Daniel Pinto Correia

Navegação Semântica Aplicada a Passagens Estreitas em Cadeira de Rodas Inteligente

Semantic Navigation Applied to Narrow Passages for Intelligent Wheelchairs

Universidade de Aveiro
2022

Daniel Pinto Correia

Navegação Semântica Aplicada a Passagens Estreitas em Cadeira de Rodas Inteligente

Semantic Navigation Applied to Narrow Passages for Intelligent Wheelchairs

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Artur José Carneiro Pereira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Eurico Farinha Pedrosa, Investigador Doutorado (Nível 1) do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Esta dissertação contou com o apoio do projeto IntellWheels 2.0 com a referência POCI-01-0247-FEDER-39898 e foi financiada pelo Fundo Europeu de Desenvolvimento Regional (FEDER)

o júri / the jury

presidente / president

Professor Doutor José Luís Costa Pinto de Azevedo
Professor Auxiliar do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Luís Paulo Gonçalves dos Reis
Professor Associado do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Artur José Carneiro Pereira
Professor Auxiliar do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro

agradecimentos / acknowledgements

Desejo agradecer a todos os que contribuíram para que este trabalho fosse possível. Em primeiro lugar quero agradecer ao Professor Doutor Artur José Carneiro Pereira por ter aceite ser orientador da minha dissertação e pelo apoio sempre presente desde o início. Agradeço ainda por todas as oportunidades de aprendizagem que me proporcionou durante estes meses de trabalho. Agradeço igualmente ao Doutor Eurico Farinha Pedrosa e ao Professor Doutor José Nuno Panelas Nunes Lau pelos contributos essenciais para a conclusão desta dissertação. Quero agradecer aos meus amigos e colegas de curso, em particular àqueles que sempre me apoiaram, não só durante a dissertação mas durante estes últimos 5 anos. Um especial agradecimento ao João Rodrigues, José Silva, João Abrantes e Luisa Vale que ativamente me ajudaram em desafios no desenvolvimento ou na escrita da dissertação. Aos meus pais e ao meu irmão: por sempre me incentivarem na prossecução dos estudos que me cativaram desde início, e que ainda hoje me apresentam novos e interessantes desafios e pelos sacrifícios pessoais que permitiram que hoje os concluísse.

Palavras Chave

Navegação Semantica, Planeamento, Mapeamento, Passagens Estreitas, ROS, Cadeira de Rodas Inteligent, Navegação em ambiente interior

Resumo

Com o desenvolvimento da tecnologia em robótica, surgem novas oportunidades para melhorar a qualidade de vida de pessoas com problemas de mobilidade. O projeto IntellWheels nasce com o objetivo de criar um kit de hardware e software capaz de transformar uma cadeira de rodas motorizada numa cadeira de rodas inteligente e autónoma. Esta dissertação encaixa neste projeto no tópico de navegação em ambientes interiores com o objetivo de adicionar uma camada semântica à framework de navegação.

Em robótica, semântica é a capacidade de um robô entender o seu ambiente. No caso de uma cadeira de rodas inteligente, isto é especialmente importante tendo em conta que transporta um passageiro. Uma solução foi desenvolvida em que conceitos de navegação semântica são usados para abordar o problema de atravessar passagens estreitas, construindo uma camada adicional de mapeamento em que estas passagens são automaticamente identificadas e marcadas no mapa e uma camada extra de planeamento que recebe esta informação e toma decisões controlando os planeadores de nível mais baixo.

Para validar esta solução, foram realizados testes, em ambiente simulado e ambiente real para verificar se os componentes individuais funcionavam como pretendido.

Esta dissertação resultou numa melhoria das capacidades de navegação em ambientes interiores da framework e é uma base para futuros desenvolvimentos na área de navegação semântica para cadeiras de rodas inteligentes.

Keywords Semantic Navigation, Planning, Mapping, Narrow Passages, ROS, Intelligent Wheelchair, Navigation in indoor environment

Abstract

With the development of robotics technology, new opportunities for improving the quality of life of people with mobility impairments arise. The IntellWheels project was born with the goal of developing a hardware and software kit that can turn a motorized wheelchair into an autonomous Intelligent Wheelchair. This dissertation fits into this project on the topic of indoor navigation with the goal of adding a semantic layer to the navigation framework.

Semantic in robotics is the ability of a robot to understand its environment. In the case of an Intelligent Wheelchair, this is especially important considering it is a robot that carries a passenger. A solution was developed where concepts of semantic navigation are used to tackle the problem of crossing narrow passages, building an additional mapping layer where these passages are automatically detected and marked on the map and an extra planning layer that takes in this semantic information and makes decisions for controlling the lower-level planners.

In order to validate this solution tests, in both simulated and real-world environments, were performed to verify if the individual components worked as intended. This thesis resulted in an improvement of the indoor navigation capabilities of the framework and is a base for further development in the area of semantic navigation of Intelligent Wheelchairs.

Contents

Contents i

List of Figures v

List of Tables vii

Glossary ix

1 Introduction 1

1.1 Motivation . . . 1

1.2 Objectives . . . 2

1.3 Document Structure . . . 2

2 Background and related work 5

2.1 Navigation . . . 5

2.1.1 World Representation and Mapping . . . 6

2.1.2 Localization . . . 10

2.1.3 Path Planning . . . 10

2.1.4 Plan Execution . . . 10

2.2 Robot Operating System . . . 11

2.2.1 Concepts . . . 11

2.2.2 Tools . . . 13

2.3 Intellwheels 2.0 . . . 15

2.3.1 Sensorization . . . 15

2.3.2 Odometry . . . 16

2.3.3 SLAM . . . 17

2.3.4 Costmaps . . . 17

2.3.5 Global Planning . . . 17
2.3.6 Local Planning . . . 17
2.4 Semantic Navigation . . . 18
2.4.1 Mapping . . . 18
2.4.2 Planning . . . 19
2.5 Semantic Navigation in Intelligent Wheelchairs . . . 20

3 Semantic Navigation for Intelligent Wheelchairs 21
3.1 Proposed System Architecture . . . 22
3.2 Semantic Plan Cost . . . 23
3.3 Semantic Mapper: Automatic Narrow Passage Marking . . . 24
3.3.1 Distance Map . . . 25
3.3.2 Extended Voronoi Graph . . . 26
3.3.3 Passage Segmentation . . . 27
3.3.4 Passage Boundaries and Approach Poses . . . 29
3.3.5 Narrow Passage Comfort Cost . . . 31
3.3.6 Narrow Passage Representation . . . 31
3.4 Semantic Planning . . . 32
3.4.1 Passage Map . . . 33
3.4.2 Passage Crossing Detection . . . 34
3.4.3 Path Splitting . . . 35
3.4.4 Semantic Plan Cost . . . 36
3.4.5 Path Execution . . . 37

4 Experiments and Results 39
4.1 Assessing the Semantic Mapper . . . 39
4.2 Assessing the Semantic Planning . . . 40
4.3 Validity of the semantic planner functionality . . . 43
4.3.1 Consecutive Passages Scenario . . . 44
4.3.2 Parallel Approach Scenario . . . 46
4.3.3 Narrow Corridor Scenario . . . 47
4.3.4 Evaluation of parameters for narrow passage crossing . . . 49
4.4 Influence of Comfort Cost in semantic planning . . . 50

5 Conclusion **53**

5.1 Discussion 53

5.2 Limitations 54

5.3 Future Work 55

References **57**

List of Figures

2.1 Occupancy Grid 6
2.2 Example of Topological and Topometric maps based on an occupancy grid of the map assembled at IRIS-Lab 7
2.3 GVG and EVG examples 8
2.4 Euclidean distance map example 9
2.5 Cost Map Example. 10
2.6 Internode communication in ROS 12
2.7 RViz visualization tool 13
2.8 Stage simulator with IRIS-Lab's map 14
2.9 Wheelchair footprint and LIDAR placement. 16
2.10 Mapping layers example based on an occupancy grid of the map assembled at IRIS-Lab 19

3.1 Conceptual Architecture 22
3.2 Narrow Passage with respective boundaries and approach poses 23
3.3 IW crossing a narrow passage 25
3.4 Distance Map Example 26
3.5 EVG for the IRIS-Lab's map 27
3.6 Segmentation of the EVG into Passages 29
3.7 Narrow Passage with Boundaries and Approach Poses 30
3.8 Critical Passage Message Structure 31
3.9 Semantic planner flowchart 33
3.10 Path cost evaluation process. 37

4.1 Results from Narrow Passage marking for the map at the IRIS-Lab. 40

4.2 Results from Narrow Passage marking for the map of a building at the University of Freiburg. 40
4.3 Photograph of the experiment scenario assembled at IRIS-Lab. 41
4.4 IW in the starting position for tests 42
4.5 Scenarios used for semantic planner functionality tests. 43
4.6 Results of plan division performed by the Semantic Planner 44
4.7 Paths taken by the wheelchair for the consecutive passages scenario. 45
4.8 Paths taken by the wheelchair for the parallel approach scenario. 47
4.9 Paths taken by the wheelchair for the narrow corridor scenario. 48
4.10 Scenario for testing the influence of comfort cost in semantic planning 51
4.11 Results of the tests of the influence of comfort cost in semantic planning. 52

List of Tables

2.1 LIDAR sensors characteristics. 16

4.1 Results for the consecutive passages scenario of tests for validation of the Semantic Planner 46

4.2 Results for the parallel approach scenario of tests for validation of the Semantic Planner 46

4.3 Results for the narrow corridor scenario of tests for validation of the Semantic Planner 49

4.4 Results of evaluation of parameters for narrow passage crossing 50

Glossary

ROS	Robot Operating System
TEB	Timed Elastic Band
SLAM	Simultaneous Localization and Mapping
LaMa	Localization and Mapping
LiDAR	Light Detection and Ranging
IW	Intelligent Wheelchair
GVG	Generalized Voronoi Graph
EVG	Extended Voronoi Graph
OG	Occupancy Grid
IRIS-Lab	Intelligent Robotics and Intelligent Systems Laboratory

CHAPTER 1

Introduction

"Starting now and lasting until forever, your health and healthcare will be determined, to a remarkable and somewhat disquieting degree, by how well the technology works."
Robert Wachter

1.1 Motivation

Intelligent mobile robotics is ever more present all around us and the use of robotics to improve the quality of life for the elderly or people with disabilities is a growing field of study. In this context, intelligent wheelchairs play an important role to improve the quality of life for people suffering from disabilities that affect their mobility. A Intelligent Wheelchair (IW) improves on the capabilities of a conventional electric wheelchair by removing the responsibility of the controls from the user, which can make a huge difference for people without the dexterity to use the controls of an electric wheelchair.

With the ambitious objective of developing a hardware and software kit for converting a conventional electric wheelchair into an IW, the IntellWheels project was born in 2007. In 2019 the IntellWheels 2.0 project was created to improve and extend the work developed in the previous version.

In 2021, Figueiredo [1] developed a platform for indoor navigation for the IntellWheels 2.0 project and this dissertation comes as a continuation of this work. The navigation of a wheelchair presents particular challenges compared to other mobile robots that do not carry a passenger and the work described in this dissertation tackles specific problems identified in previous work focusing on creating a navigation solution that best suits the objectives of an IW.

1.2 Objectives

This dissertation aims to improve the quality of navigation in indoor spaces for the Intellwheels 2.0's Framework. The navigation of an Autonomous IW presents additional challenges compared to other types of autonomous mobile robots, mainly on the count of having to consider the presence of a passenger. These challenges include things like the importance of the orientation of the wheelchair on a trajectory. While a robot without a passenger can take the most convenient orientation to get to a goal, with an IW, it would not make sense, for example, to travel backwards for more than short distances.

Previous work on the Intellwheels 2.0 indoor navigation framework identified a few particular problematic situations that remain to be solved, those situations including:

- Reliably and comfortably crossing narrow passages.
- Approaching very close to a wall (to reach a light switch for example).
- Placing the wheelchair partially under a table (to sit at the table for example).

Each of these situations has its own specificity and a unified approach to solve all of them might not be the best strategy. The main objective of this dissertation is to add to the IntellWheels 2.0's indoor navigation framework a new layer of mapping and planning. This new layer uses concepts of semantic navigation to identify problematic situations and takes specific strategies to solve them.

Focusing on the narrow passages situation, this new layer is responsible for identifying and marking the narrow passages on a map. This semantic information is then used to achieve the following objectives:

- Increase the ability to cross narrow passages.
- Provide a greater level of comfort to the user while crossing narrow passages.

Besides improving the performance of the framework when it comes to crossing narrow passages, this dissertation aims to build the basis for a semantic navigation layer that can be extended to also consider other situations.

1.3 Document Structure

This document covers the implementation of a solution to achieve the previously defined goals as well as the research that went into understanding the concepts necessary for this implementation. Several experiments and respective results are also presented and discussed reflecting on future work possibilities.

- **Chapter 2:** Presents the state of the art in autonomous robot navigation, including traditional navigation topics as well as semantic mapping and planning

topics, and a brief description of the tools and frameworks used. In this chapter, the context in which this work was developed as well as the previous work developed by Figueiredo and related work in semantic navigation for intelligent wheelchairs is also presented.

- **Chapter 3:** This chapter describes the system implemented. Firstly the architecture is laid out and then the individual components are described and explained.
- **Chapter 4:** Presents the experiments performed and respective results, along with a discussion of these results.
- **Chapter 5:** The obtained results are critically discussed and a reflection on possible improvements, different strategies and future work is presented. Final thoughts and conclusion of the dissertation are presented in this chapter.

CHAPTER 2

Background and related work

In this chapter, topics of autonomous mobile robot navigation are presented as they are relevant to the subject of this dissertation. Special attention is given to topics of semantic navigation since it is the main focus of this work. Other tools and concepts used in the development of this dissertation are also explained in this chapter. The context in which the work developed for this dissertation was done is explained. The previous work on which this work is built upon in presented. Finally, a brief overview of previous works regarding semantic navigation in intelligent wheelchairs is presented.

2.1 Navigation

Navigation in autonomous mobile robots is a set of abilities a robot possesses that allows it to move in its surrounding environment. These abilities must give the robot a way to represent the environment it is located in, localise itself in this environment, plan the strategy to get to a different position and execute this strategy without colliding with obstacles. For this, there are 4 main concepts that navigation is built upon.

- **Mapping** - This is usually done through sensors mounted on the robot, but can be acquired in different ways. The robot must have a meaningful way of representing the surrounding world so it can perform the other tasks of navigation.
- **Localization** - Is the ability of having the knowledge of where the robot is located in its environment.
- **Planning** - Having the representation of the world and knowing its location within it, the robot can then make a plan of how to get to a goal position, taking the form of a trajectory to be followed.

- **Plan Execution** - This is the part responsible for actually moving the robot, it computes velocities and changes the trajectory planned in order to conform to dynamic obstacles that may not be previously mapped.

2.1.1 World Representation and Mapping

World representation can take many different forms that are each suited for different applications. This section focuses on the world representation methods that are relevant for the work developed for this dissertation.

Occupancy Grid

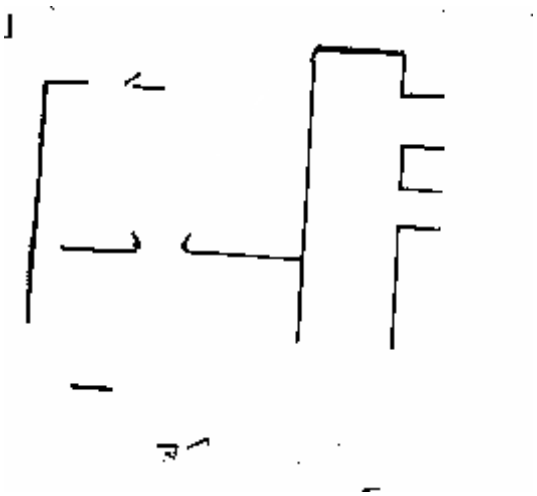
Occupancy Grid (OG), firstly proposed by Elfes [2], is a simple, widespread way to represent an indoor space in robotics. An OG consists of a 2D image that probabilistically represents a physical space by dividing it into a discrete space of cells. For given coordinates, a cell can be empty, occupied or unknown. An example can be seen in figure 2.1. This method gives a simple representation of the world that can be used for path planning without too much computational cost. It's not a perfect representation though, as it leaves out three-dimensional features.

The resolution of an OG can greatly impact both the precision of the representation and the computational cost of performing calculations on it. These two factors must be balanced also considering the precision of the sensors used to create the OG.

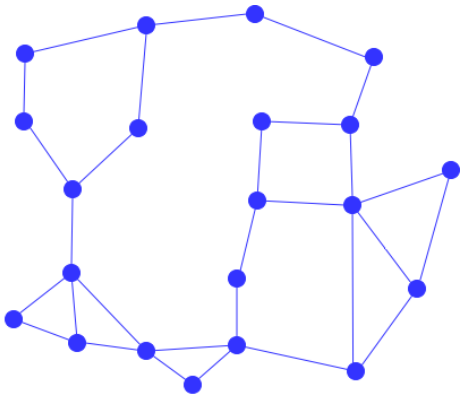
Figure 2.1: Occupancy Grid recorded in a building at University of Freiburg [3].

Topological and Topometric Maps

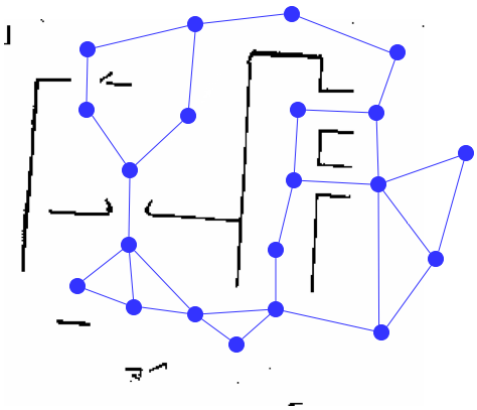
A topological map is represented by a graph composed of nodes that represent places and edges that represent a spacial relation between places [4]. A topological map leaves out the geometric detail of a metric map, necessary for navigation. Therefore, the concept of a topometric map as described in [5] and [6] is more adequate since it makes an association between the topological graph and the metric map. In practice, this is done by giving each node in the topological graph the respective coordinates in the metric map. A representation of these maps can be seen in figure 2.2.



(a) Metric map, as an occupancy gird



(b) Topological map



(c) Topometric map, a combination of the metric and topological maps

Figure 2.2: Example of Topological and Topometric maps based on an occupancy grid of the map assembled at Intelligent Robotics and Intelligent Systems Laboratory (IRIS-Lab)

Generalized Voronoi Graph

Choset and Nagatani [4] use the Generalized Voronoi Graph (GVG) that they define as "the one-dimensional set of points equidistant to m obstacles in m dimensions" to extract the topological map from an occupancy grid. For the plane, this can be specified as the set of points equidistant to two obstacles such that for each point there is no other obstacle at an inferior distance to this point, as defined in equation 2.1.

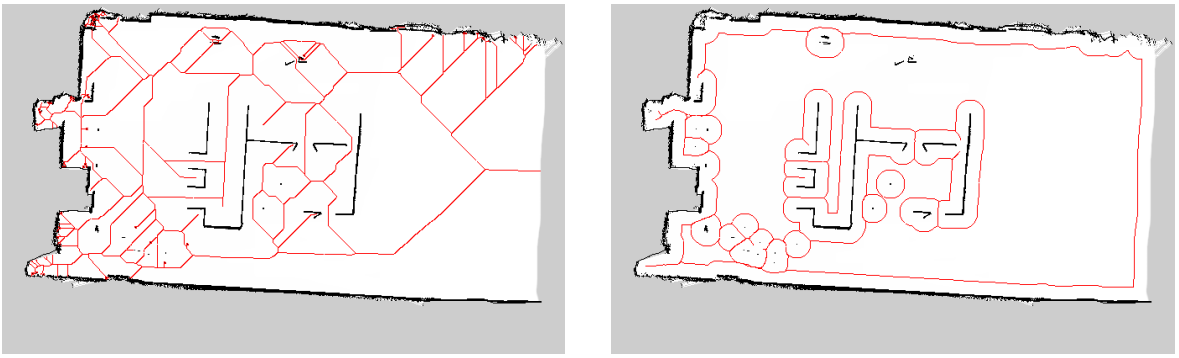
$$\begin{aligned} P = \{p : \exists o, o' \in O, \text{ where } o \neq o' \wedge dist(p, o) = dist(p, o') \\ \wedge \forall o'' \in O \, dist(p, o'') \geq dist(p, o)\} \end{aligned} \tag{2.1}$$

where p is a point, O is the set of all obstacles and $dist(p, o)$ represents the minimal distance of point p to obstacle o.

Later work by Beeson *et al.* [7] defined the Extended Voronoi Graph (EVG) as "the subset of all points in the GVG closer than a threshold of M units from any obstacle, added to the set of all points exactly M units away from the closest obstacle", as defined in equation 2.2.

$$\begin{aligned} Q = \{q : \exists o, o' \in O, \text{ where } o \neq o' \wedge dist(q, o) = dist(q, o') \\ \wedge \; dist(q, o) \leq M \wedge \forall o'' \in O \, dist(q, o'') \geq dist(q, o)\} \\ \cup \\ \{q' : \exists o \in O, \text{ where} dist(q', o) = M \\ \wedge \forall o' \in O, \; dist(q', o') \geq M\} \end{aligned} \tag{2.2}$$

where q is a point, O is the set of all obstacles and $dist(q, o)$ represents the minimal distance of point q to obstacle o.



(a) Generalized Voronoi Graph of the occupancy grid of the map assembled at IRIS-Lab.

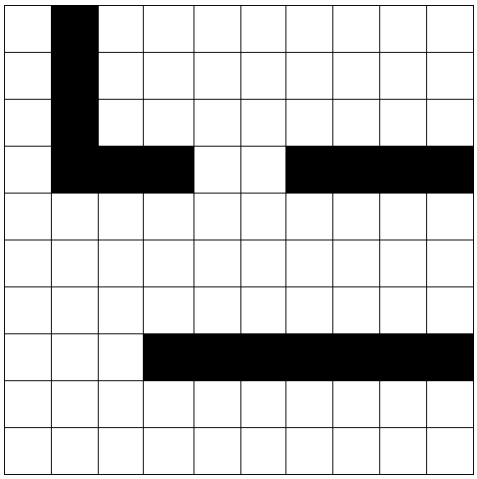
(b) Extended Voronoi Graph of the occupancy grid form the IRIS-Lab map.

Figure 2.3: GVG and EVG examples. Points in red correspond to the GVG and the EVG in each image respectively and the remaining image corresponds to the Occupancy Grid

Beeson *et al.*, as well as Choset [8][9], propose the use of Voronoi graphs for topological map-building by discretising the continuous environment into a finite set of places. Examples of both GVG and EVG can be seen in figure 2.3.

Distance Maps

Given an image full of 0's and 1's, the distance map or distance transform [10] of that image is a 2D image where each pixel contains a value equal to the distance from that pixel to the nearest 0. Distance maps can be calculated using different distance metrics. For the purpose of this dissertation, euclidean distance is used for calculating distance maps. An example of a distance map is represented in figure 2.4



(a) Example image where the black squares correspond to 0's

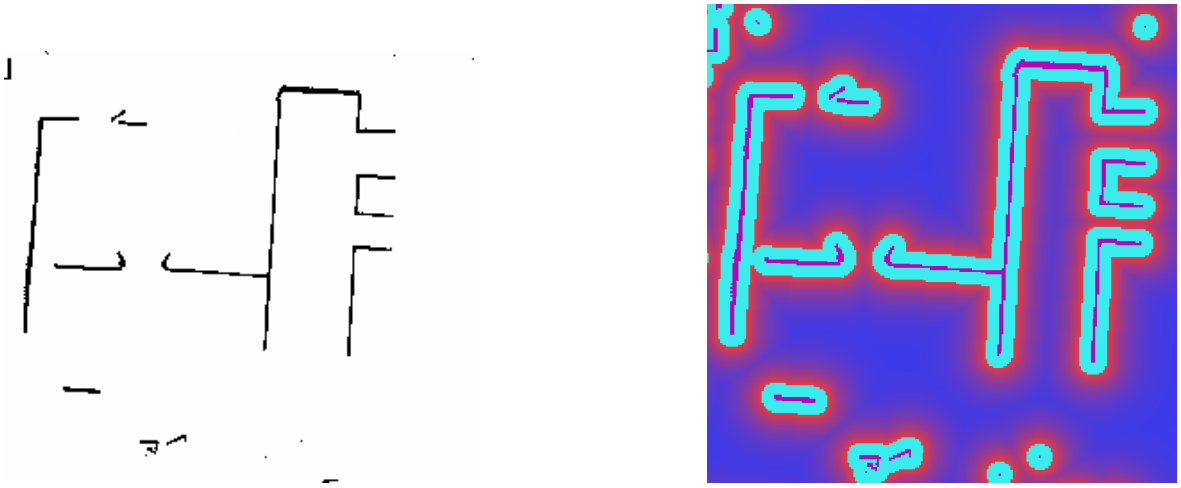
1	0	1	2	3	3.2	3	3	3	3
1	0	1	2	2.2	2.2	2	2	2	2
1	0	1	1	1.4	1.4	1	1	1	1
1	0	0	0	1	1	0	0	0	0
1.4	1	1	1	1.4	1.4	1	1	1	1
2.2	2	2	2	2	2	2	2	2	2
3.2	2.2	1.4	1	1	1	1	1	1	1
3	2	1	0	0	0	0	0	0	0
3.2	2.2	1.4	1	1	1	1	1	1	1
3.6	2.8	2	2	2	2	2	2	2	2

(b) Euclidean distance map from figure 2.4a

Figure 2.4: Euclidean distance map example

Cost Maps

Cost maps represent physical space in a discrete way where each point has an associated cost. In the context of navigation in mobile robotics, cost maps are used for representing the freedom a robot has to move in each location. Points that are far from objects have a low-cost value as the robot can move freely, as opposed to points that are coincident with objects, which have an infinite cost. Areas close to objects can have an intermediary cost representing the possibility of navigation but with a higher difficulty than completely free space. These maps are used for planning trajectories and different planners can use them differently according to the characteristics of the robot and requirements in its use. An example of these maps can be seen in figure 2.5.



(a) Section of the occupancy grid of the map assembled at IRIS-Lab.

(b) Cost map of the occupancy grid in figure 2.5a.

Figure 2.5: Cost Map Example.

2.1.2 Localization

The localization method depends on the sensorization of the robot. The localization algorithm tries to overlap the reading from the sensors with the known map to find the location of the robot.

Associated with localization, there is the concept of **Odometry**. Odometry is the measure of change in position over time. This is measured through sensors and can be used by the localization algorithm to help keep the location up to date while the robot is moving.

2.1.3 Path Planning

The path planning phase is responsible for finding a path, in a known map, between two points. The module responsible for this is usually called the Global Planner, in contrast with the Local Planner which aids in the plan execution phase.

The global planner runs a path search algorithm on the whole map, considering also the specific characteristics of the robot, to find a path from the current position to the target position. The goal of the global planner is to find the shortest path between the two positions that is possible for the robot to follow.

2.1.4 Plan Execution

Plan execution is responsible for generating velocity vectors to send to the robot in order to follow the planned path. To do this, there is a level of planning required since the original path from the global planning phase does not consider dynamic objects that are not previously mapped. Therefore, this phase is usually supported by a different planner called the local planner.

The local planner only considers the subsection of the map that is in the surroundings of the robot. For this effect, the local planner has its own cost map that considers not only the known map but also dynamic obstacles detected during plan execution time.

The goal of the local planner is to find a path from the robot to the point where the global plan intercepts the edge of the local cost map. Since this planner does not consider the whole map, it's possible the path found is not optimal. The local planner also determines the actual trajectory taken by the robot independently from the original global plan, as well as the velocity vectors applied to the robot.

2.2 Robot Operating System

In this section, Robot Operating System (ROS) is presented and the main concepts and tools used for this work are explained.

2.2.1 Concepts

According to the ROS Wiki[1], "ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management."

In other words, ROS provides an abstraction layer that is adequate for a wide range of hardware combinations. It has a modular organization focused on code reuse and collaborative research and development [11].

Here are the main concepts of ROS as defined by the ROS Wiki:

- **Node** - A Node is the smallest processing unit in ROS. It acts like a process in a typical operating system and usually performs a single purpose and communicates with other nodes.
- **Message** - Messages are data structures for node-to-node communication. They can contain simple data types (like integer, floating point or boolean) but also other messages or arrays.
- **Topic** - Topics are an asynchronous way of node-to-node communication in ROS. A publisher Node registers a topic with a message type and starts publishing messages. Any Node can subscribe to a topic and will start receiving directly the messages there published.
- **Service** - Services are a synchronous bidirectional way of node-to-node communication between a client and a server. A client requests a service with a service request message and the server responds with a service response message.

[1] http://wiki.ros.org

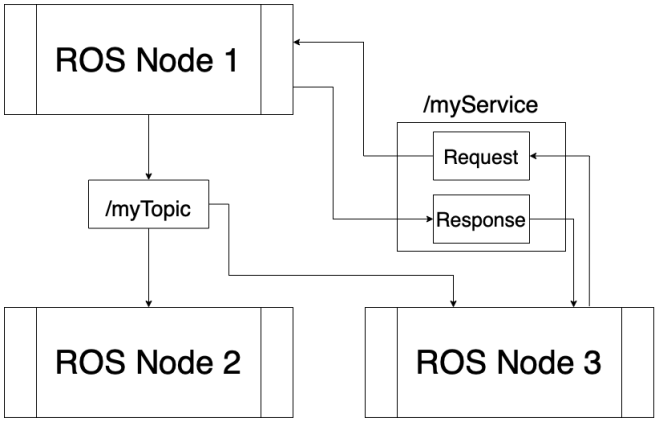


Figure 2.6: Internode communication in ROS: ROS Nodes 2 and 3 subscribe to the topic */myTopic* that is published by ROS Node 1. ROS Node 1 also provides the service */myService* that is called by ROS Node 3

- **Action** - Action is another asynchronous communication method that, unlike topics, is bidirectional. Actions are used instead of services where it takes a longer time to respond after getting the request. The action client sets the goal for the action and the action server performs the action and returns feedback and the result to the client. Actions are preemptive and can be cancelled or replaced with a different goal.
- **Package** - Software in ROS is organized in packages. Packages can contain nodes, libraries, message definitions, configuration files or anything else that might be required to run it.
- **bag** - bag is a file format to record ROS message data. A previously recorded bag file can be played at a later time which is very useful for testing.
- **Master** - The Master node acts as a directory that associates topics to nodes, enabling them to find each other and communicate.
- **Parameter Server** - It is part of the master and stores running parameters that can be changed in run time.

The relation between Nodes, Topics and Services is represented in figure 2.6. The peer-to-peer network of connected nodes is called the *Computation Graph*. This graph can be visualized by running the 'rqt_graph'[2] node in the 'rqt_graph' package which is part of the rqt set of tools explained in section 2.2.2.

ROS provides a client library[3] to support various programming languages. Currently, the main client libraries are for C++, Python and LISP programming languages.

[2]http://wiki.ros.org/rqt_graph
[3]http://wiki.ros.org/Client%20Libraries

Figure 2.7: RViz visualization tool. RViz subscribes to any ROS topics the user configures and displays them in a graphic environment. In this example, we can see the map, cost maps, the wheelchair footprint, the point cloud of the readings from the Light Detection and Ranging (LiDAR) sensors and some other topics listed on the left.

2.2.2 Tools

Visualization Tools

rqt is a framework for GUI development for ROS. A collection of rqt plugins is available as well as the possibility to create customized plugins. These plugins are very useful for the programmer and facilitate the visualization of concepts in the ROS environment.

RViz[4] is a 3D visualization tool for ROS. It is launched as a ROS node and can be configured to subscribe to any topics the programmer wants and display them in 3D, allowing visual verification of the data. RViz also allows to publish messages to topics, which can be used to send data to the robot, like estimating the current location or commanding a target position. In figure 2.7 we can see the RViz visualization tool.

Simulation Tools

Player and Stage are both software tools part of the *Player Project*[5]. Together they provide all the components for the simulation of mobile robots in a 2D environment. Player simulates the robot hardware and provides the robotic application with an interface to retrieve data from the sensors and give commands to the actuators.

Stage simulates a population of robots, sensors and objects in a two-dimensional environment and is often used as a Player plugin to provide the environment data to

[4]http://wiki.ros.org/rviz
[5]https://playerstage.sourceforge.net

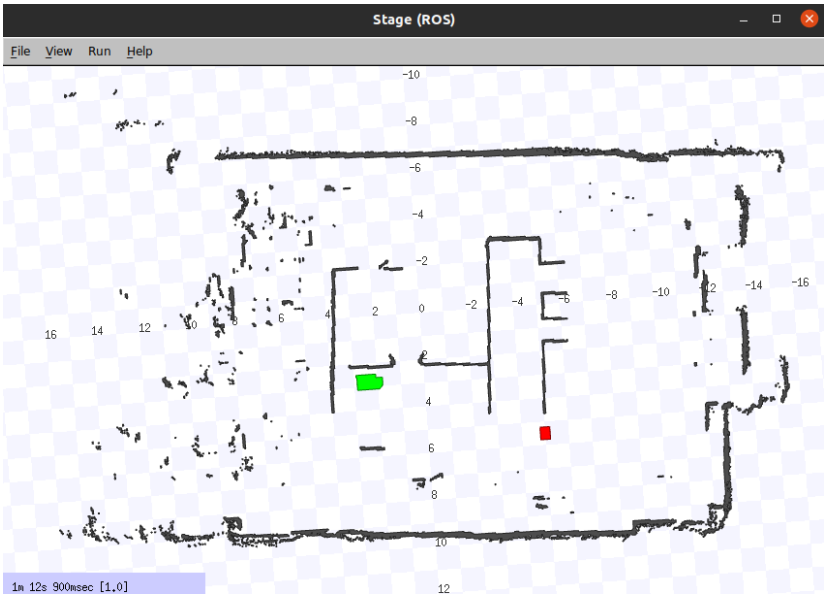


Figure 2.8: Stage simulator with IRIS-Lab's map. We can see the map, as well as the wheelchair with its footprint in green. The red square represents a dynamic obstacle that is not part of the map and can be moved in run-time.

Player. An example of a stage simulation can be seen in figure 2.8.

Gazebo[6] is a powerful 3D simulator. It allows the simulation of both indoor and outdoor spaces and of very different types of robots, making it very versatile and used a lot in robotics. The versatility and complexity of this simulator also make it computationally heavy. For this reason, the use of a simpler simulator is recommended for applications that do not require the extra features it provides.

Planning Tools

Global Planner[7] is a global planning algorithm that takes into consideration the specific dimensions and geometry of the robot and, therefore, can command direction. It provides 6 modes of operation:

- *Forward* - The robot always moves in the positive direction of the x-axis of its referential.
- *Interpolate* - The robot always moves in the direction that is most convenient to get to the goal.
- *ForwardThenInterpolate* - The robot starts moving in the positive direction of the x-axis of its referential and then takes the more convenient direction to get to the goal.
- *Backward* - The robot always moves in the negative direction of the x-axis of its referential.

[6]https://gazebosim.org/home
[7]http://wiki.ros.org/global_planner

- *Rightward* - The robot always moves in the negative direction of the y-axis of its referential.
- *Leftward* - The robot always moves in the positive direction of the y-axis of its referential.

With these available modes, it is possible to develop different navigation modes for different robots and situations.

Timed Elastic Band (TEB)[8] is a local planner responsible for global plan execution that works by deforming the global plan with attraction and repulsion forces. The repulsion is triggered by obstacles and the attraction by special points in the global plan called *via points*. TEB comes as an extension of classic *elastic band* planners by considering also the time dimension [12].

2.3 Intellwheels 2.0

The Intellwheels 2.0 project aims to research and develop a framework/kit for IW that allows the application of this concept to different types of conventional electric wheelchairs. It also aims to separate the user interface from the control system, creating a multimodal user interface that can adapt to different types of users and equipment.

In the context of this project, Universidade de Aveiro is responsible for developing the Intellwheels Framework, which consists of the hardware and software kit to be implemented in a conventional electric wheelchair, giving it autonomous navigation capabilities. For the hardware, this kit should consist of a collection of sensors and a processing unit. The software will be composed of several modules developed using ROS.

The work developed for this dissertation, was developed in the context of this project, aiming to further develop and improve the Intellwheels framework on its indoor navigation capabilities. This dissertation is built upon an already existing platform for IW navigation in indoor spaces developed by Figueiredo [1] in the context of the IntellWheels 2.0 project. This platform already includes basic navigation functionalities and is specified as follows.

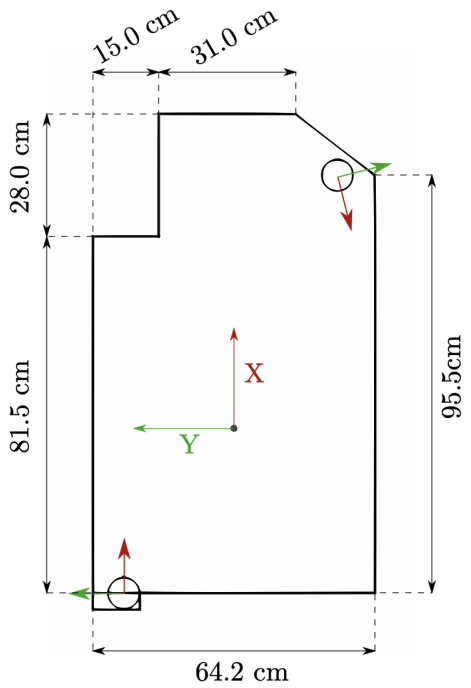
2.3.1 Sensorization

To support indoor navigation, two LiDAR sensors were placed in the IW giving a 360º view at the height of about 42 centimetres from the ground. The use of LiDAR is adequate for indoor navigation since objects and walls are typically closer to the subject than in exterior places, and the absence of sunlight means less interference in the readings.

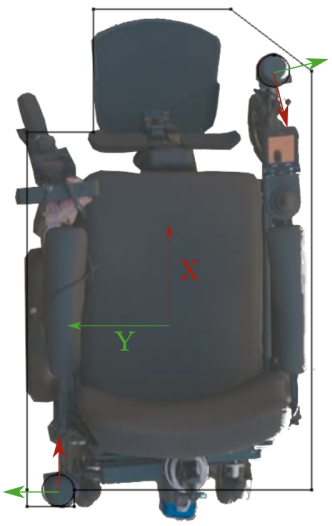
[8] http://wiki.ros.org/teb_local_planner

	RPLidar A2	RPLidar A3
Range (m)	12	25
Scanning Frequency (Hz)	12	13
View Field (degrees)	360	360
Angular Resolution (degrees)	1.0	0.250
Precision	1% up to 3 m 2% : (3-5) m 2.5% : (5-12) m	1% up to 3 m 2% : (3-5) m 2.5% : (5-25) m

Table 2.1: LiDAR sensors characteristics [1].



(a) Footprint and dimensions of the chair with the sensors assembled. The circles represent the placement of the LiDAR sensors. Sensor RPLidar A2 in front with an angle of 193.5º and sensor RPLidar A3 in back with an angle of 0º.



(b) Footprint of the wheelchair overlaying the wheelchair itself.

Figure 2.9: Wheelchair footprint and LiDAR placement [1].

The two LiDAR sensors used are named **RPLidar A2** and **RPLidar A3**. In figure 2.9 the placement of the sensors is displayed in relation to the footprint of the wheelchair. Sensor RPLidar A3 has better characteristics, as can be seen in table 2.1 and is placed in the back and sensor RPLidar A2 is placed in front.

2.3.2 Odometry

The wheelchair used does not contain encoders on its wheels. As such, the LiDAR sensors are used to compute odometry data by calculating the difference between the

positions of two consecutive frames. Only the RPLidar A3 was used for odometry approximation since, as previously explained, it has more accuracy than RPLidar A2.

2.3.3 SLAM

For Simultaneous Localization and Mapping (SLAM) operations, once again the RPLidar A3 was used for its better characteristics. The Localization and Mapping (LaMa) algorithm developed in Universidade de Aveiro (Pedrosa *et al.* [13]), is used and by taking data from the sensors and odometry, it is capable of creating a map and localizing itself in it. The algorithm can also function in location-only mode when a previously created map is provided.

2.3.4 Costmaps

Two costmaps are used by the IW for navigation. One global costmap, that contains all that known map, is created from the map provided by the SLAM library. Another costmap is created using the data collected in real-time by the LiDAR sensors, this local costmap is smaller in size since it is limited by the range of the sensors and, unlike the global costmap, considers dynamic objects that were not presented in the map. The local costmap is used by the local planner to determine the viable path for the wheelchair to follow.

The determined resolution for both costmaps is 0.05 m/cel.

2.3.5 Global Planning

The function of the global planner is, as previously explained, to find a path between the current position and a user-defined goal. For this purpose, the IW uses the ROS *Global Planner* with the ROS implementation of pathfinding Dijkstra algorithm.

The parameters cost_factor, neutral_cost e lethal_cost allow to configure the behaviour of the planner and are set with the following values determined by experimental tests:

- cost_factor = 0.55
- neutral_cost = 50
- lethal_cost = 253

These parameters are set to indicate to the global planner the characteristics of the desired path, specifically, how far from obstacles it should pass and how much importance to give to the costmaps.

2.3.6 Local Planning

The local planner sets the actual path the robot takes, considering the path generated by the global planner, and the surrounding environment detected by the sensors.

Through experimental testing, the TEB planner was considered the best for indoor IW navigation. This planner has various parameters that can be changed to slightly alter the way the robot moves.

2.4 Semantic Navigation

Semantics is the branch of linguistics concerned with meaning. Semantic navigation in autonomous mobile robotics is a concept that describes the ability of a robot to have, and act upon, information about the meaning of elements in its environment in a human-like way.

It is common, in semantic navigation approaches, to integrate a lower-level navigator. This results in a layered architecture for mapping and planning. Assuming a priori knowledge of the map in the form of an occupancy grid, upper layers of mapping can be added representing topological and semantic knowledge.

The semantic planner in its turn, operates as an intermediary between the user and the low-level navigator, creating a semantic plan for the user-defined goal that is then divided into geometric goals passed down to the low-level navigator [14].

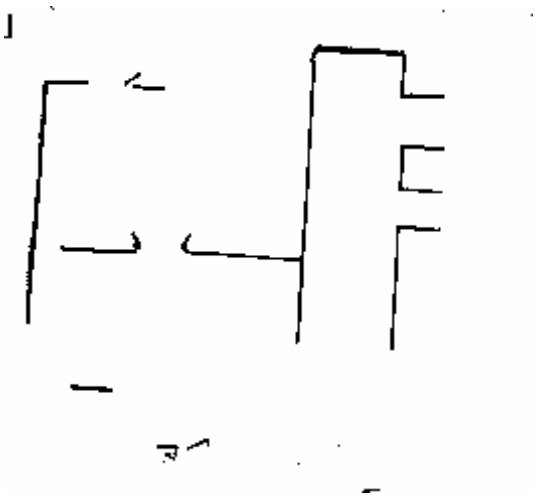
The addition of a semantic layer to the navigation of an autonomous mobile robot architecture makes the robot capable of higher-level cognition and decision-making and can adapt the navigation strategy, at a lower level, for different contexts, giving it better navigation capabilities.

2.4.1 Mapping

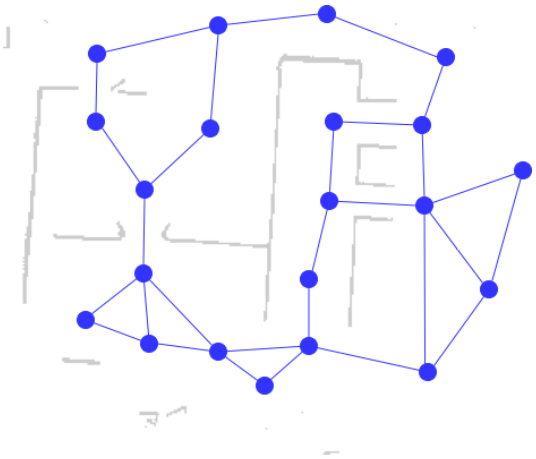
Conventional navigation in mobile robotics usually relies on extracting geometrical information from sensors and mapping this information in a way that can be used for geometrical path planning and execution.

When adding to a mobile robot the capability of understanding its environment, there is a need to represent this knowledge and connect it to the geometrical knowledge of the world. This usually results in a layered structure, as can be seen in fig. 2.10, with a metric map in its base, followed by a topological map and/or a semantics map in upper layers [15][16]. This layered map structure has inclusively been used in autonomous wheelchairs [17]

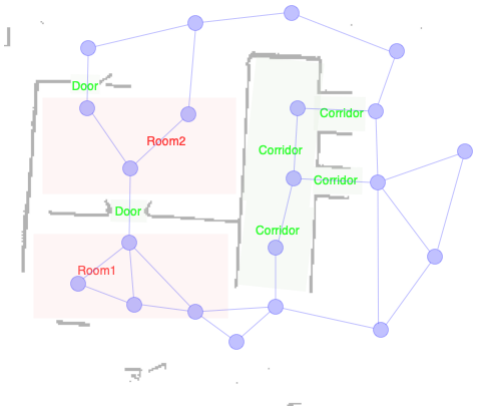
The semantic information can reside in different parts of this layered structure. In the topological layer, for example, the nodes and edges can be tagged with semantic information, like the type of space a node represents, or the cost of moving from one node to another can be represented in the edges.



(a) Metric map, as an occupancy grid



(b) Topological map over metric map which results in a topometric map.



(c) Semantic map

Figure 2.10: Mapping layers example based on an occupancy grid of the map assembled at IRIS-Lab

2.4.2 Planning

As previously said, path planning is an integral part of navigation in mobile robotics. Semantic planning considers the available semantic information of the world, as opposed to the geometry of the space in question. Semantic planning does not, however, substitute traditional geometric planning (global and local), but acts instead as an upper layer of planning that will command the geometric planners [18].

When there is an input of goal position in the map, the semantic planner uses the semantic maps available and a set of rules to decide the semantic plan to take. This semantic plan can be turned into a set of intermediate poses the robot must pass through being passed to the lower-level planners for geometric navigation.

Also, by dividing the plan into sections, the semantic planner can choose to use different path execution strategies for each section. The semantic planner can instruct

the lower-level planners with different configurations for each section, or even make use of different planners for each section. This allows the robot to be better equipped with the ability to navigate in different contexts and situations.

2.5 Semantic Navigation in Intelligent Wheelchairs

Semantic approaches to navigation in autonomous intelligent wheelchairs have been proposed in other works [19] [20] [17]. These approaches extract semantic information visually with RGB-D cameras or manually mark the semantic information on the map. These systems use this semantic information both to navigate differently in particular situations and to simplify the interaction between the user and the wheelchair.

There is also published work on intelligent wheelchairs where narrow passages were automatically detected and a controller was used for crossing them [21][22]. These systems were not fully autonomous though and were only assistive to the user in particular situations.

CHAPTER 3

Semantic Navigation for Intelligent Wheelchairs

Aiming to solve the problems found in previous work, the idea of adding a semantic layer to the indoor navigation framework was considered a worthwhile approach, not only for solving the identified issues, but also to build a more robust framework capable of making better decisions in each situation.

With robots ever more present in many application areas and interacting more with humans, it is important that robots are able to understand the meaning of the surrounding world. As such, semantic navigation and mapping are growing concepts in the field of robotics and allow robots to have not only geometric knowledge about the world but also semantic knowledge that can be used in the decision-making process [5].

In the specific context of this project, the robot having semantic information about narrow passages can improve navigation capabilities in several ways. Firstly, if the robot knows it has to cross a narrow passage, it can first align in a specific position and orientation so it can more easily pass through it.

Secondly, traditional planners try to get the shortest path to a goal, but there can be cases where the shortest path passes through a difficult narrow passage, while a slightly longer path is much easier and/or comfortable to follow. Having information about how difficult a passage is to cross, allows the robot to consider these cases.

Finally, knowing which sections of a path correspond to narrow passages, allows the robot to follow these sections in a different way, such as the use of a different local planner, or simply reducing the speed so it is more comfortable for the passenger [23].

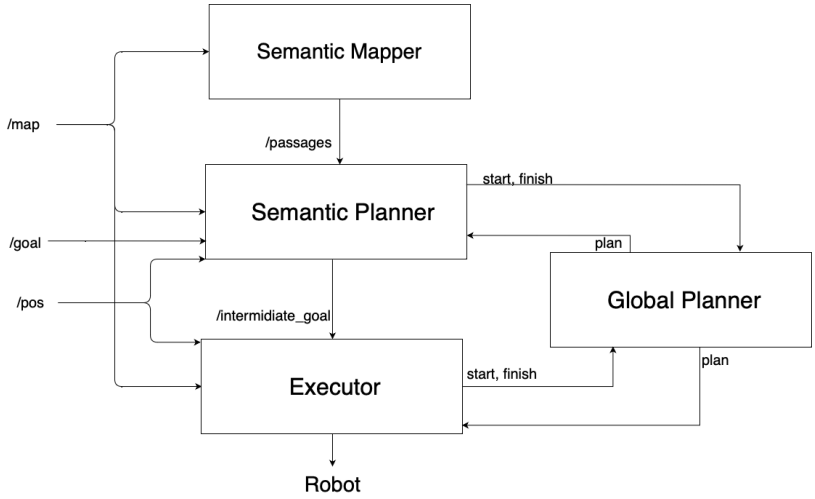


Figure 3.1: Conceptual Architecture proposed for semantic navigation. The architecture is composed of four modules, the two most relevant for the semantic navigation capabilities being the semantic mapper and semantic planner.

3.1 Proposed System Architecture

Two different aspects must be considered: acquiring semantic knowledge and acting upon the semantic knowledge. In the problem of the narrow passages, this can be mapped into two tasks:

- Mark the narrow passages on the known map.
- Consider the semantic information while navigating.

In figure 3.1 the main modules of the system are represented along whit the respective data flows. To accomplish the tasks previously listed, the following modules were developed:

- **Semantic Mapper** - Responsible for automatically marking the narrow passages on the map.
- **Semantic Planner** - Acts as an upper layer to the navigation stack and makes a semantic plan that consists of a set of intermediate goals ending in the user-defined goal.

The other two modules represented in figure 3.1 are the global planner and the executor. These modules are responsible for lower-level geometric navigation and are controlled by the semantic planner.

The marking of the narrow passages is done automatically by analysis and processing of the known map obtained by the mapping algorithm used in previous work and explained in section 2.3.3.

From this marking results a list of narrow passages. A **narrow passage** is defined as a driving section where all points of the Voronoi graph have an associated distance

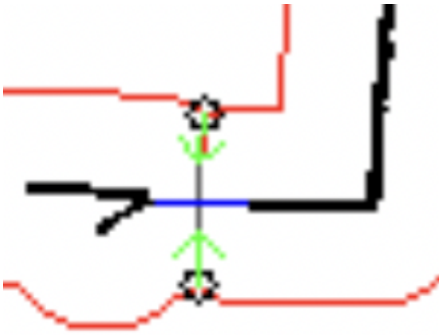


Figure 3.2: Narrow Passage with respective Boundaries and Approach Poses. The boundaries are in blue and are in this case coincident. The approach poses are in black with a green arrow indicating their respective direction.

to the closest obstacle points lower or equal to a given threshold but big enough for the wheelchair to traverse it. In the simple case, this section has only two **boundaries**, while in more general cases, three or more boundaries can exist. Each boundary is defined by a line, connecting the limit obstacle points of the passage, and an **approach pose** as can be seen in figure 3.2. The approach pose establishes the best pose to enter the narrow passage from that boundary. For this dissertation, only the simple case, of narrow passages with only two boundaries, is considered. While marking the narrow passage, a passage cost can also be calculated for a given cost function. This allows, in later stages, to plan paths considering the difficulty of certain passages avoiding them when possible and desirable.

In runtime, when the user sets a goal, the system checks if the path to the goal crosses any narrow passage. If the path does cross one or more passages, the path is then divided into sections and intermediate goals, corresponding to the passage(s) approach poses. The semantic planner can then order the executor to use specific strategies for each section of the path. This way, it is possible to pass through narrow passages by applying a different strategy that would make less sense for navigation in open space.

3.2 Semantic Plan Cost

Traditional navigation planners consider distance as the main factor for path cost. When designing a navigation system for an IW we should consider not only the distance to get to a goal but also the comfort of the passenger.

To address this problem, it was decided to consider a semantic plan cost, defined by a cost function that takes the distance cost (C_d) and the comfort cost (C_c), giving a total cost used for semantic planning. This function is represented in eq. (4.1) and can be implemented in several ways. The study of which cost function is ideal for this

purpose was not a subject of this dissertation. The distance cost is a function of the total distance of the path, and the comfort cost is a more subjective term that represents the difficulty or discomfort of certain path sections like, for example, a narrow passage. The semantic planner is responsible for calculating the semantic plan cost so it can decide which path to take.

$$C = f(C_{\mathrm{d}}, C_{\mathrm{c}}) \tag{3.1}$$

3.3 Semantic Mapper: Automatic Narrow Passage Marking

The first part of this solution is to find the narrow passages in the known map so that we can later navigate considering this information. As previously said, a narrow passage is a driving section between obstacle points, where the distance to the obstacle points associated with the Voronoi graph is inferior to a certain threshold but big enough for the wheelchair to traverse it. This interval of distances that defines what is and is not considered a narrow passage depends on several factors, mainly, the width and general shape of the IW.

In figure 3.3 the wheelchair used in this project is placed in the middle of a passage with a width of **100cm** as an example of a narrow passage. The steps for detecting and marking the narrow passages are the following and will be explained in the next sections:

- Calculate distance map
- Calculate the EVG
- Find sections of the EVG that may be a narrow passage
- Calculate the boundaries and approach poses of each narrow passage
- Calculate the associated cost for each passage

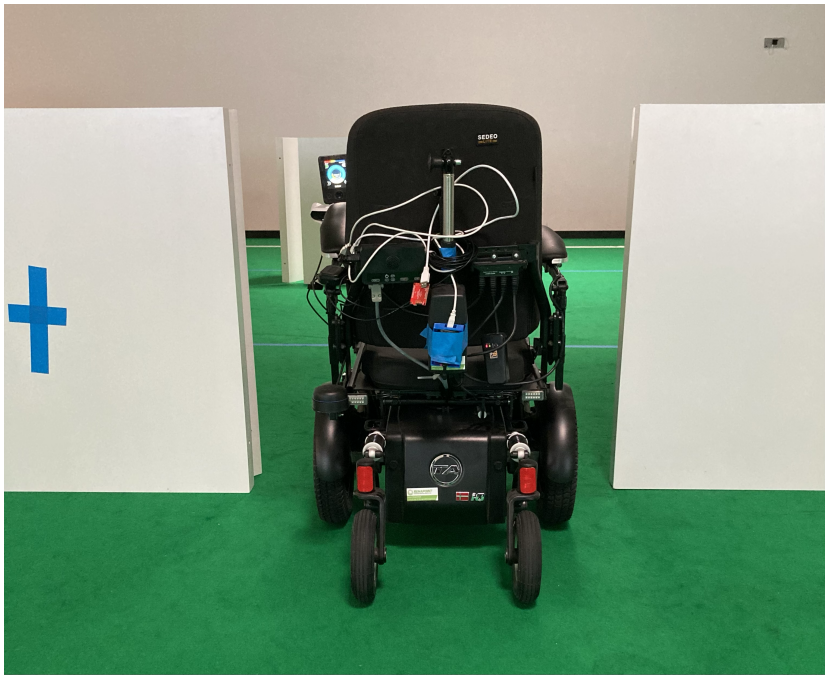


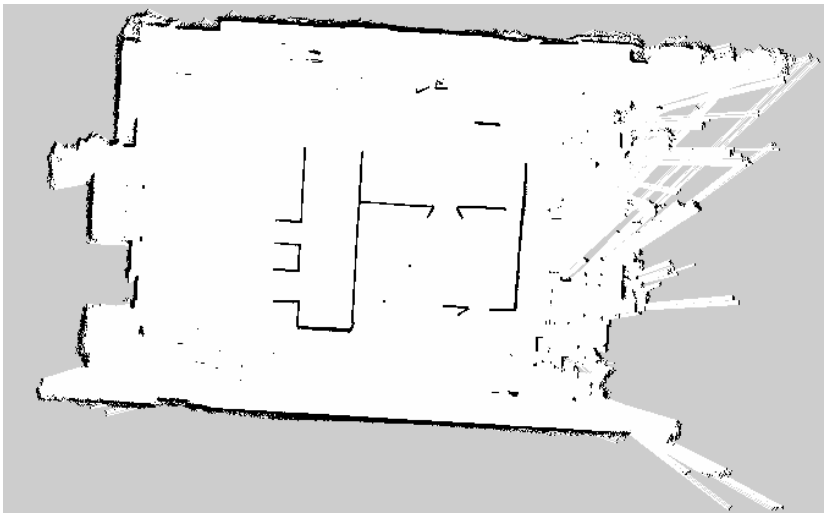
Figure 3.3: IW crossing narrow passage with 100cm of width at the IRIS-Lab map

3.3.1 Distance Map

The first step to finding narrow passages in the known map is to calculate the corresponding distance map. This will be useful as it will allow us to quickly find the distance to the nearest obstacle for any given pixel coordinates. The distance to the nearest obstacle is obviously important since the very definition of a narrow passage depends on the distance from obstacles.

The distance map is calculated using the python wrapper of the OpenCV[1] library called OpenCV-Python. OpenCV provides several methods for calculating distance maps including the method *distanceTransform()* which was the one used. Figure 3.4 corresponds to the distance map for the Iris lab map generated from an Occupancy Grid by the OpenCV library, represented in a greyscale image.

[1]https://opencv.org



(a) Ocupation grid created from the map at IRIS-Lab

(b) Distance map created form fig. 3.4a

Figure 3.4: Distance Map Example

3.3.2 Extended Voronoi Graph

As presented in section 2.1.1 the EVG can be a good tool for extracting a topological map from an occupancy grid [7]. Furthermore, the Voronoi graph, by definition, will include all the points exactly in the middle of all narrow passages, which will be useful for identifying them later.

The computation of the Voronoi graph was performed based on the work developed by Beeson, Jong and Kuipers on the EVG, which is available as open-source code[2]. This code was extended and adapted to be integrated with ROS and works as a **ROS node** in the system architecture. The occupancy grid is converted to a .pgm image and is fed to the EVG algorithm along with the following parameters:

[2]https://github.com/OpenSLAM-org/openslam_evg-thin

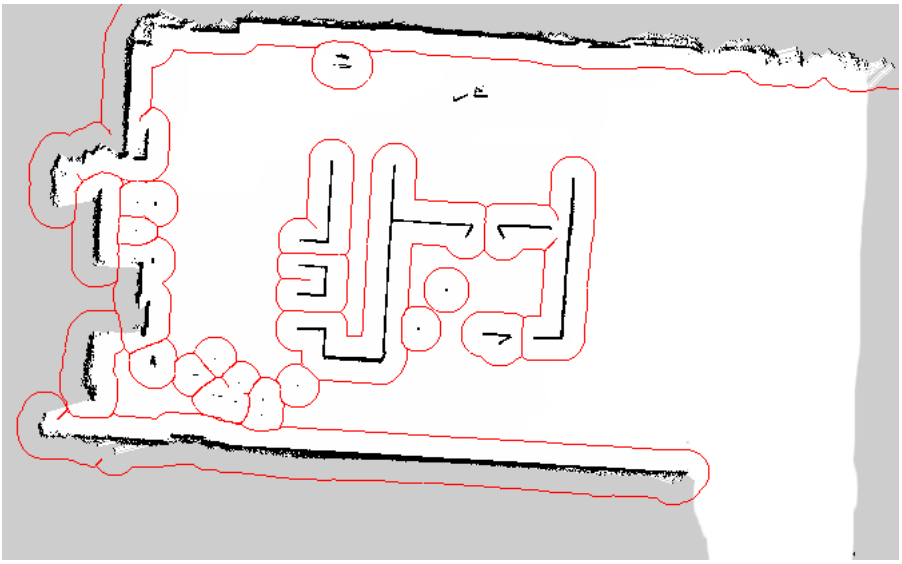


Figure 3.5: EVG generated with the OpenSLAM_evg_thin algorithm from an occupation grid of the map at IRIS-Lab. Some of the noise present in the original OG was cleared to make the EVG less irregular.

- Max distance: Maximum distance from obstacles where a point can be considered part of the Voronoi graph. This parameter corresponds to the M in equation 2.2.
- Min distance: Minimum distance from obstacles to consider a point part of the Voronoi graph, if two obstacles are closer together than double this distance, there will be no branch of the Voronoi graph in between these points. This parameter corresponds to the m in equation 2.2.
- Pruning: Boolean value that enables or disables the pruning function of the algorithm that removes superfluous branches of the Voronoi graph.

The parameters **max_distance** and **min_distance** are set respectively to half of the upper and lower ends of the interval that defines what is considered a narrow passage. The parameter pruning is set to **false** but this parameter does not affect the next steps for which the EVG is used. The result is a list of points that compose the EVG using image coordinates. Figure 3.5 shows the result of the EVG for the Iris lab map occupation grid with **min_distance = 35cm**, **max_distance = 80cm** and no **pruning**.

3.3.3 Passage Segmentation

With the EVG and the distance map, we can get the distance to the nearest obstacle for every point in the EVG. Taking the subset of points in the EVG that are at a distance contained in the narrow passage distance interval from the closest obstacle, we get all the points of the EVG that are in the middle of passages. The narrow passages can be derived from these points.

The algorithm for selecting these points from the EVG is represented in algorithm 1. This algorithm starts taking a list of points that make the EVG and the distance map for the map in use and selects all the points that are at an adequate distance from obstacles. Then this list of points is divided into its connected components, each corresponding to a narrow passage. Each group of connected points passes through the method *trimEdges* that removes the ends of each segment keeping only the centre part that corresponds to the actual passage. The result is a set of groups of points where each group corresponds to the points of the EVG that constitute the middle of a narrow passage.

In figure 3.6, sections of the EVG are marked in blue, corresponding to the groups of points that go through the middle of the narrow passages and that will be used in the next steps.

Algorithm 1 *NarrowPassageSegmentation*

```
Input: voronoiPoints, distanceMap
P = { }
foreach p ∈ voronoiPoints do
    if MIN_PASSAGE_WIDTH ≤ distanceMap[p]*2 ≤ MAX_PASSAGE_WIDTH then
        P = P ∪ {p}
    end
end
rawPassages = splitByAdjacency(P)
narrowPassages = { }
foreach passage ∈ rawPassages do
    narrowPassages = narrowPassages ∪ {trimEdges(passage)}
end
return narrowPassages
```

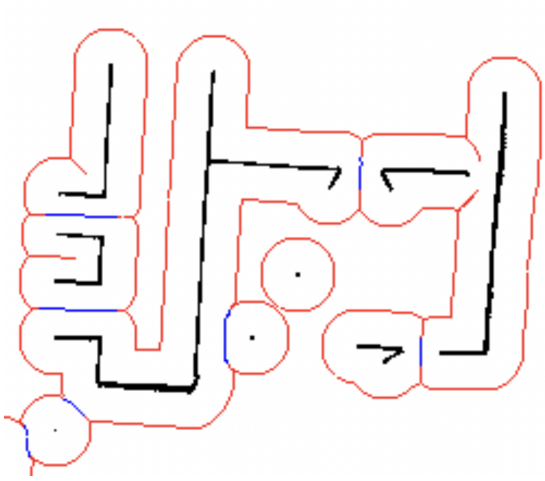


Figure 3.6: Segmentation of the EVG into Passages. The blue segments correspond to the points identified as narrow passages, and the points in red correspond to the remaining points of the EVG.

3.3.4 Passage Boundaries and Approach Poses

The previous step results in a set of groups of points, where each group of points corresponds to a narrow passage. Taking the points on opposite ends of each set and finding the closest obstacle point, gives us the entry of the passage. Lines can be traced between the two obstacle points on each side of the passage marking a **boundary**. Narrow corridors will have two boundaries, one on each side, and doors will only have one boundary, but for implementation simplicity, two coincident boundaries are considered in this case. In figure 3.7 we can see an example of a narrow passage along with its boundaries and approach poses.

On each side of a narrow passage, there is an interception of the EVG. This is one of the reasons the EVG was chosen as opposed to the GVG, as it always has an interception at a fixed distance on either side of the passages. These points, combined with an orientation vector pointing in the direction of the passage, compose the **Approach Poses**. For each passage, getting the closest EVG interception on either side gives the passage two approach pose.

In algorithm 2 we can see how this process is implemented. First, the algorithm gets a list of points called *voronoiSplits* that correspond to the interceptions of the EVG. Then, for each list of points that form a passage, from the previous step, it gets the two most apart points.

For each of these end-points, it performs the following:

- Get the closest obstacle point to the end-point.
- Get the reflection of this object over the end-point.
- Get the closest obstacle to the reflection point.

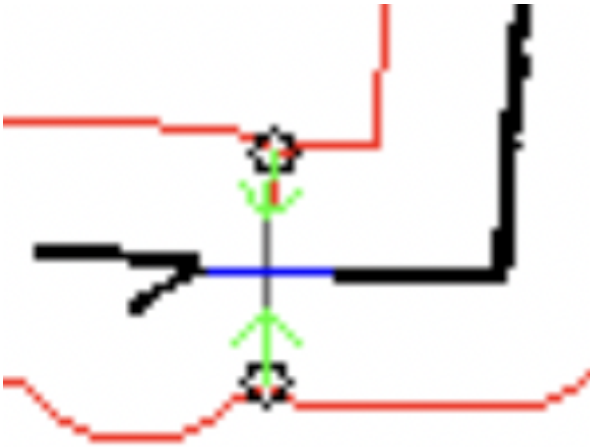


Figure 3.7: Narrow Passage with Boundaries and Approach Poses. The boundaries are in blue and the approach poses are black circles with green arrows pointing in their respective direction.

This results in two obstacle points that are on opposite sides at the edge of the passage. A line drawn between these two obstacle points forms a **boundary** of the passage. This is performed for both end-points, resulting in two boundaries.

By getting the closest point to each end-point in the EVG interceptions we get the position of the respective **approach pose**. Then the direction of the approach poses is calculated so it points to the narrow passage.

The boundaries and approach poses are stored in each passage data structure.

Algorithm 2 *FindBoundariesAndPoses*

Input: narrowPassages, voronoiPoints, distanceMap
voronoiSplits = getInterceptions(voronoiPoints)
foreach passage ∈ narrowPassages **do**
  end_points = getMostApartPoints(passage)
  foreach p ∈ end_points **do**
    obs = getClosestObs(p, distanceMap)
    obsMirror = reflectionOverPoint(obs, p)
    obsMirror = getClosestObs(obsMirror, distanceMap)
    boundary = { obs, obsMirror }
    passage.boundaries = passage.boundaries ∪ {*boundary*}
    appr_p = getClosestInList(p, voronoiSplits)
    direction = $\text{atan}_2(p.y$ - appr_p.y, $p.x$ - appr_p.x)
    approach_pose = { appr_p, direction }
    passage.approach_poses = passage.approach_poses ∪ {*approach_pose*}
  end
end

3.3.5 Narrow Passage Comfort Cost

The semantic mapper is responsible for assigning each narrow passage a comfort cost. It is difficult to measure comfort in an objective way, but some published research indicates that the main factors at play when it comes to comfort in autonomous wheelchairs are the distance from obstacles and the velocity of the wheelchair [24], [23]. Sudden changes in direction are also a factor that contributes to discomfort.

Several strategies can be defined for calculating an appropriate cost function for the comfort of each narrow passage considering the previously described metrics. For this dissertation, since it was not possible to perform adequate passage comfort experiments, the comfort cost (C_c) for narrow passages is defined as a function of the passage geometry as represented in equation 3.2. This function can later be implemented upon further study.

$$C_c = f(narrowPassage) \tag{3.2}$$

3.3.6 Narrow Passage Representation

At the end of the previous processes, the result is a set of identified narrow passages, each with two boundaries and two approach poses. The next challenge is then to represent this information in a way that can then be used by the semantic planner. For this purpose, a structure like the one represented in figure 3.8 is used. An array of narrow passages in this format is then available and used by the semantic planner.

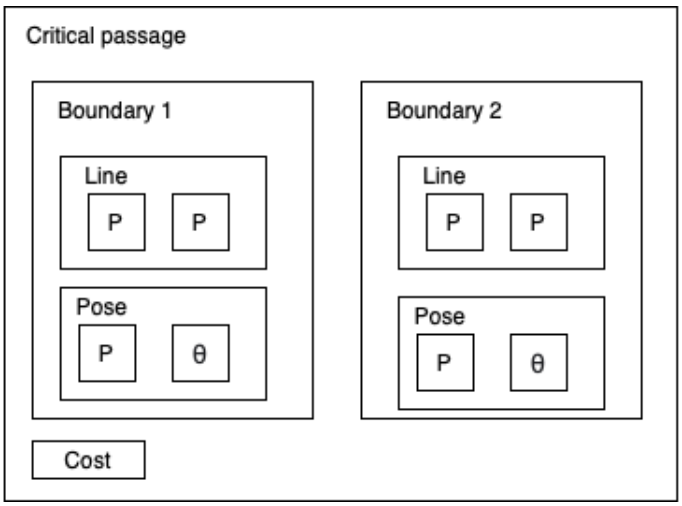


Figure 3.8: Critical Passage Message Structure

3.4 Semantic Planning

The semantic planner acts as an upper layer of planning above the global and local planners. The only semantic function it plays is related to the narrow passages, and the goal is to pass through the passage without colliding, obviously, and with the most comfort possible, minimizing sudden movements and maximizing the distance from obstacles. For this purpose, the strategy for traversing narrow passages is:

- Place the wheelchair at the **approach pose** with the correct orientation facing the passage.
- Adopt a specific navigation strategy for narrow passages (lowering the speed for example).
- Set the goal to the **exit pose** corresponding to the approach pose on the other side of the passage with its direction flipped.

The first step when the user inputs a goal position is to request a plan for the goal from the global planner and to detect if the plan crosses any narrow passage. If it does not, the semantic planner simply forwards the goal to the executor, if it does, then the plan has to be analysed by the semantic planner. This analysis checks alternative plans and, if the final plan still passes through one or more narrow passages, it has to be divided into sections. For example, if the path from the current position P to the set goal G passes through passage α, the path from P to G would be divided in 3 sections, from P to $\alpha_{approach\_pose}$, from $\alpha_{approach\_pose}$ to α_{exit_pose}, and from α_{exit_pose} to G.

These sections can then be tackled in different ways during path execution where the intermediate goals defined in the previous step are passed to the executor. All these steps can be visualized in figure 3.9 which represents the logic flow of the semantic planner.

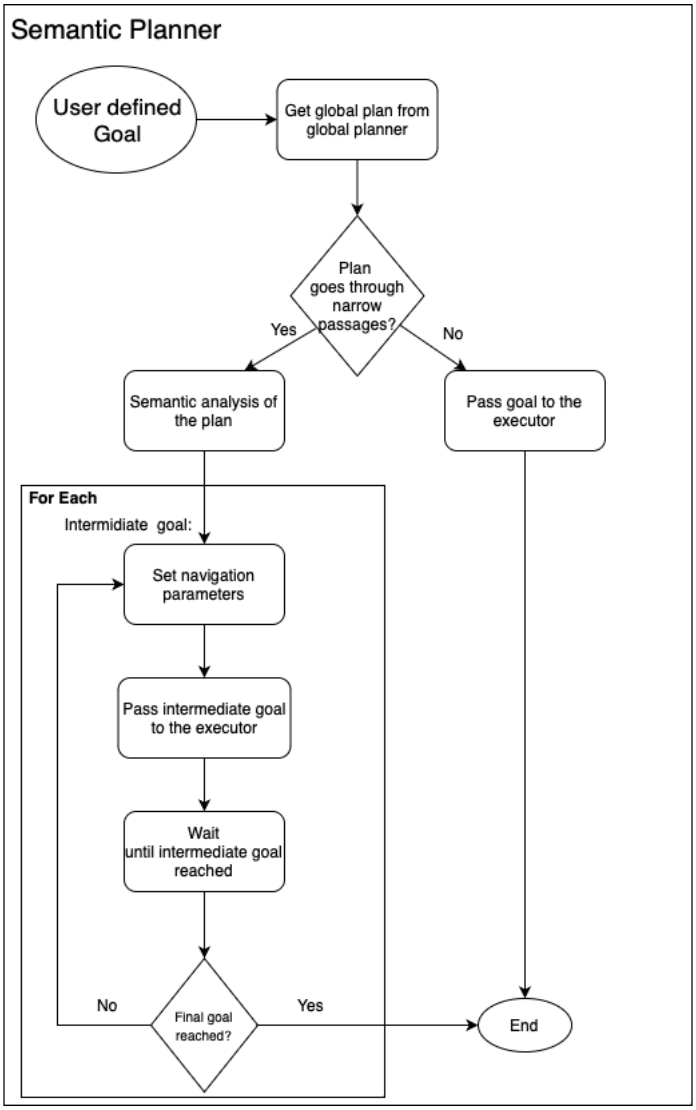


Figure 3.9: Semantic planner simplified flowchart. We can see the steps of checking narrow passage crossing, plan analysis and plan execution.

3.4.1 Passage Map

The semantic mapper produces a list of narrow passages that contain the necessary information to know where on the map each passage is located. But in this raw information, the boundaries are each represented by two points which, on its own, is not very useful for detecting if a list of points, that constitutes a path, crosses any boundary and therefore, any narrow passage.

When the semantic planner receives the list of narrow passages, it creates a passage map that acts as a mask for the current navigation map in the form of an OG. This mask consists of a 2D array image, with the same dimensions as the current map's occupation grid, where the boundaries for each passage are drawn with different values for each boundary. These values are mapped to the indexes of an array that contains the

corresponding approach pose for each boundary. In the case of short narrow passages (doors), where the two boundaries are coincident, the value in the map is multiplied by -1 so it can be identified later. With this map, it is easy to go through a path, as a sequence of points, and identify if any boundary is crossed as well as get the corresponding approach pose.

3.4.2 Passage Crossing Detection

The Global Planner receives two poses, start and finish, and returns a plan, a set of adjacent poses, from the first to the latter. When the user inputs a goal pose, the semantic planner calls this service and gets a plan from the robot's current position to the user-defined goal.

The semantic planner then checks if this plan crosses any of the narrow passage boundaries defined by the semantic mapper. It does this simply by going through all the poses in the plan and checking if any of them coincides or is adjacent to a boundary. By doing this, the semantic planner finds all the passages crossed by the path and keeps that information for the next step.

Algotithm 3 explains this process. For each point in the plan, it gets the neighbour points and verifies if any neighbour or the point itself coincides with a boundary in the passage map. If this happens, it stops going through the points in the plan, saves this point for the next step, and removes the already verifies section of the plan. The algorithm repeats this process until it reaches the end of the plan without detecting any more boundaries.

Algorithm 3 *DividePlan*

```
Input: plan, currentPose, passageMap
P = { }
detected = false
while not detected do
    detected = false
    i = 0
    forall p ∈ plan do
        neighbors = getNeighbors(p) ∪ {p}
        forall n ∈ neighbors do
            if passageMap[n] != 0 then
                P = P ∪ {n}
                detected = true
                break
            end
        end
        if detected then
            plan.drop(i+k) /* Dismiss some points in the plan to avoid double
               detection of a boundary */
            break
        end
        i++
    end
end
return P
```

3.4.3 Path Splitting

From the previous step, the planner knows the passages that are crossed by the plan and must be tackled. If no passage was detected, this step is skipped since the plan needs not to be divided. At the end of this step, the result is a list of intermediate goals, the last one being the goal defined by the user.

In fact, this step actually occurs simultaneously with the passage detection. During passage detection, whenever a **boundary** is found to be crossed the corresponding **approach pose** is found and added to the list of intermediate goals. All boundaries have a corresponding approach pose, but the orientation of these poses depends on it being an entry pose or an exit pose which depends on the trajectory of the robot. By default, the orientation of each pose points to the corresponding passage, therefore, if the approach pose is to be an exit pose, this orientation must be flipped.

Thankfully, when traversing a path, entry boundaries and exit boundaries always come interleaved so it is easy to keep track of which one it is by keeping track of the robot's semantic location. Starting with semantic location **in_free_space**, if the robot crosses a passage boundary, it enters the passage becoming **in_passage** if the robot then crosses another boundary, it comes back to **in_free_space**. This way, the

algorithm keeps track of which **approach poses** must have their orientation flipped. On a side note, if the passage is a door, this becomes less important since when a boundary is crossed, both boundaries are being crossed as they are coincident, in this case when the boundary crossing is detected both entry and exit positions are added to the list of intermediary goals straight away.

When the end of the path is reached, the result is a list of poses corresponding to the **approach poses** some with the orientation flipped that will be the intermediate goals passed to the executor and the last goal is the user-defined goal without change. Each goal has also an associated type. This type can be one of the following:

- Free Space
- Door Entry
- Corridor Entry
- Door Exit
- Corridor Exit

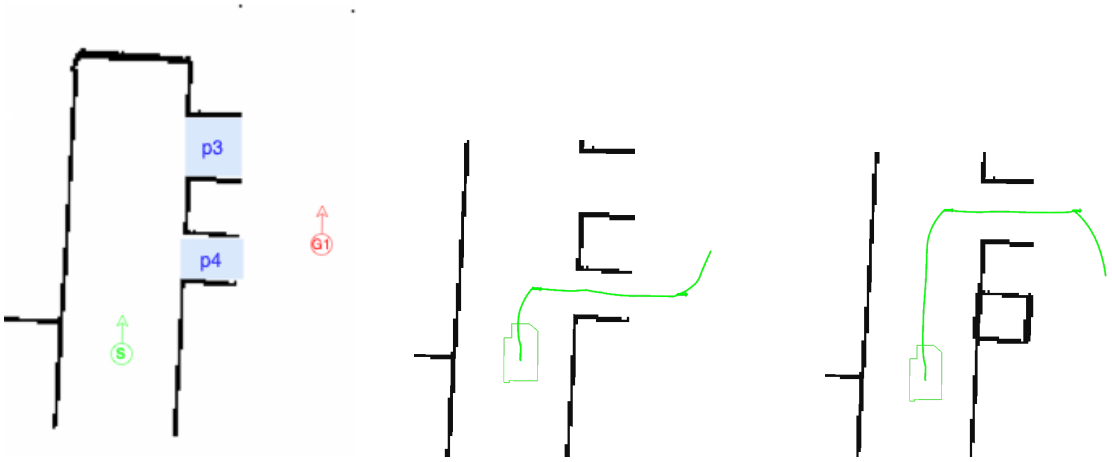
These types are important because the type of goal will define the specific navigation parameters used by the path execution step.

3.4.4 Semantic Plan Cost

As previously said, the semantic planner is responsible for calculating the total cost of a plan considering the distance cost and the comfort cost. This can be done after the path splitting step using a given cost function. The semantic planner takes the list of intermediate goals, adds the length of the path between each one and the result is the distance cost. It then takes the added **comfort costs** of the crossed passages and inputs both this cost and the **distance cost** to a cost function resulting in the **total cost**.

Then the problem becomes how to find other possible paths and evaluate if they have a lower total cost. The distance cost can not be reduced, since the initial path came from the global planner and this planner minimizes the distance when calculating a path. However, the comfort cost could be lower if the path crossed different narrow passages or none at all, at the cost of a greater total distance.

To evaluate this possibility, after acquiring a divided plan, the semantic planner proceeds to check the viability of alternate paths. It does this by taking each narrow passage crossed by this first path and closing them on the map. It closes them by drawing lines in each passage respective **boundaries**. After closing these passages the planner goes back to the step of Passage Crossing Detection described in section 3.4.2 and repeats all the subsequent steps until it gets a new divided plan. The total cost for this new plan is then calculated and compared to the total cost of the previous path.



(a) (b) (c)

Figure 3.10: Path cost evaluation process. Visual representation of the algorithm that considers the cost of alternate paths by blocking narrow passages.Considering the example scenario in figure 3.10a where the start position is in green and the goal position in red, firstly the semantic planner obtains the plan from the global planner as in figure 3.10b, verifies that it passes through passage p4 and calculates the total cost of the plan according to a cost function. Secondly, the semantic planner blocks out passage p4 on the map and requests a new plan from the global planner as in figure 3.10c, forcing this plan to pass through the other passage. The semantic planner then calculates the total cost for this new plan, compares it to the previous and chooses the final plan accordingly.

The path with the lowest cost is chosen. This process can be visualized in figure 3.10. This implementation has some limitations that are discussed in the next chapters.

3.4.5 Path Execution

Taking the list of poses from the previous step, this step feeds the intermediate goals to the executor until the IW gets to the goal.

Since the plan was divided into different sections (if any narrow passage was crossed by the global plan), it's possible to use a different navigation strategy for each one by changing the executor configurations before sending each intermediate goal. In this dissertation, only the narrow passages problem is considered, but these different sections can represent different kinds of challenges like the ones presented in chapter 1, each with its own navigation strategy.

CHAPTER 4

Experiments and Results

To validate the proposed solution and determine possible changes in configuration, several simulations and real-world tests were performed to evaluate the performance of the solution presented in chapter 3.

As explained in the previous chapter, the semantic part of the system has two parts: **Automatic Narrow Passage Detection** and **Semantic Planning**. These two parts were tested independently and the respective experiments are presented separately in this chapter.

4.1 Assessing the Semantic Mapper

In order to assess the semantic mapper (see section 3.3), and on its behalf algorithms 1 and 2, the OGs depicted in figures 3.4a and 2.1 were used. Taking into consideration the physical dimensions of the wheelchair available, the width interval used to consider passage as narrow was [88 cm, 150 cm].

In figures 4.1 and 4.2 we can see the results obtained, drawn over the corresponding OG images. Form the visual verification of these results, we can see that the algorithm correctly identifies the narrow passages as defined in the previous chapter. This does not always translate into the correct marking of what constitutes a narrow passage in the real world, since the occupation grids used contain artefacts and objects that are indistinguishable from the walls, form the LiDAR point of view.

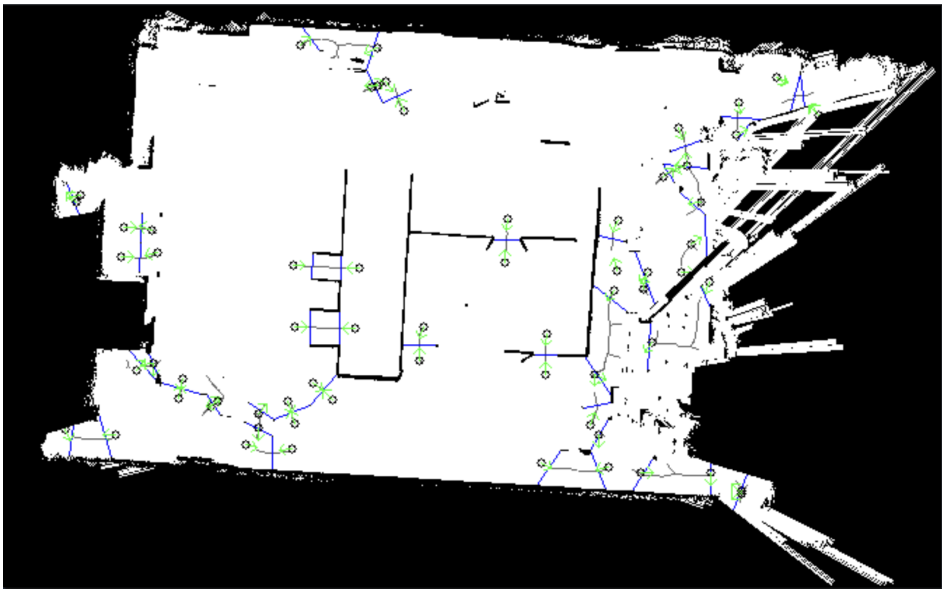


Figure 4.1: Narrow passage marking of the occupation grid in figure 3.4a. For each narrow passage, the boundaries are in blue and the approach poses are in black with a green arrow pointing in the direction of entry for the respective passage.

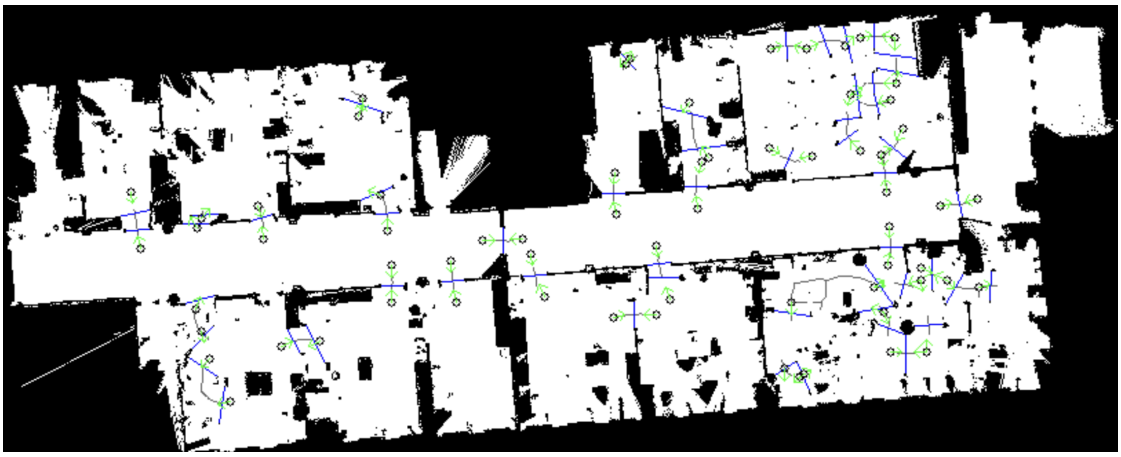


Figure 4.2: Results from Narrow Passage marking for the map of a building at the University of Freiburg.For each passage, the boundaries are in blue and the approach poses are in black with a green arrow pointing in the direction of entry for the respective passage.

4.2 Assessing the Semantic Planning

In this section, the methodology for the tests performed to evaluate the effectiveness of the semantic planning module of the solution proposed in this dissertation is presented. In figure 4.3 a photograph of the experiments scenario is displayed, this scenario was assembled at IRIS-Lab and was used for both real-world and simulation tests. An OG, created form this scenario and depicted in figure 3.4a was used for the tests. The OG was occasionally modified for performing specific tests considering also the space constraints in the lab.

Figure 4.3: Photograph of the experiment scenario assemble at IRIS-Lab.

It is important for each test to repeat each iteration several times to give a greater confidence level in the results. For this, it must be assured that the starting conditions are always, as close as possible, the same for each run, including the starting position of the IW. For the simulation, tests it is easy to start the IW always in the same position, as that can be adjusted in the simulator configuration. For the real-world tests, it was necessary to assure the starting position was as close as possible to the same for all runs, for this purpose, markers were placed on the floor in the desired position as showed in figure 4.4, and the IW was carefully placed in the begin of each run to align with the markers as close as possible. Additionally, the system requires the user input for a location estimate on startup, to remove the possible variation of manually inputting this estimation, this process was automated, giving the system the location estimate at the beginning of each run for the specific starting position.

Three tests were performed for validating the semantic planner with different objectives:

- Validity of the semantic planner functionality: The goal of this test is to verify if the semantic planner correctly divides the global plan into sections according to the narrow passages it crosses. The results were also compared with the results without the use of semantics to verify the effectiveness of the aproach.
- Evaluation of parameters for narrow passage crossing: These tests use different configurations of the local planner to find out if this can improve narrow passage crossing performance.
- Influence of Comfort Cost in Semantic Planning: Validate the comfort cost functionality of the semantic planner by applying a cost function and verifying if the planner chooses the path accordingly.

For those purposes, the following metrics were considered for the results:

- Success in completing the task

Figure 4.4: IW in the starting position for tests. The blue tape on the floor marks the position of each wheel for controlled repeatability.

- Time used to complete the task
- Comfort of the trajectory taken

The comfort metric is particularly difficult to evaluate and would require experiments performed with a group of subjects to reliably measure the comfort level. Nonetheless, as previously said, the main factors that play into passenger comfort levels in IW are the distance from walls and obstacles, and the velocity [24], [23]. Taking this into consideration, when crossing a narrow passage, a trajectory that maximizes distance from the walls, and reduces velocity when traversing the passage, can be considered more comfortable.

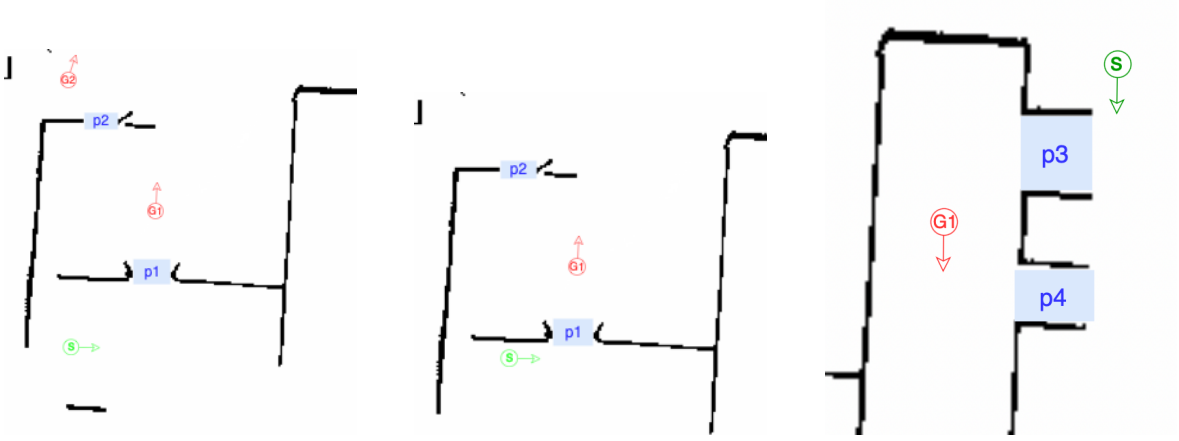
For both the real-world and simulation tests, 5 runs for each scenario were performed with and without the use of the semantic planner. An occupant was seated in the IW both for the realism of the tests, as the weight can impact the trajectory execution, as well as for safety so the IW could be quickly turned off in case of eminent collision. The presence of a person seated in the IW can also give a first-person idea of the comfort and general feel of the trajectory, although this metric can not be trusted reliably.

Finally, the purpose of these tests is to evaluate the performance of the semantic planner only, therefore, it is assumed that the semantic information available, specifically, the list of narrow passages and respective information, is correct.

4.3 Validity of the semantic planner functionality

As described in section 3.4 the function of the semantic planner is to divide the path from the current position to the user-defined goal and forward the intermediate goals to the lower-level planners. The goal of these tests is to validate if the semantic planner correctly identifies narrow passages in the path, correctly divides the path into sections passing through the way-poses and can execute each section. Control runs, without the use of the semantic planner, were also performed so the performance can be compared.

Three scenarios were considered, as can be seen in figure 4.5a, in the first scenario, the path transverses two consecutive door-like narrow passages. In the second scenario (fig 4.5b), the purpose was to force the wheelchair to start close to a wall, being the narrow passage in that wall. Finally, the third scenario (fig 4.5c), forces the wheelchair to transverse a corridor-like narrow passage.

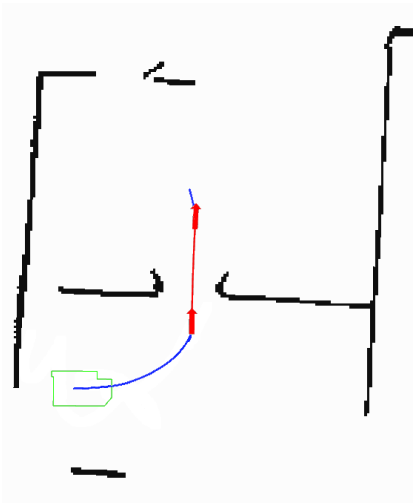


(a) **(b)** **(c)**

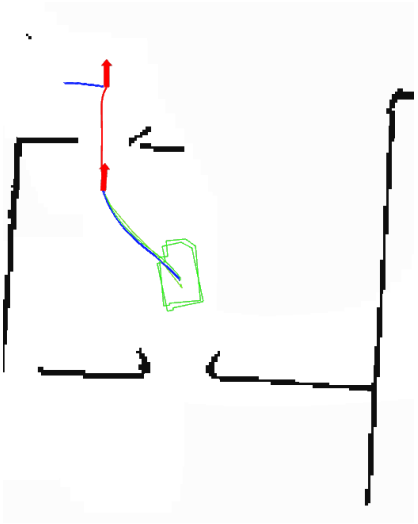
Figure 4.5: Scenarios used for semantic planner functionality tests. In fig 4.5a, the path goes from S to G1 through narrow passage P1 and then from G1 to G2 through narrow passage P2. In fig 4.5b, the path goes from S to G1 through narrow passage P1. Finally, in fig 4.5c, the path goes from S to G1 through narrow passage P2.

Semantic Plan Division Results

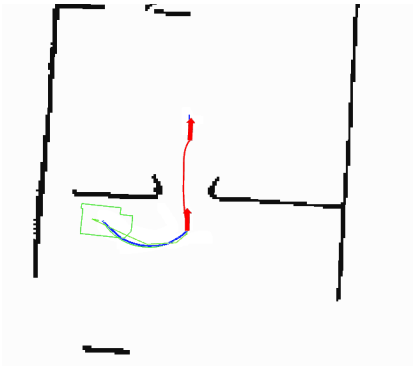
In figure 4.6 we can see the plan division performed by the semantic planner for each scenario. The sections in blue correspond to free space sections, the sections in red correspond to the narrow passages and the red arrows are the intermediate goals corresponding to the passages' approach or exit poses. As intended, the semantic planner divides the plan into sections according to the crossed narrow passages for each goal, using the passages' way-poses as intermediate goals separating plan sections.



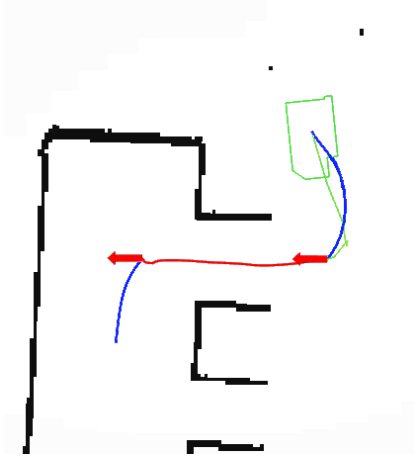
(a) Path division for goal 1 of the consecutive passages scenario.



(b) Path division for goal 2 of the consecutive passages scenario.



(c) Path division for the parallel aproach scenario.



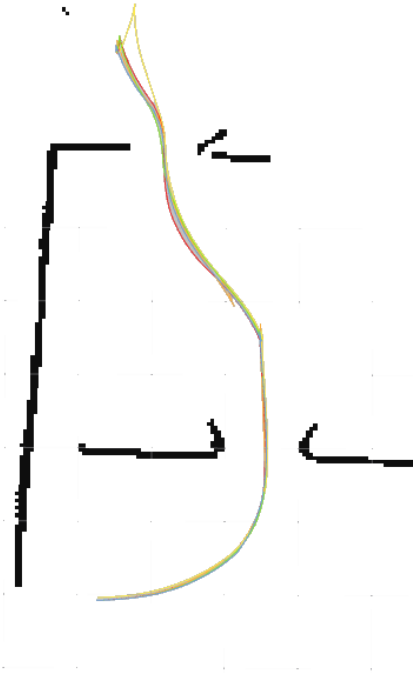
(d) Path division for the narrow corridor scenario.

Figure 4.6: Results of plan division performed by the Semantic Planner. The plan is divided into sections delimited by the narrow passages' way-poses in red. The sections of the plan are coloured in blue for free space sections and in red for narrow passage sections. In green, we can see the wheelchair footprint and the plans of the lower-level planners.

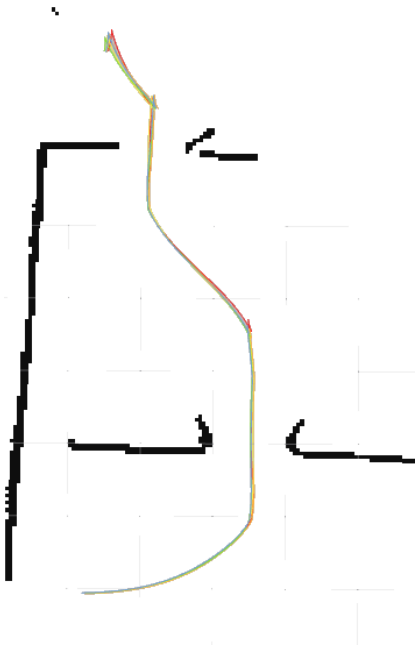
4.3.1 Consecutive Passages Scenario

The results for the consecutive passages scenario can be seen in figure 4.7 and table 4.1. From these results, we can see that the semantic planner correctly divides the path into sections going through the way poses. In figures 4.7b and 4.7d there is a clear difference in the paths compared to the paths without using the semantic planner. The paths created by the semantic planner are longer but result in a perpendicular crossing of the passages, as intended.

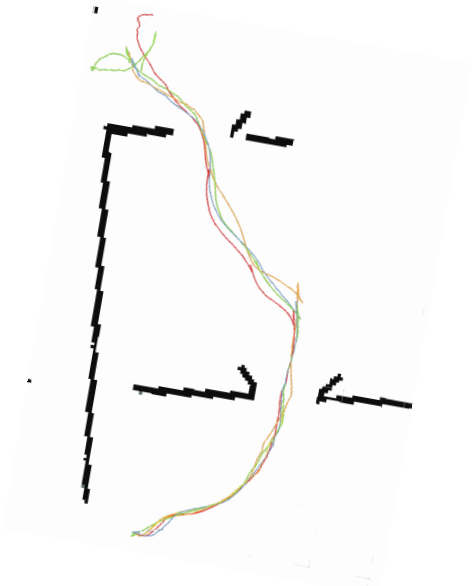
Looking at the results in table 4.1 we can see that for simulation, both the strategies,



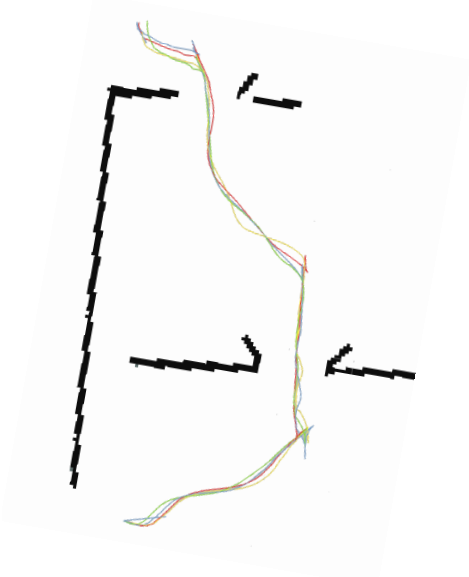
(a) Paths taken by the wheelchair for 10 iterations in simulation without the semantic planner.



(b) Paths taken by the wheelchair for 10 iterations in simulation with the semantic planner.



(c) Paths taken by the wheelchair for 5 iterations in the real world without the semantic planner.



(d) Paths taken by the wheelchair for 5 iterations in the real world with the semantic planner.

Figure 4.7: Paths taken by the wheelchair for the consecutive passages scenario. Paths for both simulation and real-world experiments with and without the semantic planner.

Configuration	N Runs	N Success	Avg S - G1	Avg G1 - G2	Max G1 - G2
Sim W/O Semantic	10	10	7.54s	33.08s	68.10s
Sim W/ Semantic	10	10	9.06s	16.56s	25.70 s
R-W W/O Semantic	5	3	27.22s	32.68s	50.71s
R-W W/ Semantic	5	4	34.49s	39.09s	46.76s

Table 4.1: Results for the consecutive passages scenario of tests for validation of the Semantic Planner. Comparison of results with and without the use of the Semantic Planner both in simulation and real-world

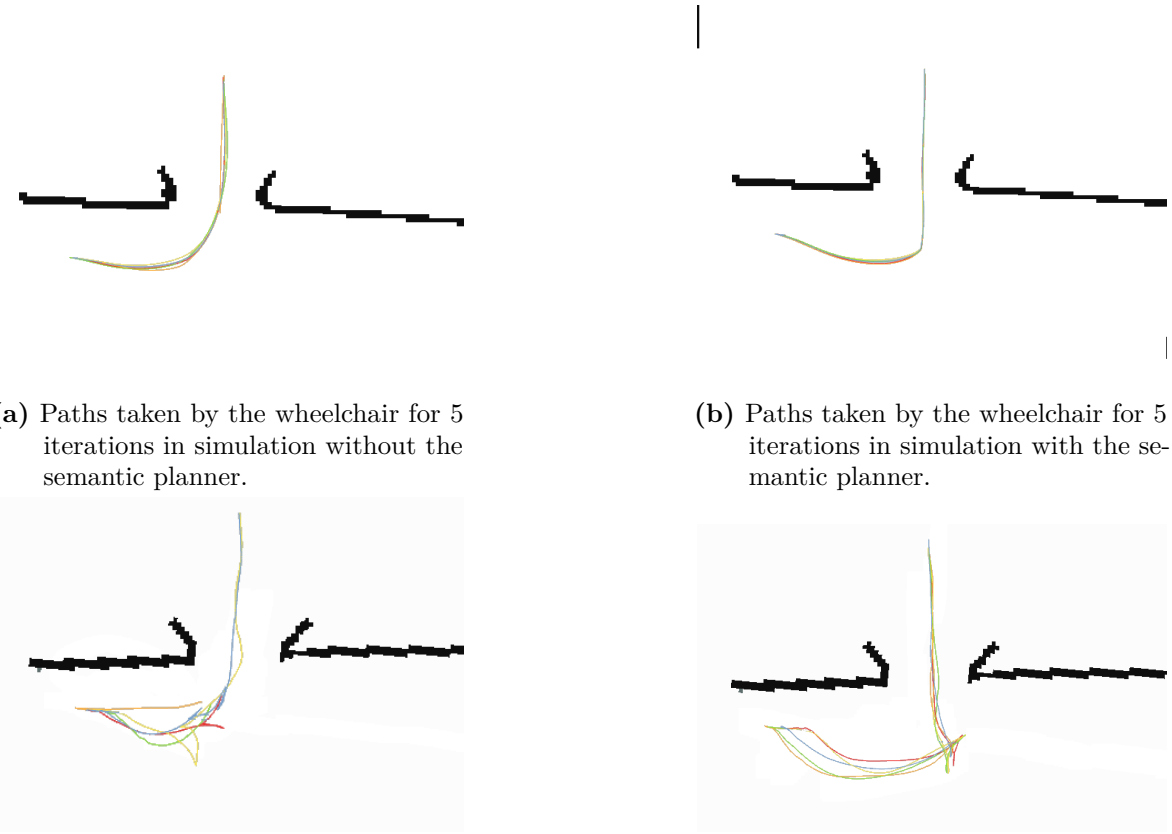
with and without the use of the semantic planner, have a 100% success rate crossing both passages. The same can not be said for the real-world tests, where the strategy without semantic failed two of the five runs performed compared to only one failed run from the semantic strategy. The most relevant result though comes from the times measured. Crossing the first narrow passage, we can see that the use of the semantic planner actually results in a slower crossing, this is to be expected since the resulting path is longer. For the second passage though, since it is narrower and more difficult to cross, we can see that the semantic approach, in spite of the longer path, results in a faster crossing since the wheelchair is better positioned to cross the passage and can do it more easily.

4.3.2 Parallel Approach Scenario

The results for the parallel approach scenario can be seen in figure 4.8 and table 4.2. In these results, there is a clear difference between the simulation and real-world results. In the simulation, even without the use of the semantic planner, the wheelchair can successfully cross the passage. In the real world though this is not the case and, without the semantic planner, the wheelchair fails to cross the passage most of the time. In figure 4.8c we can see that even when the wheelchair is able to cross the passage, the path it takes is very irregular and this combined with the times measured in table 4.2 shows that the wheelchair cannot reliably cross this passage with this solution. These results are exactly as expected since this situation has been identified in previous work as a particular problem.

Configuration	N Runs	N Success	Avg S - G1	Max S - S1
Sim W/O Semantic	10	10	8.84s	9.70s
Sim W/ Semantic	10	10	11.74s	12.90s
R-W W/O Semantic	5	2	59.04s	75.33s
R-W W/ Semantic	5	5	34.13	37.94

Table 4.2: Results for the parallel approach scenario of tests for validation of the Semantic Planner. Comparison of results with and without the use of the Semantic Planner both in simulation and real-world



(a) Paths taken by the wheelchair for 5 iterations in simulation without the semantic planner.

(b) Paths taken by the wheelchair for 5 iterations in simulation with the semantic planner.

(c) Paths taken by the wheelchair for 5 iterations in the real world without the semantic planner.

(d) Paths taken by the wheelchair for 5 iterations in the real world with the semantic planner.

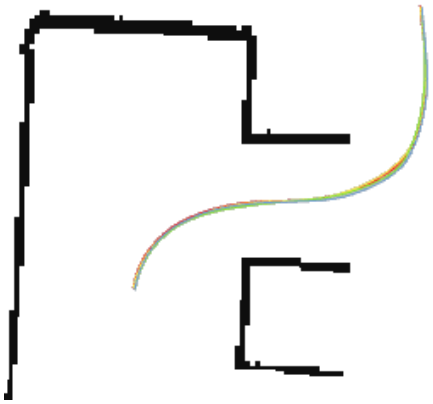
Figure 4.8: Paths taken by the wheelchair for the parallel approach scenario. Paths for both simulation and real-world experiments with and without the semantic planner.

Using the semantic planner though, there is a clear difference. The path aligns with the entrance of the passage and the wheelchair could successfully cross the passage every time in the performed tests. In the simulation results, we can see that the semantic aproach did take more time than the other, but this doesn't translate into the real world, where without the use of the semantic planner, the wheelchair has a real difficulty crossing the passage.

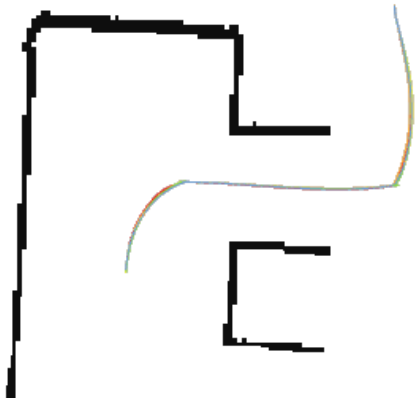
4.3.3 Narrow Corridor Scenario

The results for the narrow corridor scenario can be seen in figure 4.9 and table 4.3. Both approaches yielded similar results for this scenario. The semantic approach did increase the time to cross the corridor both in simulation and in the real world, as can be seen in table 4.3, but did not compromise the plan execution.

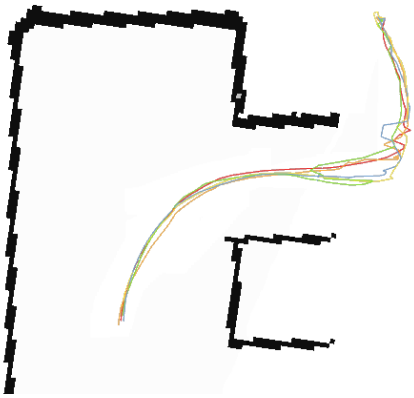
As for the path taken by each approach, in figure 4.9c. that shows the paths in the real world without the use of the semantic planner, it is noticeable the movement of the wheelchair is a bit irregular entering the passage although not as much as in



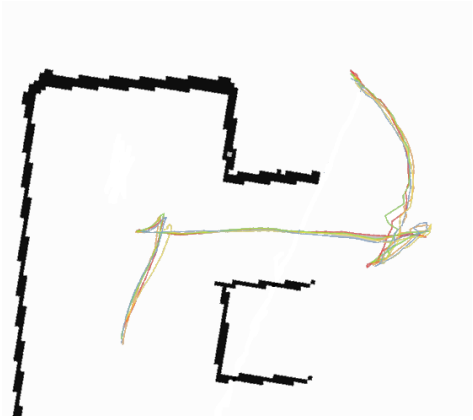
(a) Paths taken by the wheelchair for 5 iterations in simulation without the semantic planner.



(b) Paths taken by the wheelchair for 5 iterations in simulation with the semantic planner.



(c) Paths taken by the wheelchair for 5 iterations in the real world without the semantic planner.



(d) Paths taken by the wheelchair for 5 iterations in the real world with the semantic planner.

Figure 4.9: Paths taken by the wheelchair for the narrow corridor scenario. Paths for both simulation and real-world experiments with and without the semantic planner.

the previous scenario. With the use of the semantic planner, we can see that the chair executes a backwards motion to align with the passage, this motion can be a bit awkward as it might not be necessary if the wheelchair performed a wider arch approaching the passage, but it is a more consistent approach when compared with the results without the use of semantic since every iteration produces a similar movement without much unpredictability. Furthermore, the semantic approach resulted in a path that kept the wheelchair further from the walls that, as previously discussed, might be more comfortable for the passenger.

Configuration	N Runs	N Success	Avg S - G1	Max S - S1
Sim W/O Semantic	10	10	9.64s	9.70s
Sim W/ Semantic	10	10	14.87s	15.00s
R-W W/O Semantic	5	5	17.16s	20.81s
R-W W/ Semantic	5	5	32.94s	36.92s

Table 4.3: Results for the narrow corridor scenario of tests for validation of the Semantic Planner. Comparison of results with and without the use of the Semantic Planner both in simulation and real-world

4.3.4 Evaluation of parameters for narrow passage crossing

For this solution, the only changes considered for different plan sections were the valueas of the parameters of the local planner, specifically:

- Goal XY tolerance
- Goal yaw tolerance
- Max linear speed

The first two parameters are related to the tolerance given by the local planner to the defined goal. Tighter tolerances mean the final position of the IW will be closer to the goal, on the other hand, the planner might have difficulty in reaching the exact position of the goal and this can result in a long time to finish the trajectory, or even failure. The second parameter is simply the maximum linear velocity of the IW. Adjusting these parameters depending on the type the next intermediate goal of the plan might yield better results than using the same parameters for all navigation.

For these tests, different values for the parameters mentioned above are tested while crossing narrow passages. The objective of these tests is to find if the passage crossing performance can be improved by a different configuration of the local planner, since the ability to divide the path into different sections, by the semantic planner, allows the use of different values for the parameters in each section.

The parameters are set at the beginning of each path section, based on the type of intermediate goal. The types of intermediate goals, are explained in section 3.4.3 but are simplified for these tests by not distinguishing between doors and corridors. So the possible types of intermediate goals are:

- FREE_SPACE
- PASSAGE_ENTRY
- PASSAGE_EXIT

The default parameters used are the following:

- Goal xy tolerance: 0.5m
- Goal yawn tolerance: 0.5 rad

Tolerances \ Max Speed (m/s)	0.3	0.5	0.8
XY = 0.5 m Yaw = 0.5 rad	31.92s	28.01s	25.93s
XY = 0.1 m Yaw = 0.1 rad	60.82s	31.93s	31.94s
XY = 0.2 m Yaw = 0.2 rad	34.93s	30.94s	29.93s

Table 4.4: Results of evaluation of parameters for narrow passage crossing

- Max xy speed: 0.8 m/s

These defaults are always used except for 2 cases. When the goal is of type **PASSAGE_ENTRY**, the two tolerance parameters are changed. When the goal is of type **PASSAGE_EXIT**, the parameter for maximum speed is changed. This way we can control how much the chair tries to align with the passage entry and how fast it crosses the passage.

The scenario for these tests is the same as one of the scenarios used for a previous test and can be seen in Figure 4.5b. This scenario was chosen because it is the more challenging. The time taken to get from the start position to goal 1 was measured, this gives a good sense of the performance with different parameters.

The results for simulation tests can be seen in table 4.4, which correspond to the time to get from the start position to the goal represented in Figure 4.5b. In all these tests, the wheelchair could successfully cross the passage in this scenario. The main differences are related to time and comfort. With tighter tolerances for the entrance way-pose, the wheelchair, sometimes, has difficulty placing itself in the position within the tolerances, resulting in an awkward back and forward motion and possibly even the necessity to make a 360º turn to achieve the position which significantly increases the time taken to cross the passage. This does not result in greater success crossing the passage and therefore using tighter tolerance parameters for approaching narrow passages is not helpful for narrow passage crossing.

Decreasing the speed, while crossing the narrow passages, results, unsurprisingly, in a longer time to cross the passage. However, there can be some benefits to comfort form the lower speed. Further tests must be performed to find the ideal speed which may depend on the passage width, as well as the individual preference of the passenger.

4.4 Influence of Comfort Cost in semantic planning

As explained in chapter 3, the developed solution considers not only the distance, but also a comfort cost for semantic path planning. The study of the optimal implementation

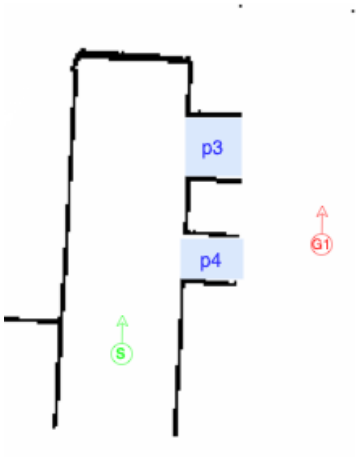


Figure 4.10: Scenario for testing the influence of comfort cost in semantic planning.

of previously presented equations 3.1 and 3.2, is not a subject of this dissertation, however, in order to test the functionality of semantic plan cost, this functions had to be implemented.

It was considered that the semantic plan cost would be calculated according to equation 4.1. The semantic plan cost for these tests is the sum of the distance cost (total distance of the path), with the sum of the comfort factors, defined in equation 4.2, for all crossed narrow passages, multiplied by a factor Kc representing the maximum distance in meters a narrow passage, with the highest comfort cost, can add to the cost of a path.

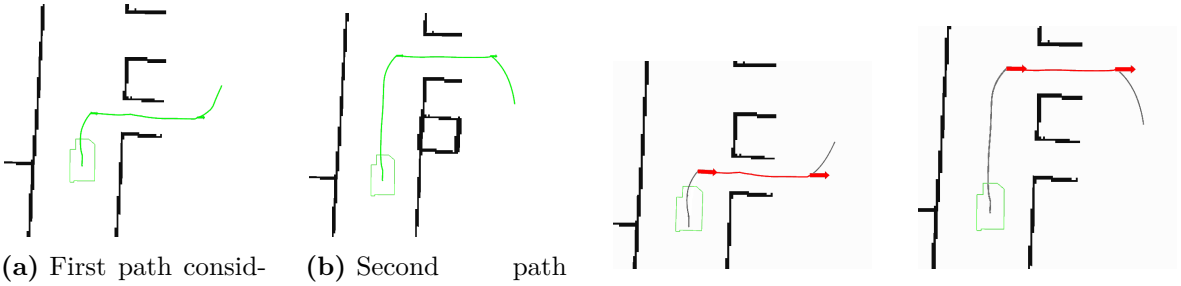
$$C = C_{\mathrm{d}} + Kc\, F_{\mathrm{c}} \tag{4.1}$$

$$F_{\mathrm{c}} = \frac{M - w}{M - m} \tag{4.2}$$

The scenario used for this test is the one represented in fig. 4.10, and depending on the value of Kc, the planner should choose to go through narrow passage P3 or P4 since the shortest path goes through passage P4 but passage P3 is wider and therefore, more comfortable.

Since these tests only relate to the ability of the semantic planner to choose the correct path and not the execution of said path, simulation tests were sufficient to validate this functionality. Two tests were performed, one with $\boldsymbol{Kc = 1}$ and another with $\boldsymbol{Kc = 10}$. It is expected that for the first test the semantic planner chooses the shortest path, through passage p4 and for the second test it chooses the more comfortable path through passage p3.

The *distanceCost* and *comfotCost* for the two possible paths are as following:



(a) First path considered by the semantic planner that goes through passage p4.

(b) Second path considered by the semantic planner that goes through passage p3.

(c) Final path chosen by the semantic planner for test 1 with $Kc = 1$

(d) Final path chosen by the semantic planner for test 2 with $Kc = 10$

Figure 4.11: Results of the tests of the influence of comfort cost in semantic planning.

- Path 1, through passage p4:
  $C_d = 5.49$
  $F_c = 0.75$
- Path 2, through passage p3:
  $C_d = 7.78$
  $F_c = 0.375$

When applying equation 4.2 to this costs with different values of Kc we get:

- $Kc = 1$:
  $C_{path1} = 6.23$
  $C_{path2} = 8.15$
- $Kc = 10$:
  $C_{path1} = 12.96$
  $C_{path2} = 11.61$

From these calculations, is to be expected that the semantic planner chooses **path1** for $\boldsymbol{Kc = 1}$ and chooses **path2** for $\boldsymbol{Kc = 10}$. This expectation is confirmed by the results in figure 4.11. In this figure we can see that the semantic planner considers both paths (figs 4.11a and 4.11b), and then decides the path to take according to the calculated costs.

This test showed that the semantic planner correctly analyses possible alternate paths by closing the narrow passage found in the first path. It also shows that it correctly calculates the total cost of each path for the given cost function and selects the most adequate path accordingly.

CHAPTER 5

Conclusion

The work developed for this dissertation resulted in a solution for improving the navigation of a IW in indoor spaces based on semantic navigation, particularly on the matter of crossing narrow passages. In this chapter, I discuss the chosen implementation and the results presented in the previous chapter, as well as reflect on the future work to be developed in this project.

5.1 Discussion

The results were generally satisfactory. The algorithm for automatically marking narrow passages correctly identified all narrow passages although it had some false positives due to imperfections of the OG map. This could be solved by a pre-processing of the occupation grid removing the artefacts that do not correspond to actual obstacles. The implementation of the algorithm is also not optimised for execution time and as such, it needs some time to run when the system starts and it needs a pre-generated map.

The semantic planner performed as intended and correctly divided the path into sections divided by intermediate goals and correctly passed these goals to the executor. It also considers correctly the comfort cost of each passage and chooses the trajectory accordingly. Using different values for navigation parameters, of the local planner, when crossing narrow passages, we obtained positive results but it did not have a determining impact on the success rate.

The objectives of this dissertation were achieved. Namely, with this semantic navigation approach, the wheelchair can cross narrow passages with an increased

success rate and approach each passage from different angles. The ability to use a different speed when crossing narrow passages can also increase passenger comfort. Although no experiments with subjects were performed to evaluate the comfort, some things were noticed while running the experiments for this dissertation. It was noticed that the semantic solution, by aligning with the narrow passage before crossing it, gave an increased confidence level to the passenger while without the semantic solution, there was a perception of bigger collision probability.

Although the results were not perfect when it comes to the performance crossing narrow passages, the results did validate the general architecture of the solution. The use of a semantic mapper and semantic planner proved to unlock several possibilities for navigation that are discussed in section 5.3.

5.2 Limitations

During the development and experiments processes, some limitations of the proposed system and of the wheelchair itself were identified. Firstly, relating to the used wheelchair, as can be seen in figure 4.4, the front wheels of the wheelchair are free-turning wheels, which can cause some issues. When the wheelchair completes a section of its path where it turns on itself, or moves backwards, these wheels can become in a side-ways position and, when the wheelchair then initiates a forward motion, this position of the front wheels causes an unpredictable, (from the plan executor point of view) sudden angular movement. This is especially noticeable when approaching narrow passages. When the wheelchair tries to position itself in the approach pose, this can force it to turn on itself or go a little backwards to align better with the narrow passage, then, when the wheelchair moves forward to go through the passage, the unpredictable position of the front wheels causes a different movement of the wheelchair than the plan executor expected. This is particularly problematic when trying to cross narrow passages, as we are working with tight margins and this unpredictable motion may cause the wheelchair to get into a position where it becomes impossible to cross the passage or even cause a collision. The back wheels of the wheelchair are also free-turning and cause a similar effect.

The sensorization of the wheelchair also as some limitations. The accuracy of the used LiDARs greatly impacts the indoor navigation capabilities. By increasing the accuracy and resolution of the LiDAR readings, the wheelchair could get a more precise localisation which greatly impacts the ability to cross narrow passages. The presence of encoders on the wheelchair's wheels would give better odometry data than the used method which would also improve the localization capabilities.

Secondly, regarding the implementation of the semantic plan cost, the final solution

is not a complete one. When the semantic planner identifies that the shortest path to a goal crosses narrow passages, it should consider all other possible paths to verify if there is any with a lower semantic plan cost. This problem gets increasingly complicated the more alternative paths there are to a narrow passage. The implemented solution is a simplification of this functionality that only considers one possible alternative path, by blocking all narrow passages crossed by the first path obtained from the global planner, and requesting a new path. The path with the lowest cost of the two is selected. A solution for this problem is discussed in section 5.3.

5.3 Future Work

The main success of the presented solution was the implementation of a semantic layer for autonomous IW navigation. This implementation allowed to use of a different strategy for crossing narrow passages. This strategy was to divide the path and place the IW in an approach position for each narrow passage that facilitated the crossing of said passage. Although this yielded good results, this strategy did not prove to be optimal and future work could involve the development of a specific path following algorithm for the narrow passage sections of the path. Other path following algorithms can also be added to the system as the semantic capabilities extend form only considering narrow passages, to also consider other situations as presented in chapter 1 like placing the chair partially under a table.

The semantic mapper correctly marks the narrow passages on the map but its functionality could be extended to include complete topological and semantic mapping. This would allow the semantic planner to create a plan based on the semantic and topological information without having to request a geometric plan form the global planner. This semantic planning phase would be much faster than the current solution since it wouldn't have to perform a path search on the entire metric map. A complete topological map would also solve the problem of considering the alternative paths according to semantic plan cost. The topological map could have on its edges all the needed distance and comfort cost information so the semantic planner could consider them in the semantic planing phase without having to block out passages and requesting plans from the global planner.

The sensorization of the wheelchair could also be revisited. The use of LiDARs with better accuracy, or the installation of encoders on the wheels, would probably increase the ability of narrow passage crossing as well as the indoor navigation capabilities overall. This assertion comes from the experience of performing tests with the wheelchair, were it was noticeable that the inaccuracies in localization and mapping significantly impacted the navigation.

References

[1] P. F. d. S. Figueiredo, "Navegação no interior de cadeira de rodas inteligente," M.S. thesis, Universidade de Aveiro, Dec. 2021.

[2] A. Elfes, "Sonar based real-world mapping navigation," *Robotics and Automation, IEEE Journal of*, vol. RA-3, pp. 249–265, Jul. 1987. DOI: 10.1109/JRA.1987.1087096.

[3] D. Holz and S. Behnke, "Sancta simplicitas - on the efficiency and achievable results of slam using icp-based incremental registration," Jun. 2010, pp. 1380–1387. DOI: 10.1109/ROBOT.2010.5509918.

[4] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (slam): Toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 125–137, 2001. DOI: 10.1109/70.928558.

[5] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2014.12.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889014003030.

[6] M. Hiller, C. Qiu, F. Particke, C. Hofmann, and J. Thielecke, "Learning topometric semantic maps from occupancy grids," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4190–4197. DOI: 10.1109/IROS40897.2019.8968111.

[7] P. Beeson, N. Jong, and B. Kuipers, "Towards autonomous topological place detection using the extended voronoi graph," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 4373–4379. DOI: 10.1109/ROBOT.2005.1570793.

[8] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000. DOI: 10.1177/02783640022066770. eprint: https://doi.org/10.1177/02783640022066770. [Online]. Available: https://doi.org/10.1177/02783640022066770.

[9] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, "Sensor-based exploration: Incremental construction of the hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 126–148, 2000. DOI: 10.1177/02783640022066789. eprint: https://doi.org/10.1177/02783640022066789. [Online]. Available: https://doi.org/10.1177/02783640022066789.

[10] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM (JACM)*, vol. 13, no. 4, pp. 471–494, 1966.

[11] P. YoonSeok, C. HanCheol, J. RyuWoon, and T. Lim, *ROS Robot Programing*. 1505, 145, Gasan Digital 1-ro, GeumCheon-gu, Seoul, Republic of Korea: ROBOTIS Co.,Ltd., 2017.

[12] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *2013 European Conference on Mobile Robots*, 2013, pp. 138–143. DOI: 10.1109/ECMR.2013.6698833.

[13] E. Pedrosa, A. Pereira, and N. Lau, "Efficient localization based on scan matching with a continuous likelihood field," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2017, pp. 61–66. DOI: 10.1109/ICARSC.2017.7964053.

[14] J. Crespo, "Arquitectura y diseño de un sistema completo de navegación semántica. descripción de su ontología y gestión de conocimiento," Ph.D. dissertation, Universidad Carlos III de Madrid. Departamento de Ingeniería de Sistemas y Automática, 2017. [Online]. Available: http://hdl.handle.net/10016/25277.

[15] T. Levitt and B. Kuipers, "Navigation and mapping in large scale space," *AI Magazine*, vol. 9, pp. 25–43, Jun. 1988. DOI: 10.1609/aimag.v9i2.674.

[16] J. Crespo, J. C. Castillo, O. M. Mozos, and R. Barber, "Semantic information for robot navigation: A survey," *Applied Sciences*, vol. 10, no. 2, 2020, ISSN: 2076-3417. DOI: 10.3390/app10020497. [Online]. Available: https://www.mdpi.com/2076-3417/10/2/497.

[17] R. Li, L. Wei, D. Gu, H. Hu, and K. D. McDonald-Maier, "Multi-layered map based navigation and interaction for an intelligent wheelchair," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013, pp. 115–120. DOI: 10.1109/ROBIO.2013.6739445.

[18] S. Cailhol, P. Fillatreau, Y. Zhao, and J.-Y. Fourquet, "Multi-layer path planning control for the simulation of manipulation tasks: Involving semantics and topology," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 17–28, 2019, ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2018.10.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584517301187.

[19] Z. Wei, W. Chen, and J. Wang, "3d semantic map-based shared control for smart wheelchair," in *Intelligent Robotics and Applications*, C.-Y. Su, S. Rakheja, and H. Liu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 41–51.

[20] B. Decoux, R. Khemmar, N. Ragot, *et al.*, "A dataset for temporal semantic segmentation dedicated to smart mobility of wheelchairs on sidewalks," *Journal of Imaging*, vol. 8, no. 8, 2022, ISSN: 2313-433X. [Online]. Available: https://www.mdpi.com/2313-433X/8/8/216.

[21] M. F. Pinto, I. C. d. S. Júnior, and A. L. M. Marcato, "Methodology for autonomous crossing narrow passages applied on assistive mobile robots," vol. 30, 2019, pp. 943–953. DOI: 10.1007/s40313-019-00499-2. [Online]. Available: https://doi.org/10.1007/s40313-019-00499-2.

[22] F. Leishman, O. Horn, and G. Bourhis, "Smart wheelchair control through a deictic approach," *Robotics and Autonomous Systems*, vol. 58, no. 10, pp. 1149–1158, 2010, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2010.06.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889010001272.

[23] Y. Morales, N. Kallakuri, K. Shinozawa, T. Miyashita, and N. Hagita, "Human-comfortable navigation for an autonomous robotic wheelchair," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2737–2743. DOI: 10.1109/IROS.2013.6696743.

[24] T. Sawabe, T. Okajima, M. Kanbara, and N. Hagita, "Evaluating passenger characteristics for ride comfort in autonomous wheelchairs," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 102–107. DOI: 10.1109/ITSC.2017.8317910.