HMP are related. As the name indicates, HMP is a hybrid planner, so it combines the advantages of global approaches with local approaches and it can be divided in two separated planners: (a) a global planner which is a modification of the A* algorithm, expanded to a 3D-path planner for position $(x, y)$ and orientation $\theta$, using an interpolation module, along with a smoothing algorithm based on elastic bands [36]; (b) a local planner, to solve undetected situations that are not solved by the global planner, where two DWA are applied to different control points of the platform.

For this planning, three different types of costmaps are needed: (a) a 2D global costmap, used for keeping the robot away from obstacles, which is a low resolution map obtained from the *a priori* map, therefore including the static obstacles of the indoor environment, and it also includes the obstacles detected by the laserscanner in a specified radius; (b) low resolution costmap, which includes all the

obstacles detected by the laserscanner and it is used for most Double-Dynamic Window Approach (D-DWA) proceedings; (c) high resolution map, which is used for verifying collisions of the robot footprint in particular trajectories given by the D-DWA local plan and the 3D-global path planner. These costmaps are mentioned in the following sections that describe the main parts of the planner.

### 3.3.1 3D Global Path Planner

As this is the first part of HMP, the path go through a very fast superficial approach although the resulting path is unlikely to be performed, considering the geometry of most robots.

The result of A* applied to the inflated 2D global costmap is a geometric path composed by a sequence of points $s_j = [x_j, y_j]$, so it does not consider orientation which is required for an efficient navigation. Using an interpolation module it is possible to determine the orientation for each point in the resulting geometric path by computing $\theta_j = \arctan\frac{y_j - y_{j-1}}{x_j - x_{j-1}}$. After the interpolation, each point is represented by $s_j = [x_j, y_j, \theta_j]$, so the 2D global path is transformed to a 3D global path.

Additionally, invalid poses are corrected to valid ones in situations where the robot's footprint collides with obstacles present in the high resolution costmap or in locations like doorways or narrow passages.

In figure 3.3 is represented a global path computed for a specific trajectory where a robot needs to pass through a doorway and move to a goal situated in the middle of the corridor. The path corresponds to the black line and it is only computed at the first iteration, so if there are any obstacle not noticed by the robot, this path would be impossible to execute. This kind of situations are solved by the next parts of HMP.


Figure 3.3: Global path planner result visualized in Rviz.

### 3.3.2 Smoothing

The obtained global plan probably presents rough variations in the orientation after the validation process so it is applied an algorithm for smoothing based on elastic bands [36] to avoid these variations.

All the global plan points are considered as objects with mass $m$ and subsequent points, $j+1$ and

$j$, are connected by a 1-DOF (Degree of Freedom)spring with equilibrium length $d_j^0 = |s_{j+1} - s_j|^0$ and elastic constant $K_e$. With the exception of the initial and final plan points, each point will be subjected to a force $Fl_j$, such that:

$$Fl_j = K_e\Big( -\Big(d_{j-1} - d_{j-1}^0\Big)\frac{s_j - s_{j-1}}{d_{j-1}} + \Big(d_j - d_j^0\Big)\frac{s_{j+1} - s_j}{d_j}\Big), \tag{3.5}$$

where $d_j = |s_{j+1} - s_j|$.

For a smooth turning and to avoid sudden variations in orientation, it is considered a torque $\tau_j$ applied at the spring's center, given by:

$$\tau_j = K_T(-(\theta_j - \theta_{j-1}) + (\theta_{j+1} - \theta_j)), \tag{3.6}$$

where $\theta_j$ is the orientation of the plan point $s_j$ and $K_T$ is the torque constant.

The effect of $\tau_j$ is applied on two points: $s_j$ and $s_{j+1}$, taking the form of two forces, perpendicular to the vector $s_{j+1} - s_j$, and given by:

$$Fb1_{j+1} = \frac{\tau_j}{\frac{1}{2}d_j} \;;\; Fb2_j = \frac{\tau_{j-1}}{\frac{1}{2}d_{j-1}}. \tag{3.7}$$

Figure 3.4 represents the applied torque and the effect of these forces at the platform.

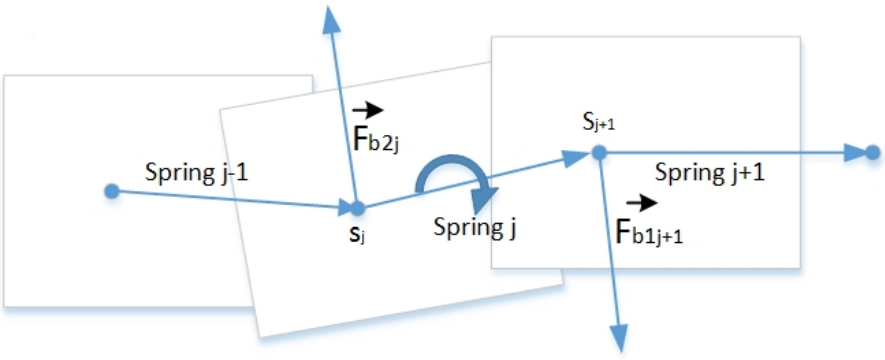



Figure 3.4: Representation of the applied torque $\tau_j$ and the resulting forces $Fb1_{j+1}$ and $Fb2_j$. [1]

Finally, each point $j$ is going to be subjected to a force $F_j$ given by:

$$F_j = Fl_j + (Fb2_j - Fb1_{j-1}) \cdot \hat{n}, \tag{3.8}$$

where $\hat{n}$ is the perpendicular versor of $s_{j+1} - s_j$.

As any force, $F_j$ generates an acceleration to the respective point of the path plan that is equal to $a_j = \frac{F_j}{m}$ and this acceleration causes a position's displacement of $j$, which is computed by the Leapfrog integration.

As in the 3D global path planner, at the end of the smoothing for each trajectory point, interpolation and validation modules are applied to guarantee that any point of the footprint does not collide with an obstacle in the high resolution costmap.

Figure 3.5 shows the smoothing process applied during the performance of the trajectory. The blue line corresponds the computed trajectory for the center of the robot which is translated to a trajectory for the front's center point of the platform, represented by the red line, and also a

trajectory for the back of the robot, represented by the black line. It can be noticed that the smoothing trajectory changes at almost every iteration, adapting to the robot's surrounding and getting smoother and smoother along the path.

### 3.3.3 Double Dynamic Window Approach

Based on the 3D plan originated previously, two 2D plans are determined for a point at the robot's center of mass and for a middle point located at the front of the robot. This approach guarantees the position and orientation needed for avoiding obstacles, considering that the robot does not have a circular geometry. Figure 3.6 shows how the algorithm is implemented.

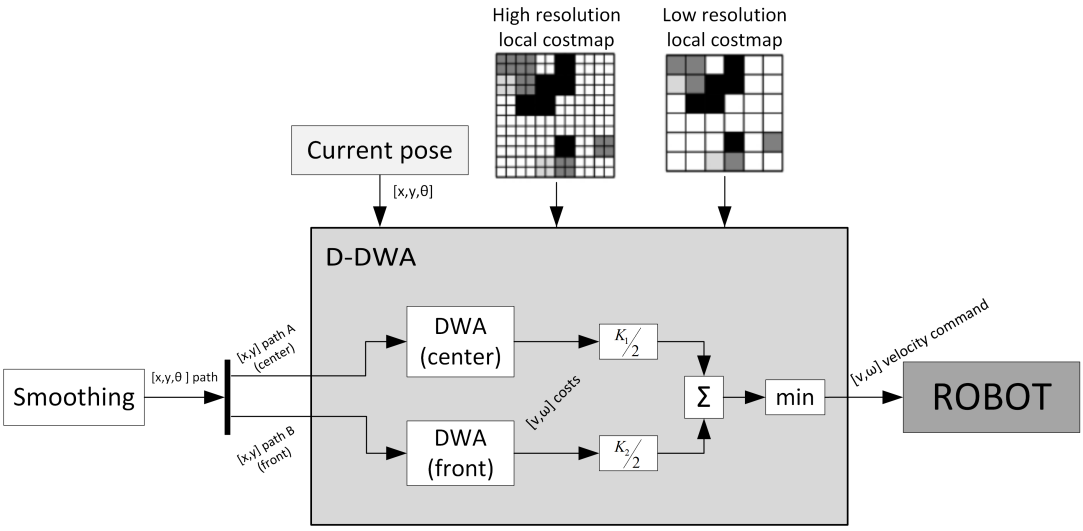
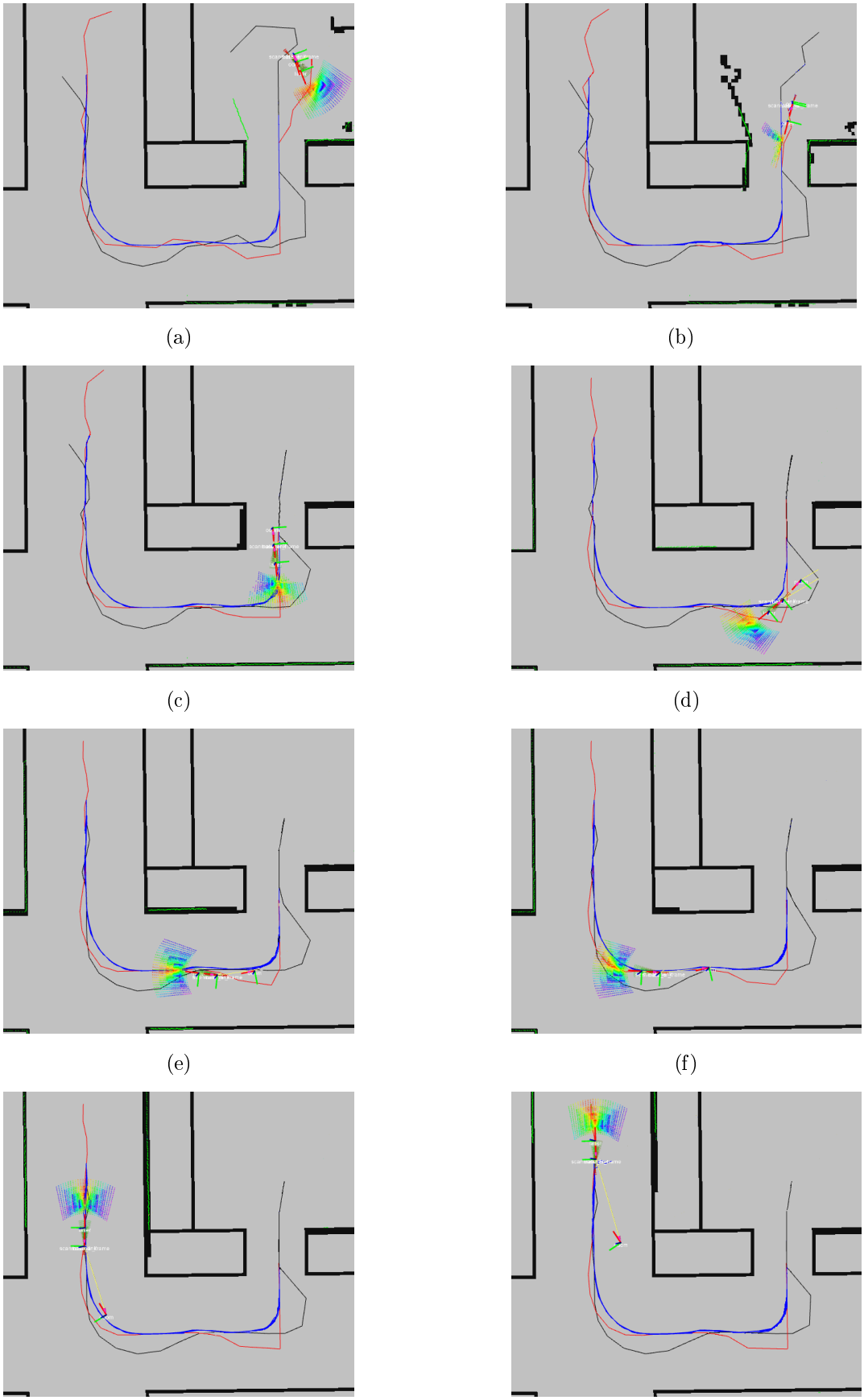



Figure 3.6: Double Dynamic Window Approach (adapted from [1]).

In each control cycle, the DWA module computes the velocity dynamic windows $[v, \omega]$ for the next center and front plan points, taking into account the current velocity and the imposed acceleration and velocity limits. Two trajectory sets are determined for the resulting velocity dynamic windows and it is checked if there is any point of the trajectory that collides with any obstacle present in the high resolution map, so that point can be discarded.

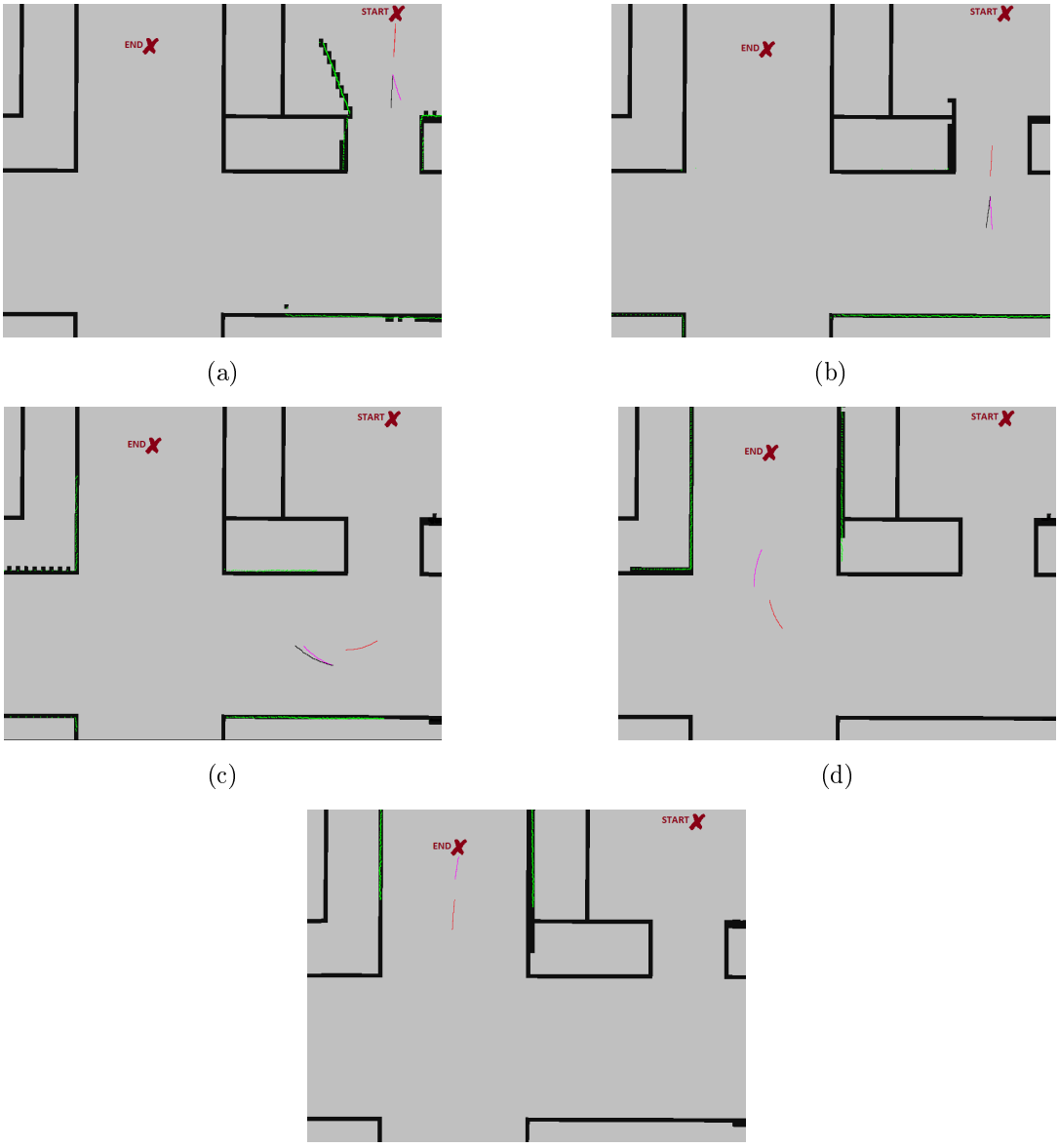
A cost is computed for each trajectory, adding the value of the low resolution local costmap for each point of the corresponding trajectory. The desired trajectory is the one that has the lower weighted cost or, in other words, the one that minimizes $\frac{K_1}{2}Cost_c + \frac{K_2}{2}Cost_f$, where $Cost_c$ and $Cost_f$ are the trajectory costs for the center and front points, respectively.

Figure 3.7 represents several plans generated by the D-DWA algorithm. The red line corresponds to the trajectory dictated by the DWA applied at the platform center and the black one to the trajectory given by the DWA applied at the front. The sum of those two trajectories multiplied by a constant results in the trajectory represented by the pink line. As expected, the plan changes during the performance of the trajectory as this is a local path planning algorithm, and it is just used for avoiding unforeseen obstacles and to adapt to unexpected situations.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3.5: Smoothing effect at the trajectory visualized in rviz.




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

Figure 3.7: D-DWA effect at the trajectory visualized in Rviz.

## 3.4 Floyd-Warshall Algorithm

Floyd-Warshall Algorithm (described in [37]) is a global planning approach used to compute the shortest path from a specific location to another. Having a graph, for instance, from a topological map, composed by nodes numbered from 1 to $n$ and arcs with a given distance, it is possible to use this algorithm for computing the shortest path from a node $i$ to the node $j$.

Considering $d_{ijk}(i, j, k$ from 1 to $n)$ as the distance of the shortest path from node $i$ to $j$, passing through the node $k$, and $d_{ij0}$ as the distance between the nodes which are directly connected, a matrix with dimension $n * n$ is initialized containing all the elements assigned to infinity, except for those elements corresponding to nodes which are directly connected, that are equal to $d_{ij0}$ and the matrix diagonal elements, that are equal to 0. For other values of $d_{ijk}$, there are two different situations: (a) if the shortest path from $i$ to $j$, does not pass through $k$, $d_{ijk} = d_{ijk-1}$; (b) if the

shortest path from $i$ to $j$, passes through $k$, $d_{ijk} = d_{ikk-1} + d_{kjk-1}$, because this path is composed by two secondary paths, one that goes from $i$ to $k$ and another one that goes from $k$ to $j$. Finally, for determining the shortest path between two different nodes, it is necessary to find the minimum value between $d_{ij0}$ and $d_{ijk} = min(d_{ijk-1}, d_{ikk-1} + d_{kjk-1})$. The Algorithm **??** presents the Floyd-Warshall algorithm written in pseudo-code.

**Algorithm 1:** Floyd-Warshall Algorithm

```
1  for x ← 1 to n do
2      for y ← 1 to n do
3          dist(x, y) ← ∞
4  foreach node z do
5      dist(z, z) ← 0
6  foreach path (u, v) do
7      dist(u, v) ← w(u, v)        ▷ w(u, v) is the weight of the path (u, v) or, in other words, the
         distance between node u and v
8  for k ← 1 to n do
9      for i ← 1 to n do
10         for j ← 1 to n do
11             if dist(i, j) > dist(i, k) + dist(k, j) then
12                 dist(i, j) ← dist(i, k) + dist(k, j)
13 return dist
```

After computing the algorithm presented above, each element of the matrix $dist(i, j)$ contains the distance between the node $i$ and $j$.

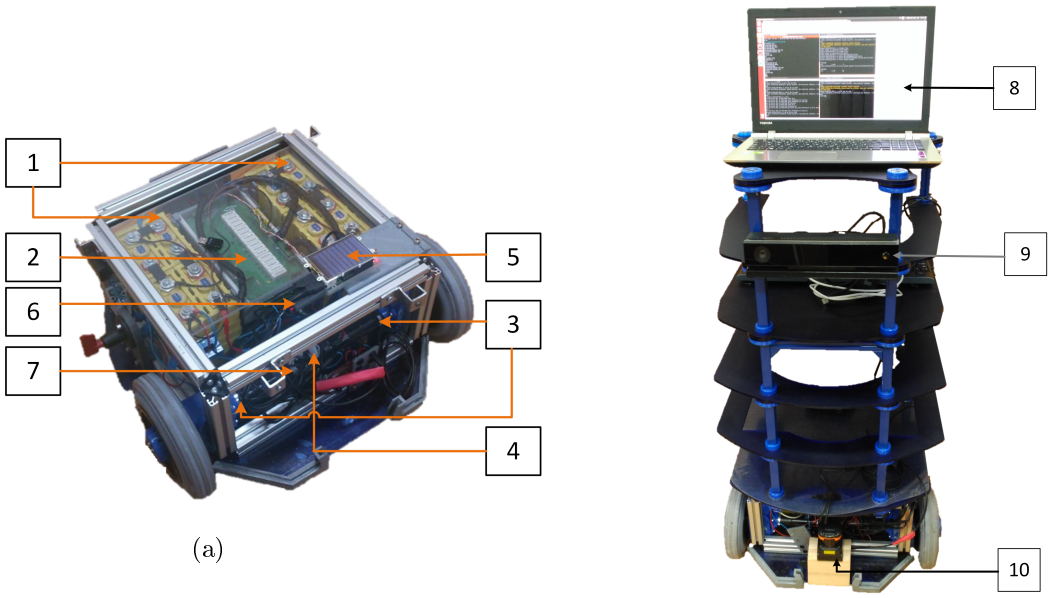
# Chapter 4

# Interbot Platform

Described in [38], Interbot is a prototype platform which has been developed in ISR for the project "AMS-HMI12 - Assisted Mobility Supported by Shared-Control and Advanced Human-Machine Interfaces" [1]. Its main purpose is to guide, interact with people and perform some tasks while navigating autonomously in populated office environments. This chapter presents the setup where all the work was implemented as well as the hardware and software that compose the system.

## 4.1 System Overview

Figure 4.1 represents the system's components. It is composed by a base (Figure 4.1a) containing all the low-level components, including the wheels, the motor's driver and the battery, for instance, and by a attached structure (Figure 4.1b) composed by metal bars and acrylic boards to sustain all the high-level components.




(a) (b)

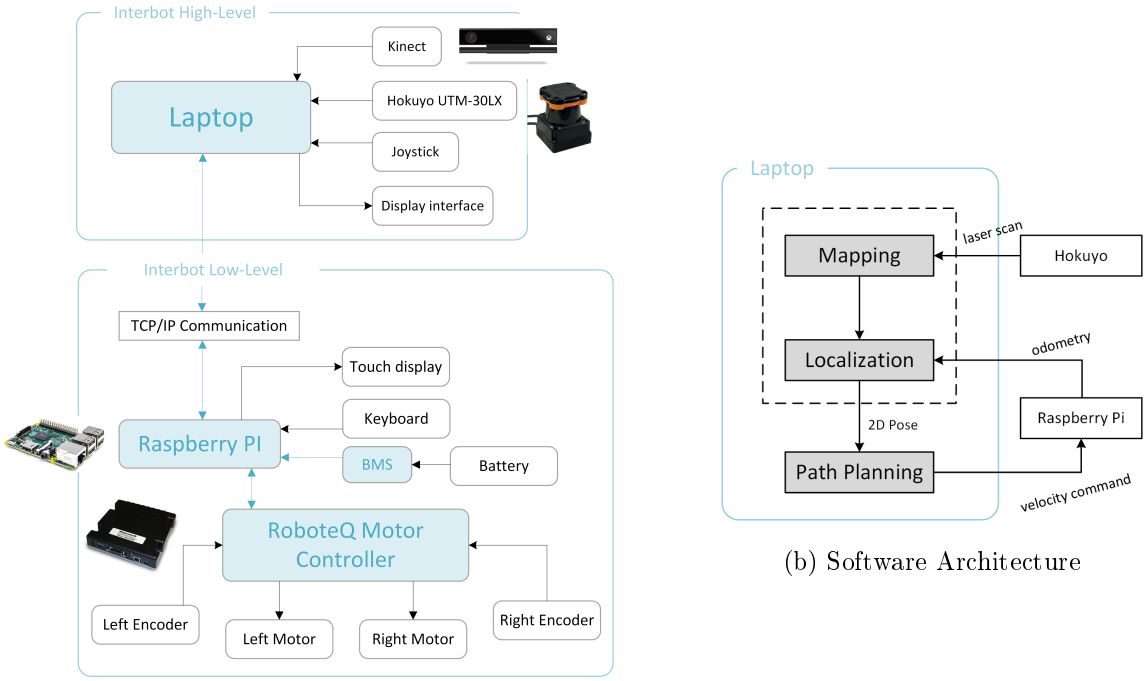
Figure 4.1: Interbot platform: (1) a 8 cells lithium battery; (2) a Battery Management System (BMS); (3) Two 24V DC permanent magnet motors, coupled to gearboxes with 29:1 and encoders; (4) Roboteq MDC2230 Dual channel motor controller; (5) Raspberry-Pi 2 with a touch display; (6) 24V DC to 12V DC converter; (7) 12V DC to 5V DC converter; (8) Laptop; (9) Microsoft Kinect for Xbox One; (10)Hokuyo UTM-30lX.

[1] `https://sites.google.com/site/amshmi12/`, at 5 September, 2016

Interbot is a differential robot with two motorized wheels connected to encoders used for odometry purposes, and a caster wheel to ensure the platform stability. A Roboteq motor driver is used to convert the speed commands from a Raspberry Pi into high voltage and high current output for driving two 24V DC permanent magnet wheel motors. It also converts pulses from the encoder to readable messages for the raspberry so they can be transmitted to the system's high-level part, the laptop, using a TCP/IP communication protocol. Each motor is connected to a gearbox with factor 29:1 (one complete wheel revolution corresponds to 29 complete motor revolutions). A BMS is used to manage the 8 cell lithium battery, responsible for powering the system, and to transmit messages to the Raspberry Pi, so that battery status information can be displayed to the user. Finally, the laptop is also connected to an Hokuyo UTM-30lX and a Microsoft Xbox Kinect and transforms the collected data into messages that can be used by ROS.

Due to the DC/DC converters, 24V, 12V and 5V are the available output voltages that can be used to power any device that can be connected to the platform. The 24V output is used to power the laptop and Roboteq. The 12V output powers the Microsoft Kinect and the Hokuyo laser range finder. Finally, Raspberry Pi is powered by the 5V output. Using other converters, it is possible to get any other output voltages and expand the outputs to use additional devices.

## 4.2 System Architecture




(a) Hardware Architecture

(b) Software Architecture

Figure 4.2: System Architecture.

As it can be seen in figure 4.2a, Interbot system is composed by a low-level part related to the power and mechanical components, which sends odometry information to the high-level part, that in charge of processing all the information and sending the speed commands to the low-level part. Raspberry Pi separates the low-level from the high-level architecture in a way that it receives pulse

messages from Roboteq, using a Universal Serial Bus (USB) connection, converts these messages to odometry information and then transmits it through a TCP/IP connection to the laptop. In the opposite direction, the laptop sends speed commands using the TCP/IP connection to the Raspberry Pi and then it transmits them to the motor's driver.

### 4.2.1 Hardware description

Each part of the hardware that composes Interbot is described below.

#### 4.2.1.1 BMS

The BMS controls the 8-cell lithium battery, protecting it from operating outside admissible values and providing its status.

#### 4.2.1.2 Roboteq's MDC2230 Motor Controller

It is a motor driver which has the purpose of converting speed command from a microcomputer into voltage and current outputs to separately control the right and left DC wheel motors. It also features a high-performance 32-bit microcomputer and quadrature encoder inputs to read pulses and transmit them to the Raspberry, using USB communication, so that they can be converted into odometry information.

#### 4.2.1.3 Raspberry Pi

Raspberry Pi is a microcomputer that runs Linux used in may applications. In this case, it is in charge of being the interpreter between the low and the high-level parts of Interbot. Its inputs are the Roboteq controller, the BMS and touch display plus a keyboard used for user's interface. As mentioned above, Raspberry receives pulses data from Roboteq, converts it into odometry data and sends it to the laptop, using a TCP/IP communication protocol. It also receives speed commands for each motor from the laptop through the same TCP/IP communication and sends them to Roboteq using a USB connection. Additionally it reads messages from the BMS so that information about the battery status can be displayed to the user.

#### 4.2.1.4 Hokuyo laser range finder

For the purpose of this work, the navigation rely on this Hokuyo UTM-30lX laser range finder which has a 30m and 270° scanning range. It generates scan messages of the involving environment and they are used as input data by the laptop software related to mapping and localization.

#### 4.2.1.5 Kinect

This is a 3D depth camera developed by Microsoft and Prime Sense in order to allow user to interact with videogames using body movements without requiring any physical control although it can also be used to robotics and computer vision purposes. One of its advantages is the price, which is significantly low, relatively to other 3D sensors and laser range finders. On the other hand, its range is not so large (0.6m to 5m) and it has a poor field of view, which can be dangerous if it is

desired a safe navigation. It has one RGB camera, one infrared camera and an infrared projector and using these 3 devices it is capable of generating depth images that can be used for 3D map building and navigation. This device is not used for the purpose of this work, although it can be used for future work.

#### 4.2.1.6 Laptop

It receives odometry information from Raspberry, joystick commands, laserscan data from Hokuyo laser range finder and image data from Kinect and processes this information to use it mainly for navigation. This is where all the software resides and its objective is to send velocity commands to Raspberry and to be an user's interface as well.

### 4.2.2 Software description

As it can be seen in figure 4.2b, the system's software relative to navigation which is present in the high-level architecture can be divided into two main modules. It was used open-source software for ROS, available in the ROS community along with software developed at ISR.

These next subsections specify the software that was used for this work.

#### 4.2.2.1 Mapping and Localization

Using odometry information and laser scan data from Raspberry and Hokuyo, respectively, it was possible for the robot to localize itself with an *a priori* map using the AMCL algorithm or to build a map and localize itself in real-time using the Hector SLAM or Gmapping algorithms. These methods are open-source, free to be used, and they are provided online.

#### 4.2.2.2 Path-Planning

Using the current pose, a map of the involving environment and laser scan information, path-planning algorithms output speed commands that are going to be provided to the Raspberry, so that they are transmitted to the motor's driver to move the robot. The used methods for path planning were move_base, which is an algorithm provided online and Hybrid Motion Planner, which was developed at ISR and it is not online available yet.

# Chapter 5

# Interbot: Software Modules

This chapter will resume the way the navigation system is integrated in Interbot. ROS was the robot middleware used for the majority of the software implementations, mainly because it supports C++ programming and there is an active online community which provides frequently updated open-source software for the most of robotic applications, supported by useful documentation. In [39], it is presented a ROS technical overview which defines important concepts, such as package, node, topic, message, subscriber and publisher. Basically, a package contains one or several nodes which can be defined as publishers, subscribers or both. A publisher node publishes topics containing messages of a particular type and a subscriber node subscribes topics from publishers, processing the obtained information.

Figure 5.1 shows a general overview of the implemented framework; modules will be specified in subsequent sections. In section 5.5 the different software architectures are summarized.

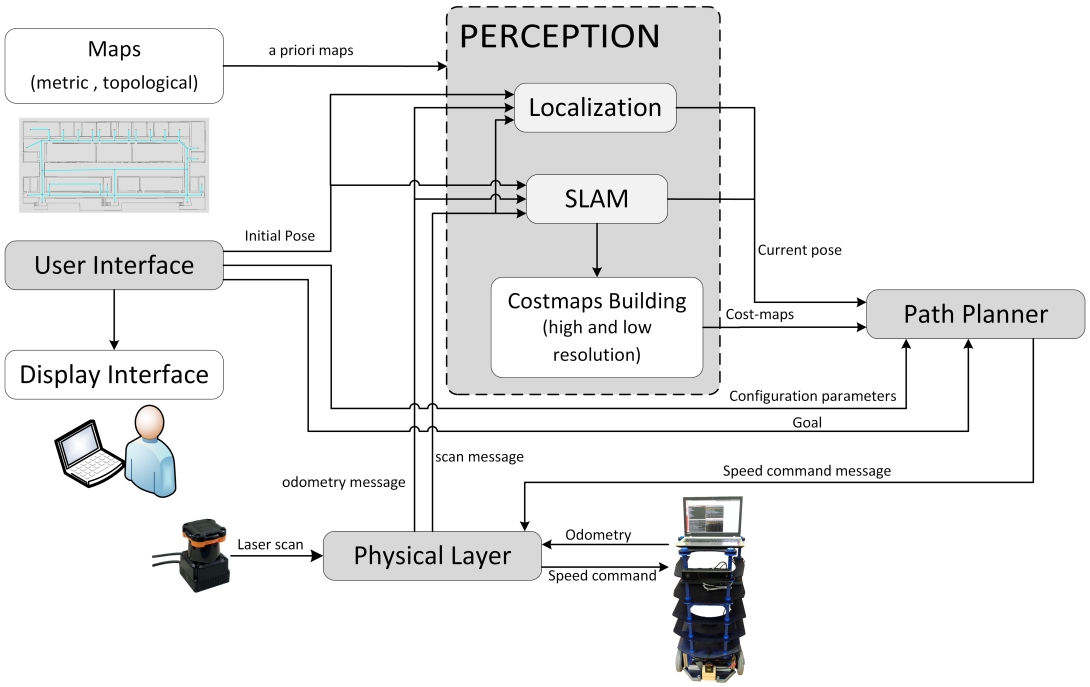



Figure 5.1: General overview of the implemented framework.

## 5.1 Physical Layer

Figure 5.2 represents the Physical Layer module. This module is the one responsible for linking the software and hardware, being in charge of receiving sensor data and delivering commands for Interbot actuation. This hardware-software link is guaranteed recurring to drivers which connect

the system's main hardware devices, e.g. Interbot micro-controllers and Hokuyo laser range finder, to the software modules that process the received information and generate velocity commands to move the robot. The `robchair_driver` was developed for the ISR intelligent wheelchair Robchair ([39],[1]) and, although it was made for a different platform, this driver was used here because the low-level system part is not much different. One of its main purposes is to receive odometry data from the Raspberry Pi and publish the transform of the base_link frame relatively to the odometry frame in order to define the relative position of the robot from a start point. It also receives the /cmd_vel topic containing the speed command message and sends it to the low-level part of Interbot. The `hokuyo_node` is a driver provided by the ROS community to be used exclusively along with an Hokuyo laser range finder and its purpose is to publish a /scan topic containing scan messages based on obtained laserscanner data.

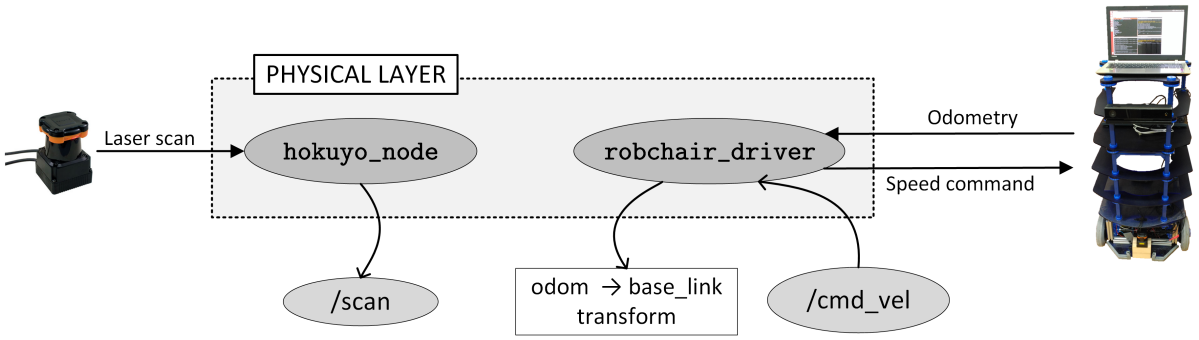



Figure 5.2: Physical layer drivers, including the subscribed and published topics.

## 5.2 User Interface

As the main purpose of Interbot is to interact with humans, it has to seem friendly and present to the user a simple and intuitive display interface. Using this interface, the user only has to inform the robot about its initial position and orientation (Initial Pose) using Rviz, a tool provided by ROS for visualizing the available topics, and a target point of the metric map where the robot is suppose to go (goal). Along with this information, the user is able to setup the path planner module (configuration parameters), configuring, for instance, the maximum and minimum linear speed, the goal tolerance and a weighting value related to the minimum distance among the robot and its surrounding environment.

## 5.3 Perception Module

To navigate from a point to another, the robot has to perceive its surroundings and a way to localize itself in the environment where it is inserted. For that, a perception module is needed in order to estimate some variables about the robot's base frame and this environment. Figure 5.3 presents a scheme of the perception module, including its input and output modules: the Physical layer, which provides odometry and laser scans; the User Interface, which gives the robot's initial pose; *a priori* maps of the environment, including metric and topological maps; the Path Planner, which needs costmaps and the robot's pose estimation.

Depending on the algorithms and sensors that are going to be used, the perception module can be divided into two operating modes: 1) if it is used a simple localization method, for instance, AMCL, the inputs are going to generate the robot's pose estimation; 2) if it is used a SLAM method, e.g. Hector SLAM or Gmapping, the inputs are going to generate the robot's pose estimation as well as an updated map of the environment, relatively to the initial *a priori* map. It is noted that costmaps are generated recurring to: the *a priori* maps if the localization method is used; the updated map if the SLAM method is used. These pose estimation and generated costmaps are the outputs of this perception module and are used by the Path Planner module. The following subsections describe these perception methods.

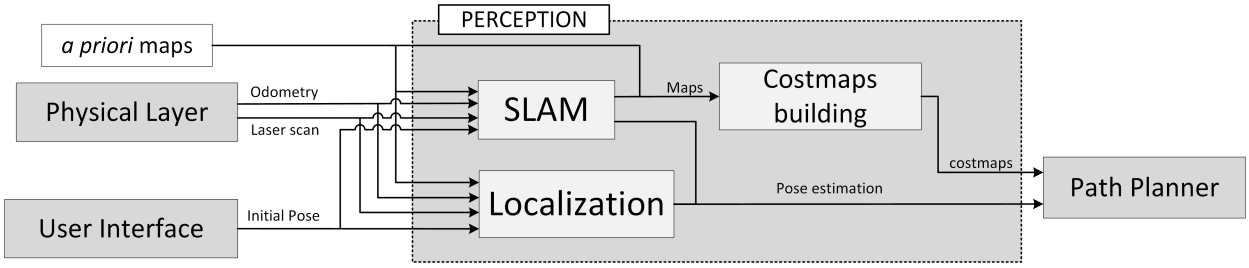



Figure 5.3: Perception module representation.

### 5.3.1 SLAM

Using sensor information, the SLAM module is capable of producing and updating a map and the robot's pose estimation relatively to an initial point. Depending on the used SLAM method, the needed information for generating the outputs is going to be different. Hector SLAM needs laser scans, an initial pose and an *a priori* map, while Gmapping inputs are the laser scans and odometry. Each one of these methods are provided by the ROS community and are specified below.

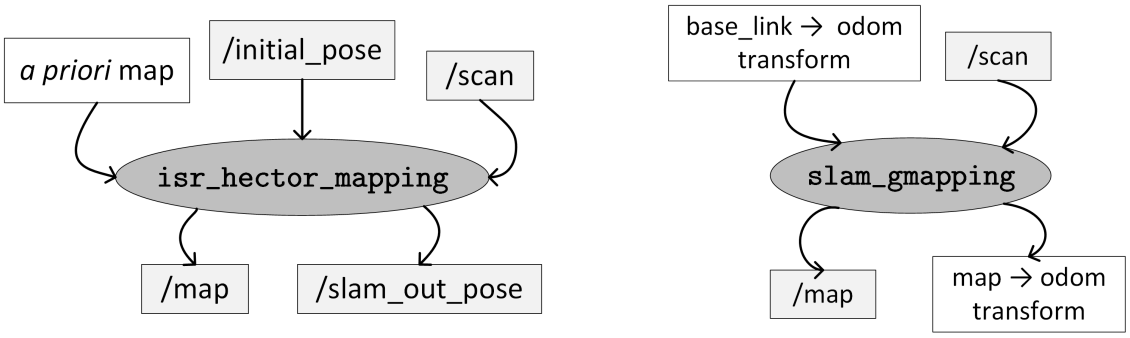
#### 5.3.1.1 Hector_SLAM node

In this work, it was used a modified version of the Hector SLAM package available online.

`isr_hector_mapping` is a node developed in [40] and it was altered for being able to receive an *a priori* map of the environment, in this particular case from the ground floor of ISR. Using this initial gridmap, an initial pose given by the user and laser scans from the topic /scan, it is generated an updated map of the environment (/map) as well as the robot's pose estimation (/slam_out_pose). Figure 5.4a represents the input and output topics of this node.

#### 5.3.1.2 Gmapping node

Figure 5.4b represents the Gmapping node. `slam_gmapping` belongs to the gmapping package provided by the ROS community and it is used by most of the mobile robotics applications due to its precise pose estimation. Using the the odometry, provided by the Interbot low-level part, translated into a transform of the base frame (base_link) relatively to the robot's initial point (given by the fixed odom frame), and the laser readings given by the /scan topic, `slam_gmapping` outputs the /map topic and the transform from the map to the odom frame.




(a) Hector mapping node.

(b) Gmapping node.

Figure 5.4: Published and subscribed topics of the SLAM nodes used in this work.

### 5.3.2 Localization

The Localization module is an alternative for the SLAM methods. The difference from those methods is that the provided *a priori* map is not updated and only the robot's pose estimation is generated. The used AMCL package was the one provided at the official ROS wiki page. As it was mentioned, using an *a priori* metric map of the environment, the topic containing the initial pose (/initial_pose), the laser scan provided by the /scan topic and the base_link transform about the robot's initial point (odom frame), the `amcl` node outputs the estimated robot's localization (/amcl_pose). The topics that are published and subscribed by the `amcl` node are represented in Figure 5.5.

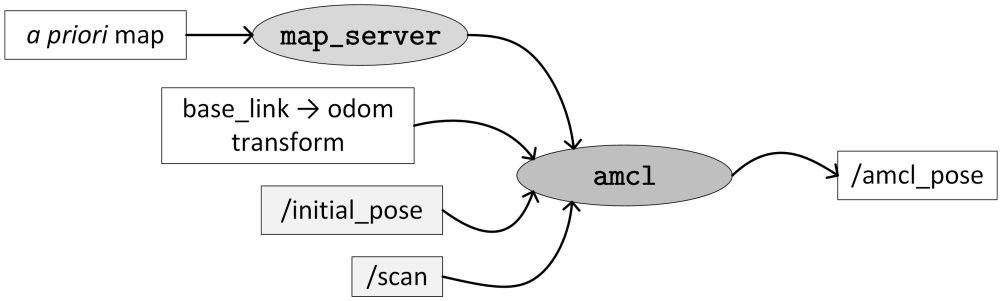



Figure 5.5: Published and subscribed topics of AMCL node.

### 5.3.3 Costmaps building

The last piece of the perception module corresponds to the costmaps building. Using laser scans and the updated gridmap from the SLAM module or the one based on the *a priori* map, costmaps of the environment are generated by inflating the occupied cells to the size of the robot, and are further used by the path planner module. There are built three types of costmaps: a 2D global costmap where each cell has a resolution of $5cm$ x $5cm$, including all the static obstacles from the global map and the obstacles detected by the laser scanner in a radius of 3 meters; two local costmaps with different cell resolutions ($3cm$ x $3cm$ and $1cm$ x $1cm$) and dimension $3m$ x $3m$, which include all the obstacles detected by the laser scanner. This module is exclusive for the path planner HMP, which is going to be specified below.

## 5.4 Path Planner module

This section is related to the Path Planner module represented in Figure 5.1. Here, are presented the path planner methods integrated in Interbot for navigation in the ROS point-of-view, which were already described in Chapter 3. Taking a higher view of this module, its inputs are the current pose, a variable representing the robot's surroundings (a map or a costmap), configuration parameters for the trajectory that is going to be generated and a target location representing the point where the robot is supposed to drive. Using these inputs, a speed command is generated and is transmitted to the physical layer by the mean of the topic /cmd_vel. In addiction to the path planner methods, move_base and HMP, which are specified below, it was developed a higher-level global planning algorithm to improve the performance of the final planned path.

### 5.4.1 Move_base node

Move_base is a ROS package available online and it is one of the most used path planners in ROS environments because it is simple to use and it has a good performance in many platforms. Figure 5.6 shows the topics that are subscribed and published by the `move_base` node.

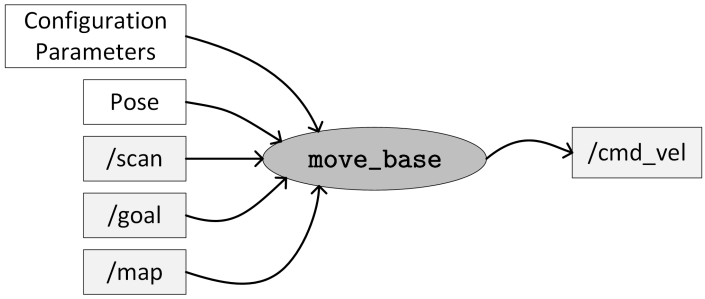



Figure 5.6: Published and subscribed topics of move_base node.

As it can be seen, in this case, the topic /map is subscribed, instead of the costmaps that are built in the perception module, because the global and local costmaps are built by the node itself, using this /map topic and laser scans from the /scan topic. Using the global costmap, the initial pose and a target location given by the /goal topic, a global path planner based in Dijskstra method computes the global optimal path. This global path is a reference to the local path planner, which using a local costmap that is built recurring to the /scan topic, generates a velocity command message that is going to be transmitted to the node `robchair_driver` by the mean of the topic /cmd_vel. Several configuration parameters can be defined for modifying the behavior of the local path planner and influence the decision of the generated trajectory. Those which have a greater effect on the trajectory must be the maximum and minimum value for the linear and rotational speeds (in meters and radians, respectively), the acceleration limit (in meters/sec$^2$), distance and rotational goal tolerances (in meters per second and radians per second, respectively), the step size to take between points of the trajectory (in meters) and weight values to influence the distance from the global path and from the obstacles.

### 5.4.2 Hybrid Motion Planner node

The `hybrid_motion_planner` node is based on the HMP approach, already described in section 3.3. Subscribing the 2D global costmap, the initial pose and the /goal topic, the A* approach is applied and it is obtained a 2D path with no orientation. An interpolation module determines the orientation for each 2D point of the 2D path and transforms it into a 3D point. A smoothing process is applied to avoid sharp variations of the orientation on each subsequent point of the trajectory. Then, two dynamic velocity windows are computed, one for a point at the center of the platform and another at the front. The velocities obtained inside those two dynamic windows generate trajectories for each one of the points, which are simulated to check if the robot's footprint collides with any obstacle present in the subscribed high resolution local costmap. If exists a collision at any point of the trajectory, it is automatically rejected. A cost is assigned to each trajectory, adding the value of the subscribed low resolution local costmap of each trajectory's point. The chosen trajectory is the one that have the lower cost and the corresponding velocity command is send to the physical layer using the /cmd_vel topic. Configuration parameters, for instance, the maximum and minimum linear and rotational speeds, and the distance and orientation tolerance for the target goal, are subscribed as well by the `hybrid_motion_planner` node. Figure 5.7 represents the inputs and outputs of this node.

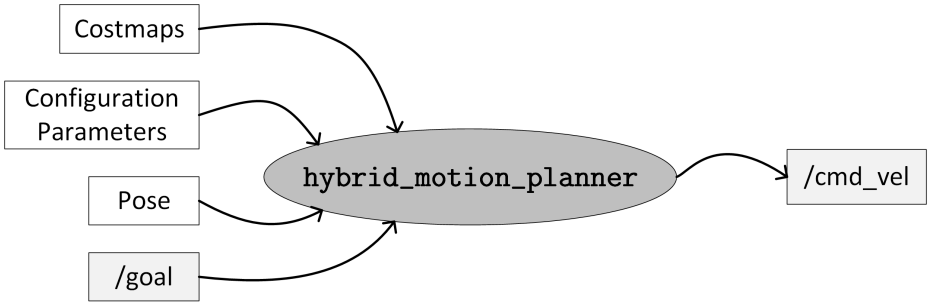



Figure 5.7: Published and subscribed topics of HMP node.

### 5.4.3 HMP_improvement Algorithm

Some of the HMP planned paths have problems if the platform needs to execute long distance routes, so one of the solutions was to publish intermediate goals along the path, depending on the start pose and the target location. Basically, each intermediate goal corresponds to positions where the robot has to perform 90° rotations without causing localization problems. For this, the robot needs to perform in-place rotations, instead of executing round turns. These specific situations are presented in Section XYZ. For this purpose, an algorithm based on Floyd-Warshall method was implemented, so that for each planned path it was computed a set of intermediate goals taking into account the initial pose, the target location and the shortest path between these two points. Each intermediate goal is represented by a node (corridor corners or doorways, for instance) and it is connected to one or more nodes using arcs with a defined distance.

Although there are other algorithms that can compute a shortest path, quicker, efficiently and using less computational resources, like Dijkstra's algorithm, for instance, this last one finds the

optimal route from one node to all the other nodes, while the Floyd-Warshall Algorithm finds the optimal route for all node pairings. This can be an advantage, because there is no need for computing another shortest path every time the user sets a new target location, so it is more efficient for a long-term usage. Algorithm 2 is an adaptation of Algorithm 1, where the output corresponds to a matrix $P$ with dimension $N$, where each element $P(i, j)$ contains the first intermediate node of the shortest path.

The matrix $P$ is the one which will be used to set the intermediate goals in Algorithm 3. The `HMP_improvement` node is based on Algorithm 3 and it asks the user for choosing the location where he wants for the robot to drive, and several intermediate goals are automatically set, depending on the chosen final goal. For this, the matrix $P$ indicates which nodes the robot needs to pass for reaching its main goal, and those nodes that correspond to corridors corners are marked as intermediate goals. After selecting the intermediate goals, the robot starts the navigation, tries to drive to the first intermediate goal, and after reaching it, the next intermediate goal is sent to the `hybrid_motion_planner` node by sending the topic /goal. After reaching its final goal, the algorithm is ready to ask the user for a new one. Figure 5.8 represents the node graph used for the purpose of this work and that is integrated in HMP_improvement algorithm.

**Algorithm 2:** floyd_algorithm(int P[N][N],D[N][N])

**Input:** A pointer to matrix P[N][N] and a matrix D[N][N] containing distances between all nodes

**Output:** Matrix P containing the first intermediate nodes of the shortest path

```
1  for i ← 1 to N do
2      for j ← 1 to N do
3          P[i − 1][j − 1] ← j
4  for j ← 1 to N do
5      for i ← 1 to N do
6          for k ← 1 to N do
7              if i ≠ j and k ≠ j then
8                  if D[i − 1][k − 1] > (D[i − 1][j − 1] + D[j − 1][k − 1]) then
9                      D[i − 1][k − 1] ← (D[i − 1][j − 1] + D[j − 1][k − 1])
10                     P[i − 1][k − 1] ← P[i − 1][j − 1]
11 return P
```

**Algorithm 3:** HMP_improvement(float *node*[2][$N$])

**Input:** a matrix *node*[2][$N$] containing the position $(x, y)$ of all the $N$ nodes of the topological map

```
1  D[i][j] ← matrix with distances between each node i and node j
2  P ← floyd_algorithm (D)
3  new_goal ← 1
4  while 1 do
5      pose_x=current_position_x
6      pose_y=current_position_y
7      if new_goal == 1 then
8          goal_index ← 1
9          node_f ←choose_destination()
10         node_i ← closest node to the initial position
11         SP[0] ← node_i
12         for i ← 1 to N do
13             if P[node_i − 1][node_f − 1] == node_f then
14                 break
15             node_i ← P[node_i − 1][node_f − 1]
16             SP[i] ← node_i
17         SP[i] ← node_f   ▷ SP contains all the intermediate goals from node_i to node_f
18         node_∗ ← poses (x,y,θ) of the important nodes (corridor corners and node_f)
            goal_number ← total number of goals
19     if aux ≠ goal_index then
20         aux ← goal_index
21         goal.pose.position.x ← node_ ∗ [0]
22         goal.pose.position.y ← node_ ∗ [1]
23         goal.pose.orientation ← node_ ∗ [2]
24         publish(goal)
25     dist ← distance between the current pose and the goal pose
26     if dist < xy_tolerance then
27         dif ← angular difference between the current pose and the goal pose
28         if dif < yaw_tolerance then
29             destination_check ← 1                ▷ The robot has reached a goal.
30     if destination_check == 1 and goal_index == (goal_number − 1) then
31                                  ▷ The robot has reached its final goal.
32         new_goal ← 1
33         destination_check ← 0
34     if destination_check == 1 then
35                             ▷ The robot has reached an intermediate goal.
36         destination_check ← 0
37         goal_index + +
```

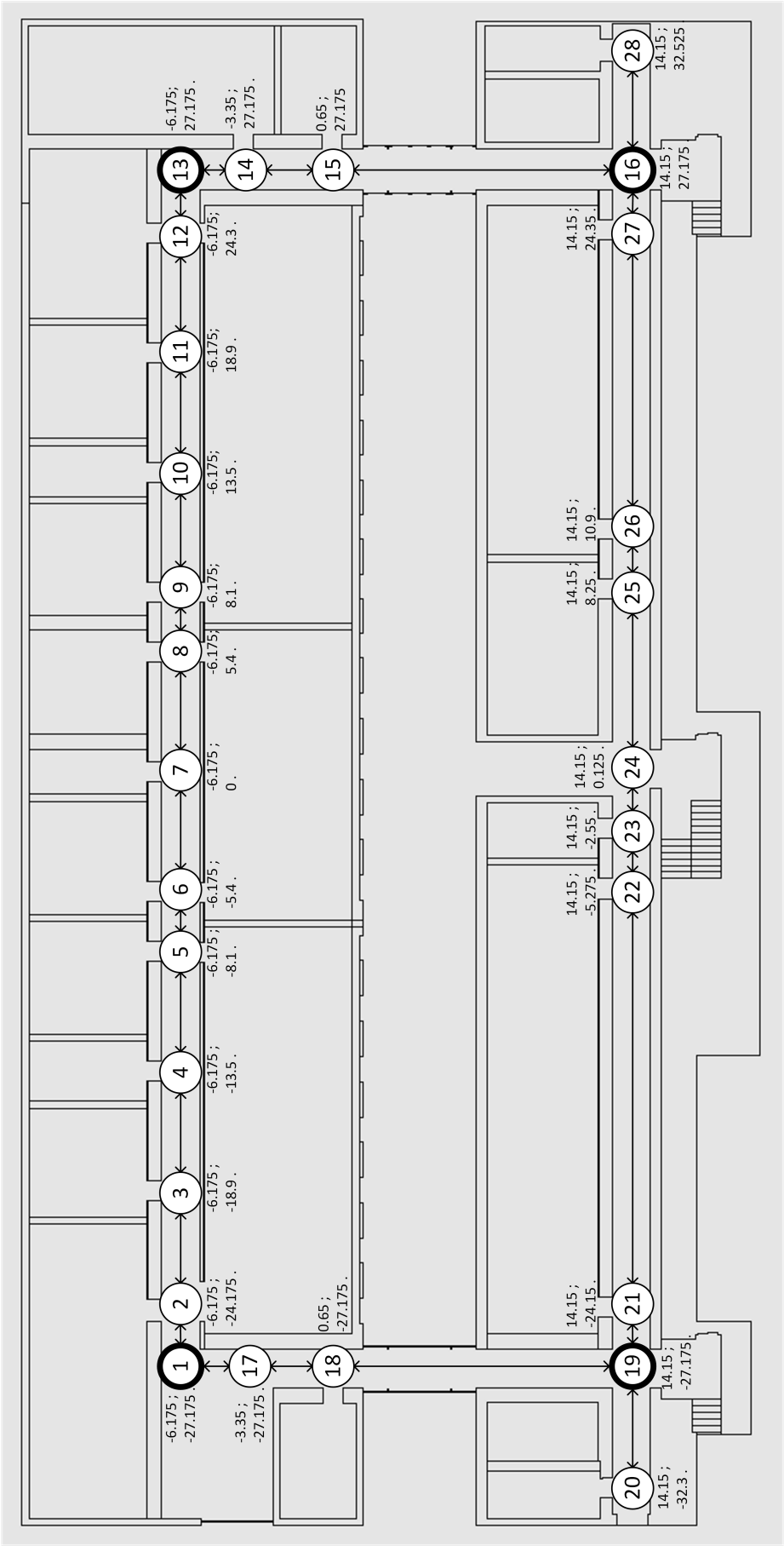



Figure 5.8: Node diagram for the nodes composing the graph used by the HMP_improvement algorithm, representing the position $(x, y)$ of each node and those nodes that can be used as intermediate goals.

## 5.5 Interbot Integrations

This last section explains the way the perception and path planner modules were integrated in Interbot. Different software architectures were tested for this integration, each one of them using one method of the perception module along with one method of the path planner module. The result is four architectures which combine a perception method, such as Hector SLAM or AMCL, with a path planner method, in particular move_base or HMP. Each one of these architectures was properly tested and all the experimental cases and test scenarios with different setups are going to be documented at Chapter 6. The next subsections describe these four architectures, indicating the used nodes and which topics they publish and subscribe.

### 5.5.1 Architecture 1

The first architecture uses `isr_hector_mapping` as the perception method and `move_base` as the path planner method. Figure 5.9 represents the used software architecture.

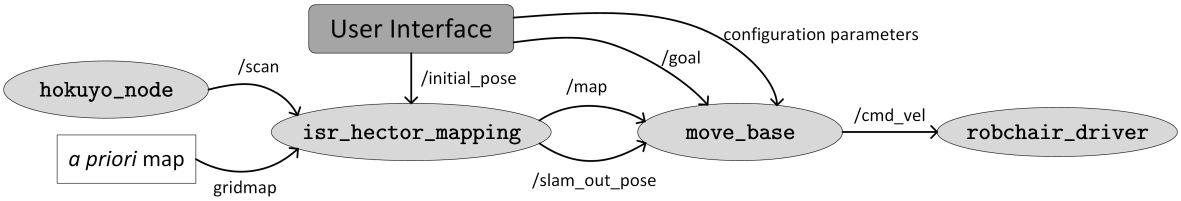



Figure 5.9: Architecture 1 : `isr_hector_mapping` node along with `move_base` node.

### 5.5.2 Architecture 2

Figure 5.10 represents the second software architecture, which uses the perception node `isr_hector_mapping` along with the path planner node `hybrid_motion_planner`.

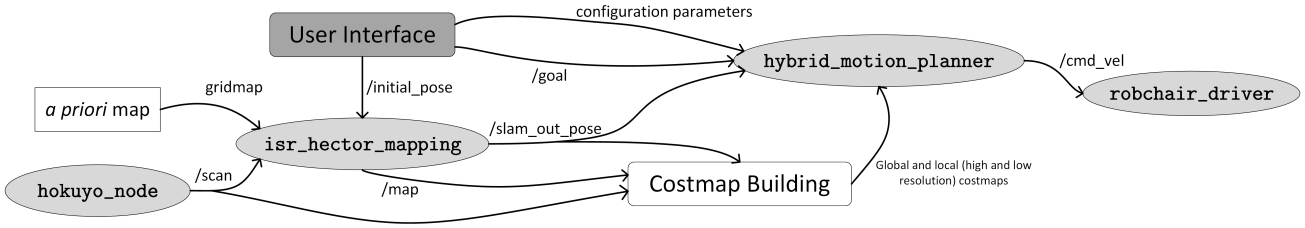



Figure 5.10: Architecture 2 : `isr_hector_mapping` node along with `hybrid_motion_planner` node.

### 5.5.3 Architecture 3

The third software architecture is represented in Figure 5.11 and it uses the `amcl` node and the `move_base` node.

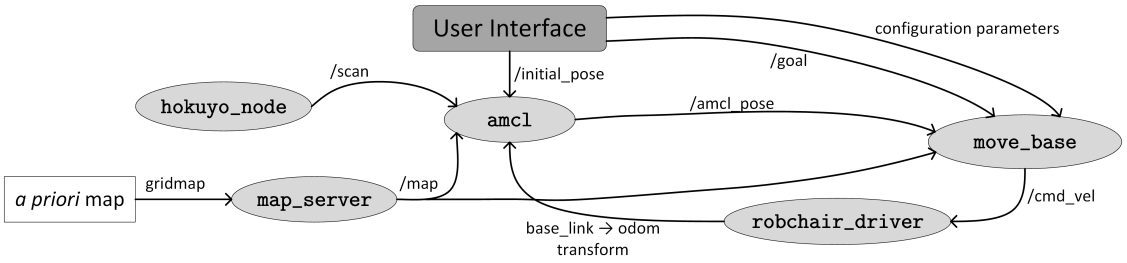



Figure 5.11: Architecture 3 : `amcl` node along with `move_base` node.

### 5.5.4 Architecture 4

The last architecture uses AMCL for the robot's localization and HMP for the path planning. It is represented in Figure 5.13

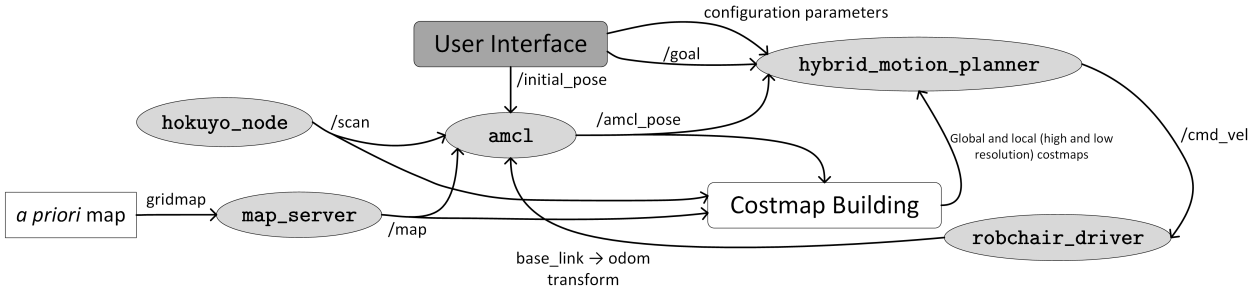



Figure 5.12: Architecture 4 : `amcl` node along with `hybrid_motion_planner` node.

### 5.5.5 Architecture 5

This architecture adds the `HMP_improvement` node to the path planning module of Architecture 2.

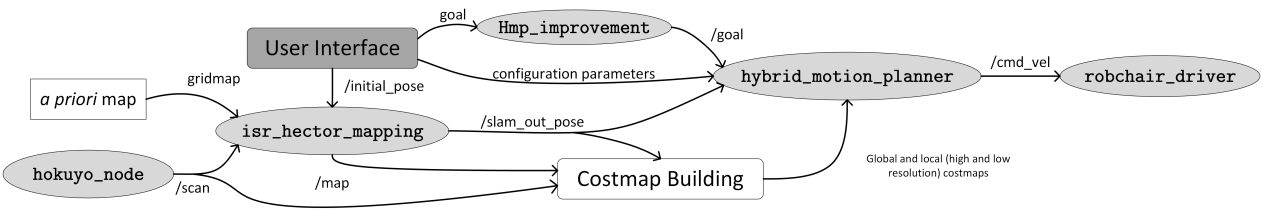



Figure 5.13: Architecture 5 : `isr_hector_mapping` along with `HMP_improvement` and `hybrid_motion_planner` nodes.

### 5.5.6 Discussion

Table 5.1 resumes the implemented architectures, presenting which nodes were used for each method. This can be useful to understand the terminology that is going to be used in the next chapter.

Table 5.1: ROS nodes used in each architecture.

| Arch. | Laser driver | Perception | Path Planner | Robot driver |
|---|---|---|---|---|
| **1** | `hokuyo_node` | `isr_hector_mapping` | `move_base` | `robchair_driver` |
| **2** | `hokuyo_node` | `isr_hector_mapping` + Costmaps building | `hybrid_motion_planner` | `robchair_driver` |
| **3** | `hokuyo_node` | `map_server` + `amcl` | `move_base` | `robchair_driver` |
| **4** | `hokuyo_node` | `map_server` + `amcl` + Costmaps building | `hybrid_motion_planner` | `robchair_driver` |
| **5** | `hokuyo_node` | `isr_hector_mapping` + Costmaps building | `HMP_improvement` + `hybrid_motion_planner` | `robchair_driver` |

Gmapping was a good method to be used in a new software architecture because it is a SLAM method used in many implementations. For the purpose of this work this could be a good choice for being part of the perception module because the estimated odometry could be used for complementing the scan-matching process, which can be a significant advantage comparatively to other SLAM methods, in particular Hector SLAM, which relay its SLAM method on the laser scans alone. Unfortunately, the ROS package which is available online does not provide a way to integrate an *a priori* map of the environment, as opposed to Hector SLAM. The `slam_gmapping` node uses a grid representation of the map and estimates probability distributions over maps and robot's trajectories so its internal state is difficult to save and to be used in the future. Of course that this is a disadvantage for this kind of systems because they need to use long-term navigation and, for each new usage, the map had to be built from the beginning. In conclusion, and after some thoughts into this matter, the `slam_gmapping` node was discarded, although there were developed some experiments with it to test its accuracy.

# Chapter 6

# Experimental Results

This chapter is dedicated to the documentation of the experimental results carried out by the Interbot platform. Using different architectures presented in Table 5.1, three test scenarios were executed in order to evaluate the navigation system performance, in particular the Path Planning module. For this, the robot was deployed on the ground floor of ISR in order to navigate through a set of goals in a office like environment. The tests were performed during nighttime in order to avoid populated corridors, which could compromise the navigation process, so people and other dynamic obstacles were not considered for the purpose of this work.

Table 6.1 contains the configuration parameters set for the architectures that used `move_base` as the path planning node. Some of these parameters are still the default values defined by move_base package creators, such as the rotational goal tolerance, the step size between trajectory points and the weight values for obstacle and global path distance. The other values were suited to the platform capacities and to the path type that was required. Table 6.2 presents the parameters for the `hybrid_motion_planner` node.

Table 6.1: Configuration Parameters used by the `move_base` node

| **Maximum Speed** | | **Minimum Speed** | | **Acceleration Limit** ($m/s^2$) | **Goal Tolerances** | | **Step size between trajectory points** (m) | **Weight value for distance** | |
|---|---|---|---|---|---|---|---|---|---|
| Linear (m/s) | Rotational (rad/s) | Linear (m/s) | Rotational (rad/s) | | Linear (m) | Rotational (rad) | | Obstacles | Global Path |
| 0.25 | 0.5 | 0.05 | -0.5 | 0.8 | 0.5 | 0.05 | 0.025 | 0.6 | 0.01 |

Table 6.2: Configuration Parameters used by the `hybrid_motion_planner` node

| **Maximum Speed** | | **Minimum Speed** | | **Goal Tolerances** | |
|---|---|---|---|---|---|
| Linear (m/s) | Rotational (rad/s) | Linear (m/s) | Rotational (rad/s) | Linear (m) | Rotational (rad) |
| 0.25 | 1 | 0 | -1 | 0.5 | 0.05 |

## 6.1 Evaluation Metrics

For evaluating each test and to support the drawn conclusions about the robot's behaviour during the navigation, several metrics were obtained. The following metrics are the most appropriated to resume the platform performance along the path. For each test, a table containing all these metrics

is presented and it is used to decide which architecture better fits the navigation in indoor office environments.

- **Duration**: time, in seconds, to perform the defined trajectory;
- **Success**: classification of the test success: YES, if the test was concluded with success; NO, if the test was not concluded;
- **Collisions**: number of obstacle collisions along the path;
- **Localization Losses**: number of times that the robot lost his own localization;
- **Minimum Clearance**: minimum, maximum and mean of the minimum obstacle clearance detected by the laser range finder along the path. Each iteration, the minimum clearance corresponds to the closest obstacle point to the robot, which detected by the laser.
- **Mean Linear Speed**: the mean of the linear speed measured along the path.

## 6.2 Test Scenario 1

For the first test scenario, the robot had the simple task of navigating from a start point to a target goal, wait for five seconds and then return to the start point again. To perform this navigation, it was defined a set of intermediate goals of strategic corridor zones for simplifying the navigation. This goals correspond to points where the robot need to do 90° rotations, for instance, in corridor corners and to enter or to leave rooms. Figure 6.1 shows the experimental setup for this first test, where the main and intermediate goals are properly represented.

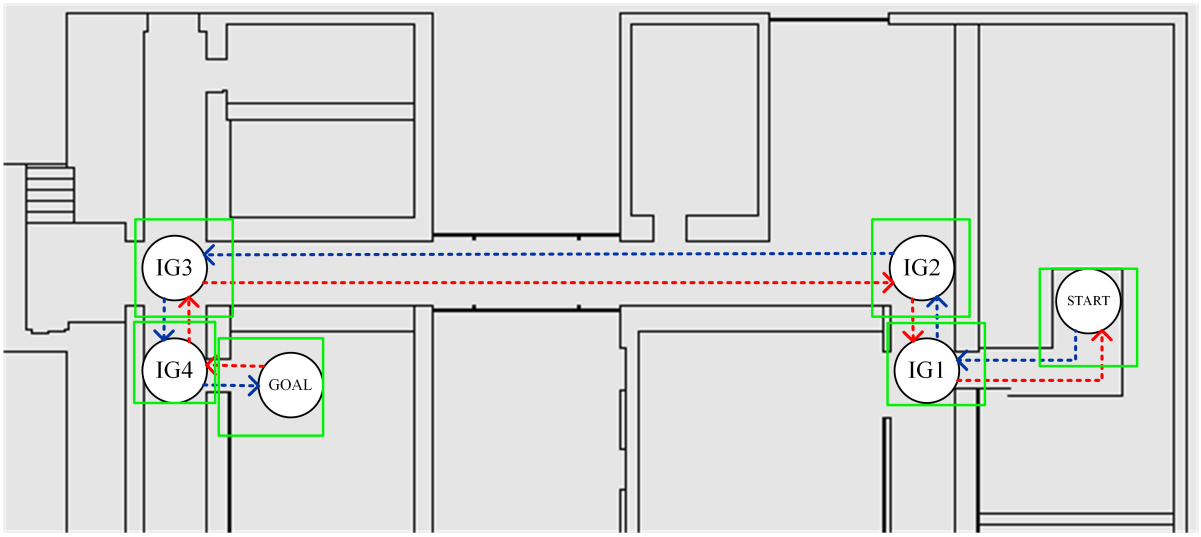



Figure 6.1: Experimental setup 1.

As it can be seen, the robot starts at point START, navigates to point GOAL passing through intermediate goals IG1, IG2, IG3 and IG4, then returns to point START again using the reverse trajectory, i.e., passing through IG4, IG3, IG2 and IG1. This trajectory has an approximated length of 71 meters. During this navigation, Interbot has to pass four times through two different doorways

and it needs to perform eight in-place 90° rotations. Those goals correspond to zones that require a most challenging path planning process and they are represented by green zones in Figure 6.1 and by pictures in Figure 6.2.

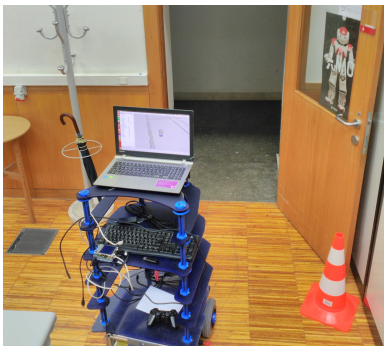
(a) START

(b) IG1

(c) IG2

(d) IG3

(e) IG4



(f) GOAL

Figure 6.2: Goal locations representation.

This test scenario was performed by Architectures 1, 2, 3 and 4, while the laser range finder sensor was fixed to the platform at 32.5 cm from the ground level as it is showed in Figure 6.3. The tests results are specified in the following subsections.

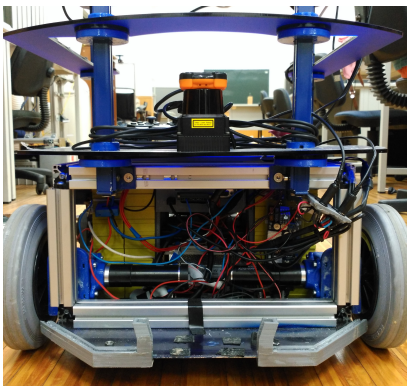


Figure 6.3: Laser range finder positioning for the tests of Scenario 1.

### 6.2.1 Architecture 1

The metrics presented at Table 6.3 show the results of the experimental tests performed by the platform using Architecture 1. The trajectory defined on the Test Scenario 1 were performed using this Architecture, achieving a success rate of 70% for a total of 10 tests. Considering only concluded tests, these were performed with an average time of 6 minutes and eight seconds, with an average of 1.57 collisions per test and 1 localization lost per test. The best obtained results are for test number 6, which had a duration of 5 minutes and 18 seconds. This test was the only one without collisions. The trajectory followed by test number 6 is represented in Figure 6.4.

Table 6.3: Metrics obtained using Architecture 1 for Scenario 1.

| **Test** # | **Duration** (mm:ss) | **Success** | **Collisions** (#) | **Localization Losses** (#) | **Minimum Clearance** (cm) | | | **Mean linear speed** (m/s) |
|---|---|---|---|---|---|---|---|---|
| | | | | | **min** | **max** | **mean** | |
| **1** | 06:20 | YES | 1 | 2 | 19 | 186.1 | 72.66 | 0.1881 |
| **2** | 06:22 | YES | 1 | 1 | 29.5 | 190.3 | 73.92 | 0.1902 |
| **3** | 03:20 | NO | 1 | 0 | 26.7 | 189.5 | 74.82 | 0.1776 |
| **4** | 05:57 | YES | 1 | 1 | 29.8 | 192.7 | 65.78 | 0.1918 |
| **5** | 05:43 | NO | 2 | 0 | 21.2 | 183.2 | 56.89 | 0.1715 |
| **6** | 05:18 | YES | 0 | 0 | 28.4 | 187.4 | 67.28 | 0.2097 |
| **7** | 06:22 | YES | 4 | 2 | 27.6 | 174 | 64.8 | 0.1759 |
| **8** | 07:01 | YES | 3 | 1 | 23.5 | 179.8 | 67.34 | 0.1623 |
| **9** | 05:38 | YES | 1 | 0 | 30.40 | 176.9 | 62.92 | 0.1901 |
| **10** | 03:11 | NO | 1 | 1 | 14.2 | 171.4 | 63.93 | 0.1517 |

The reason for the high number of collisions may be due to the fact that the trajectory planned by the `move_base` node drives the robot too close from obstacles. This trajectory planning can be an advantage if the robot needs to pass through an environment full of obstacles close to each other, but in this case, where there is enough space for the robot to drive, having a path planner that cause such a number of collisions its clearly not a acceptable option. Although the test number 6 (Figure 6.4) does not have any collision, it is a good example where the trajectory is planned too close to walls.

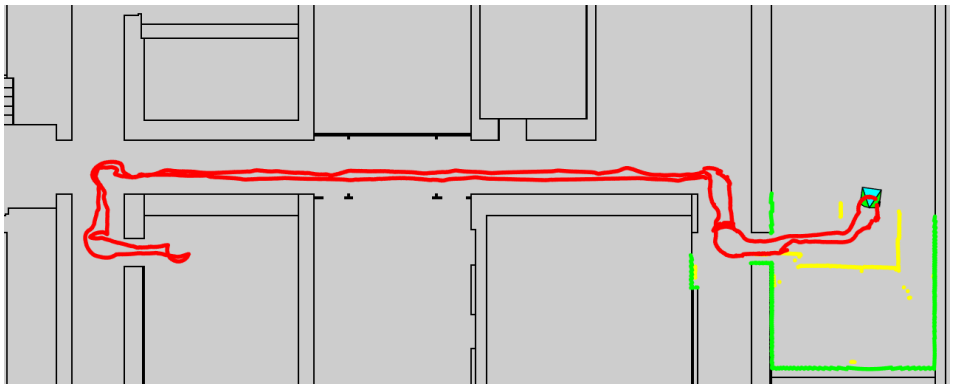


Figure 6.4: Obtained trajectory for test number 6 of Scenario 1 using Architecture 1.

### 6.2.2 Architecture 2

The obtained results for the navigation using Architecture 2 are showed in Table 6.4. Using this architecture, in a set of 15 tests, it was achieved a success rate of 73.33%. During this tests, not a single collision was recorded and an average of 0.67 localization losses . The average time to accomplish the whole trajectory was 8 minutes and 55 seconds. The test with the best performance was the test number 8, which concluded the trajectory in 7 minutes and 13 seconds and it was one of the few that had 0 localization losses along the path. Figure 6.5 represents the path followed by the robot in this specific test. Considering that, for the first 10 tests using Architecture 2, there were better results in terms of number of collisions, comparing to the Architecture 1, 5 more tests were performed, in order to confirm the tendency of not having any collisions using HMP method.

Table 6.4: Metrics obtained using Architecture 2 for Scenario 1.

| **Test** # | **Duration** (mm:ss) | **Success** | **Collisions** (#) | **Localization Losses** (#) | **Minimum Clearance** (cm) | | | **Mean linear speed** (m/s) |
|---|---|---|---|---|---|---|---|---|
| | | | | | **min** | **max** | **mean** | |
| **1** | 09:51 | YES | 0 | 1 | 33.1 | 193.4 | 86.26 | 0.0972 |
| **2** | 09:10 | NO | 0 | 2 | 15.1 | 182.1 | 69.61 | 0.1447 |
| **3** | 09:39 | YES | 0 | 2 | 34.2 | 193 | 84.42 | 0.1223 |
| **4** | 12:16 | YES | 0 | 2 | 36.1 | 191.8 | 90.24 | 0.0656 |
| **5** | 07:54 | YES | 0 | 0 | 32.9 | 192.7 | 81.98 | 0.1217 |
| **6** | 04:10 | NO | 0 | 1 | 39.2 | 175.9 | 79.94 | 0.1 |
| **7** | 08:37 | YES | 0 | 1 | 34.6 | 187.5 | 83.32 | 0.1118 |
| **8** | 07:13 | YES | 0 | 0 | 32.4 | 196 | 85.53 | 0.1396 |
| **9** | 09:28 | NO | 0 | 2 | 10.5 | 181.2 | 74.85 | 0.1263 |
| **10** | 02:27 | NO | 0 | 1 | 31.9 | 181.8 | 101.67 | 0.0535 |
| **11** | 08:24 | YES | 0 | 0 | 16.2 | 192.3 | 85.11 | 0.1120 |
| **12** | 08:54 | YES | 0 | 3 | 27.6 | 194.1 | 80.21 | 0.1036 |
| **13** | 09:09 | YES | 0 | 1 | 34.4 | 197.2 | 77.94 | 0.1095 |
| **14** | 07:49 | YES | 0 | 0 | 22.6 | 197.1 | 85.94 | 0.1275 |
| **15** | 08:22 | YES | 0 | 0 | 22.7 | 190.2 | 90.04 | 0.1107 |

The previous values show that the trajectories were more carefully planned comparing to the ones obtained using Architecture 1. This can be concluded by comparing the columns corresponding to the mean linear speed, for Table 6.3 and Table 6.4, where the speed values are significantly lower using Architecture 2. This results in an increased test duration, which can be a disadvantage if the robot is supposed to guide between to locations separated by a long distance, but considering short distances, the fact that the robot drive with low speed can be an advantage for indoor populated environments, where can be many dynamic obstacles. On the other hand, observing the column containing the mean of the Minimum Obstacle Clearance along with Figure 6.5, it can be seen that the trajectory is planned farther from the walls, which can be another advantage because the probability of colliding with a wall is lower, comparing with Architecture 1, for instance.

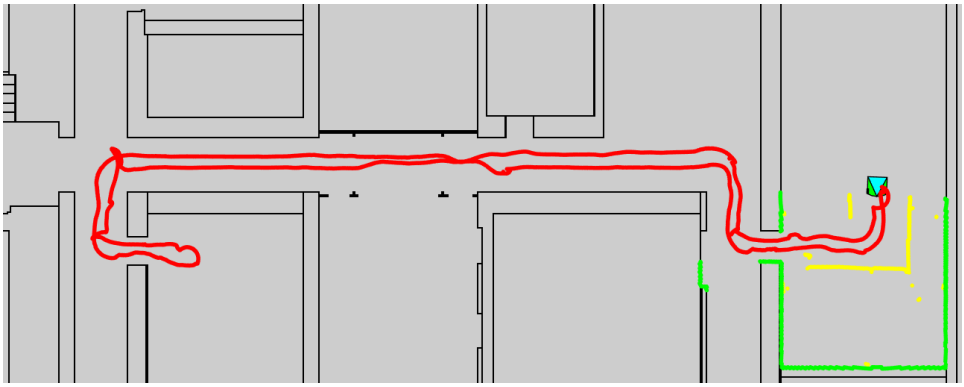


Figure 6.5: Obtained trajectory for test number 8 of Scenario 1 using Architecture 2.

### 6.2.3 Architecture 3 and 4

It was a good idea to use the `amcl_node` instead of `isr_hector_mapping` because the odometry obtained from the encoders granted an additional degree for improving the perception module. Unfortunately, after some tests of particular situations such as sharp turns and door passages, the short-term localization failed, i.e., this perception module required some time and several measures for performing a good localization. This can be a disadvantage for this type of applications, because there can be planned some trajectories with many subsequent rotations, where the robot need to almost instantaneously adapt to the environment changes. If the robot's localization fails, the planned path cannot be executed and it is not possible to navigate. Also, if there is an angular displacement between the *a priori* map and the laser readings, the environment costmaps are wrongly built and the path planner may think that it is not possible for the robot to drive through a clear zone, such as doorways or other narrow passages. Figure 6.6 represents one situation where the localization fails and there can be noticed that there is an angular displacement between the laser readings and the map provided to the robot. For this reason, from now on, Architecture 3 and 4 are completely discarded and are not considered for the purpose of this work.

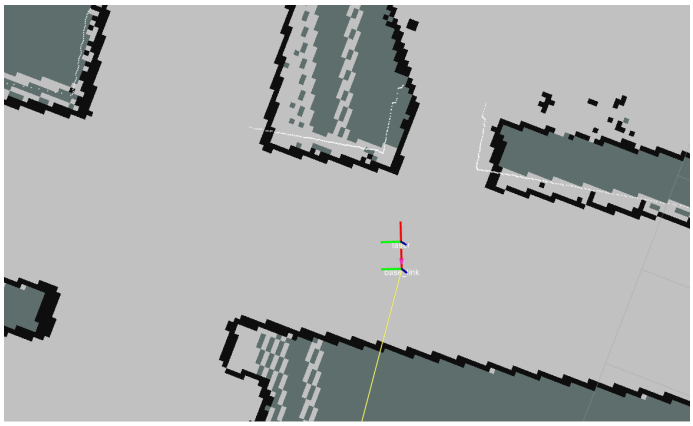


Figure 6.6: Rviz representation of a situation where the localization using AMCL fails.

### 6.2.4 Conclusions

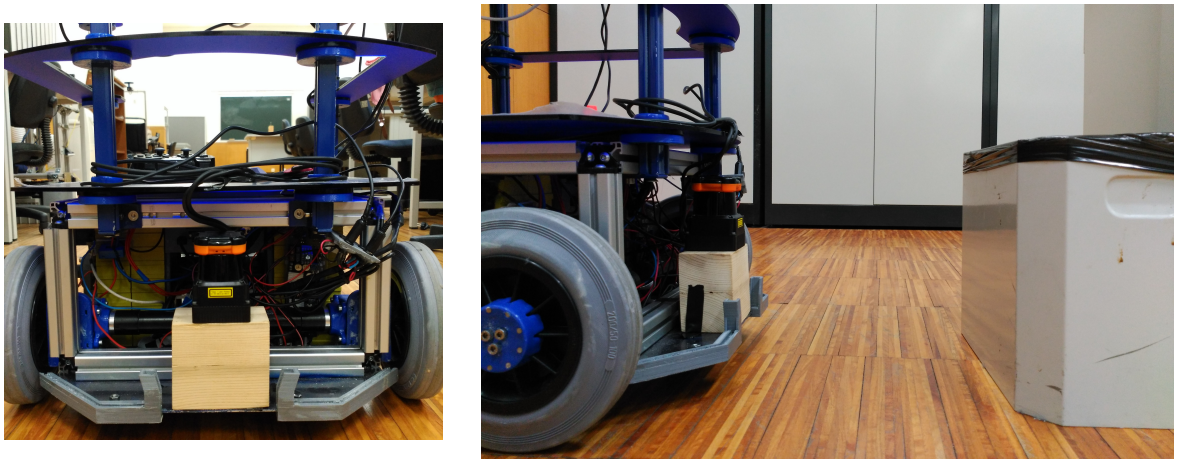
Considering that Architecture 3 and 4 were discarded, the choice of the best method to be used in this navigation system relies on the remaining two architectures. Architecture 1 had the advantage

of performing the defined trajectory faster than Architecture 2, a fact that can be confirmed by comparing the duration and mean linear speed columns of Table 6.3 and Table 6.4: the duration values are lower and the mean linear speed values are higher when using Architecture 1. In terms of number of collisions the Architecture 2 is obviously advantageous because there was not any collision during the several tests performance, unlike the Architecture 1 which had a total of 15 collisions. As it was mentioned before, this collision problem can happen due to the fact that the resulting trajectories from the Architecture 1 are planned too close to the obstacles, which do not happen when using the Architecture 2. The metrics related to the minimum obstacle clearance, in particular, the mean of the minimum obstacle clearance, can prove this fact because the values are lower in Table 6.3 comparing with the values present in Table 6.4.

According to these reasons, and considering that it is most important to have a collision free than a faster trajectory, for avoiding platform and environment damages, the chosen architecture to be integrated in the navigation system must be the Architecture 2. So, for the following tests scenarios it was only used Architecture 2.

## 6.3 Test Scenario 2

In this case, the experimental setup of Scenario 1 was repeated, with the difference of the laser range finder position. The strategy was to low the positioning of the laser to a height such that it was able to detect even more features and objects as garbage cans. So, the laser was placed as it is showed by Figure 6.7a, at 19 cm high, using a woodblock with dimensions 10cmx6cmx9cm designed for this purpose. As it can be seen in Figure 6.7b, at this high, the laser is able to detect objects such as garbage cans which can assign additional features for the environment, improving the robot's localization. The same trajectory was defined for the robot navigation, represented in Figure 6.1. To cause the desired effect on the navigation, a new map of the environment was required, in order to add those new features, so it was generated a new gridmap for being used as *a priori* map along with this new design.



(a) (b)

Figure 6.7: Laser range finder positioning for the tests of Scenario 2.

Using this improvement with System Architecture 2 augmented significantly the performance of the navigation because there was not any localization loss during the tests for the Scenario 2. Table 6.5 represents the obtained metrics for this scenario and it can be seen that all the 5 tests were concluded with success, without any collision or localization loss. Each test had an average time of 7 minutes and 38 seconds.

Table 6.5: Metrics obtained using Architecture 2 for Scenario 2.

| **Test** # | **Duration** (mm:ss) | **Success** | **Collisions** (#) | **Localization Losses** (#) | **Minimum Clearance** (cm) | | | **Mean linear speed** (m/s) |
|---|---|---|---|---|---|---|---|---|
| | | | | | **min** | **max** | **mean** | |
| **1** | 08:10 | YES | 0 | 0 | 25.7 | 193.3 | 81.45 | 0.1053 |
| **2** | 06:49 | YES | 0 | 0 | 24.3 | 196.7 | 79.26 | 0.144 |
| **3** | 07:30 | YES | 0 | 0 | 23.2 | 197 | 83.49 | 0.1304 |
| **4** | 08:22 | YES | 0 | 0 | 37 | 197.7 | 83.68 | 0.1123 |
| **5** | 07:20 | YES | 0 | 0 | 26.8 | 196.9 | 82.01 | 0.1352 |

This new laser positioning brought another advantage to the SLAM process used by Architecture 2. Due to the fact that the slam node `isr_hector_mapping` relies only on the information obtained from the laser, if the space that surrounds the robot is featureless, for instance, a long corridor with straight walls, the localization presents several failures because, in this cases, hector slam does not have a way to know that the robot is moving. A possible solution for this problem is to use the garbage cans of ISR corridors, which can add features to the environment so that the robot can obtain a good localization. Using the Scenario 1, the robot was not able to detect such objects, but in Scenario 2, as it was explained and showed in Figure 6.7b, the garbage cans were detected and could now be used for the localization process. Figure 6.8a shows the case where the laser is too high to detect the garbage cans (Scenario 1), while Figure 6.8b shows the same corridor, but with the detected garbage cans using Scenario 2 where the laser was lower.

(a) Localization without garbage cans

(b) Localization with garbage cans

Figure 6.8: Rviz representation of a featureless corridor where the robot has localization problems.

In conclusion, when the laser was lower, the robot presented significant improvements in terms

of localization, which resulted in a better navigation for specific locations, such as corridors with few features, and for trajectories that include several rotations, on which hector slam requires a map with more features so that the robot does not get lost its own localization.

## 6.4 Test Scenario 3

Having a system capable of performing a reliable navigation, it was time to deploy the robot in a controlled environment and test for how long it was able to navigate without supervision. Using Architecture 5 and modifying the `HMP_improvement`, so that it was automatically given a random new goal for the robot everytime it reached a destination, a test was made to check if it was possible to leave Interbot unattended at the ground floor of ISR, navigating through several goals. Although the test had a duration of approximately one hour, the metrics were only obtained for 25 minutes and 40 seconds, because ROS automatically stopped recording data due to the computer limited memory capacity. Table 6.6 indicates the metrics obtained for that time and, as it can be seen, there were no collisions nor localization losses.

Table 6.6: Metrics obtained using Architecture 2 for Scenario 3.

| **Test** # | **Duration** (mm:ss) | **Success** | **Collisions** (#) | **Localization Losses** (#) | **Minimum Clearance** (cm) | | | **Mean linear speed** (m/s) |
|---|---|---|---|---|---|---|---|---|
| | | | | | **min** | **max** | **mean** | |
| **1** | 25:40 | YES | 0 | 0 | 65.2 | 189.2 | 87.7 | 0.1563 |

Interbot reached its final destination after one hour, during which it reached 17 goals and have navigated approximately 505 meters. Figure 6.9 shows the driven path, and the goals that were accomplished are represented by the respective node number. Figure 5.8 helps to understand each node localization at the ground floor of ISR, were the test were conducted.

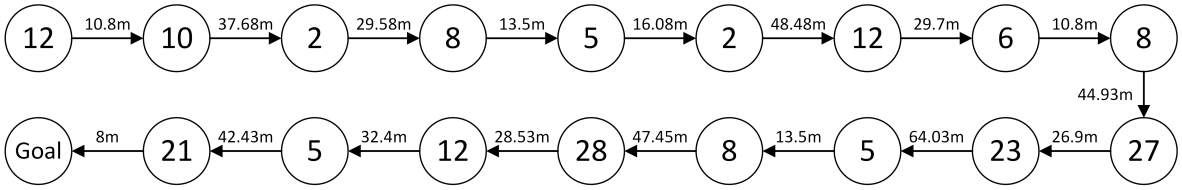



Figure 6.9: Path driven by Interbot during the test scenario 3. The distance between each node is showed above the arrows.

There are some factors that are need to be taken into account, such as limits on the computer memory capacity and on the lithium batteries that power the whole system. For being able to travel for several hours and long distances, Interbot needs to have the ability of charging itself or to inform the user that it needs to be charged. Furthermore, in this specific test, the robot did not have any navigation problem because the environment where the test were performed were static and there were not any dynamic obstacles. Although this was not the purpose of this work, if there were people or other dynamic obstacles the navigation would not be such perfect.

# Chapter 7

# Conclusion and Future Work

## 7.1 Conclusion

This work was focused on building the best framework to be applied to Interbot's navigation system in order to make it possible to perform long-term navigations in indoor populated office like environments. Interbot has never been used before for navigation purposes, so the challenge of developing a ROS based framework for accomplishing this objective was completed, as it was concluded in Chapter 6, although there were some problems with localization and path planning methods, which limited the situations where this navigation system could be tested. The purpose of this work was to develop an architecture to be applied at Interbot, taking into account static environments. So, for the path planning method, the HMP was chosen instead of move_base because, for the first case, the path was planned farther from obstacles than for the second case. This can be an advantage if there are not people or dynamic obstacles obstructing the path, but in populated environments, planning a path in the middle of the corridor is not such a good idea. On the other hand, a trajectory planned too close from walls can cause collisions with a higher probability, which is a disadvantage as well.

For accomplishing this main goal, several sub-goals were concluded were:

- Studied different methods for localization, mapping and path planning;
- Adapted the HMP framework integrated in Collabnav node [1] into Interbot;
- Integrated several architecture systems for navigation;
- Implemented an integrated navigation architecture, in ROS environment, composed by the selected methods;
- Tested the implemented navigation architecture, to confirm that a long-term navigation is possible to accomplish using Interbot.

At the end of this work, Interbot has now an integrated navigation system that has a large improvement margin and that can be used at several applications in indoor office environments.

## 7.2 Future work

Starting with the SLAM method, which has some problems in terms of localization that need to be solved, thus the path planning can be more efficient and cause fewer failures. Obviously, people and dynamic obstacles must be included in the robot's surroundings, so path planning strategies to safely avoid these types of obstacles, need to be studied. The social component for this type of systems need to be studied as well, so that Interbot can know how to behave in situations were people are involved. Perform a thorough set of tests to the HMP method, so that it can be published in the ROS community. Finally, a better user interface can be developed in order to make the Interbot more appealing and intuitive from the user point of view.

# Chapter 8

# Bibliography


[1] A. C. Lopes, J. Rodrigues, J. Perdigão, G. Pires, and U. Nunes, "A new hybrid motion planner applied in a brain-actuated robotic wheelchair," *IEEE Robotics and Automation Magazine (in press)*.

[2] S. Thrun and A. Bücken, "Integrating grid-based and topological maps for mobile robot navigation," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 944–951, 1996.

[3] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 249–265, 1987.

[4] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI magazine*, vol. 9, no. 2, p. 61, 1988.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[6] A. Lopes, *Mobile Robot Assisted Navigation based on Collaborative Control*. PhD thesis, Department of Electrical and Computer Engineering, University of Coimbra, 2012.

[7] H. F. Durrant-Whyte, "An autonomous guided vehicle for cargo handling applications," *The International Journal of Robotics Research*, vol. 15, no. 5, pp. 407–440, 1996.

[8] I. J. Cox and S. L. Hingorani, "An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 18, no. 2, pp. 138–150, 1996.

[9] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 155–160, IEEE, 2011.

[10] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *Robotics, IEEE Transactions on*, vol. 23, no. 1, pp. 34–46, 2007.

[11] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2432–2437, IEEE, 2005.

[12] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg, "6d slam-preliminary report on closing the loop in six dimensions," in *In Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV'04), Lisabon*, Citeseer, 2004.

[13] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Aaai/iaai*, pp. 593–598, 2002.

[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[15] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, pp. 2383–2388, IEEE, 2002.

[16] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[17] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning a*," *Artificial Intelligence*, vol. 155, no. 1, pp. 93–146, 2004.

[18] S. Koenig and M. Likhachev, "D* lite," in *AAAI/IAAI*, pp. 476–483, 2002.

[19] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[20] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[21] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1986–1991, IEEE, 2007.

[22] D. Vanhooydonck, E. Demeester, A. Hüntemann, J. Philips, G. Vanacker, H. Van Brussel, and M. Nuttin, "Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model," *Robotics and Autonomous Systems*, vol. 58, no. 8, pp. 963–977, 2010.

[23] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, pp. 341–346, IEEE, 1999.

[24] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *Aaai/iaai*, pp. 11–18, 1998.

[25] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, and S. Thrun, "The mobile robot rhino," *AI Magazine*, vol. 16, no. 2, p. 31, 1995.

[26] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, *et al.*, "Minerva: A second-generation museum tour-guide robot," in *Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on*, vol. 3, IEEE, 1999.

[27] R. Siegwart, K. O. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, *et al.*, "Robox at expo. 02: A large-scale installation of personal robots," *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 203–222, 2003.

[28] K. O. Arras, J. A. Castellanos, and R. Siegwart, "Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2, pp. 1371–1377, IEEE, 2002.

[29] G. Kim, W. Chung, K.-R. Kim, M. Kim, S. Han, and R. H. Shinn, "The autonomous tour-guide robot jinny," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4, pp. 3450–3455, IEEE, 2004.

[30] W. Chung, G. Kim, M. Kim, and C. Lee, "Integrated navigation system for indoor service robots in large-scale environments," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5, pp. 5099–5104, IEEE, 2004.

[31] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon," 2010.

[32] J. Biswas and M. M. Veloso, "Localization and navigation of the cobots over long-term deployments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1679–1694, 2013.

[33] I. Ferreira, "Depth camera based image processing for indoor mobile robot localization and navigation," Master's thesis, Department of Electrical and Computer Engineering, University of Coimbra, 2016.

[34] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers, "A hybrid collision avoidance method for mobile robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2, pp. 1238–1243, IEEE, 1998.

[35] D. F. W. B. S. Thrun, D. Fox, and W. Burgard, "The dynamic window approach to collision avoidance," *IEEE Transactions on Robotics and Automation*, vol. 4, p. 1, 1997.

[36] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 802–807, IEEE, 1993.

[37] K. Magzhan and H. M. Jani, "A review and evaluations of shortest path algorithms," *International journal of scientific & technology research*, vol. 2, no. 6, 2013.

[38] A. Conceição, D. Pereira, and U. Nunes, "Assisted mobility supported by shared-control and advanced human-machine interfaces," tech. rep., Department of Electrical and Computer Engineering, University of Coimbra.

[39] D. Gonçalves, "Robchair 2.0: Simultaneous localization and mapping and hardware/software frameworks," Master's thesis, Department of Electrical and Computer Engineering, University of Coimbra, 2013.

[40] J. Perdigão, "Collaborative-control based navigation of mobile human-centered robots," Master's thesis, Department of Electrical and Computer Engineering, University of Coimbra, 2014.