uc3m | Universidad Carlos III de Madrid

PATH PLANNING IN ROS

By

RAUL SERRANO MARTIN

Department of Systems and Automation
UNIVERSIDAD CARLOS III

Supervisor:
Pablo Marín Plaza

JUNE 2018

Abstract

The imminent inclusion of autonomous vehicles in society has created a need for tools and human resources specialized in the subject. Usually universities and research groups have to start from scratch when it comes to research in the field of robotics and above all, autonomous driving. It is not easy to find a flexible environment focused on autonomous research that you can customize in a short period of time.

The main objective of this project is to develop a Local Path Planner integrated in a robotic operating system which posses modularity. For this, the problem of how to trace the path to be followed arises. Bézier curves are the right solution for its simplicity and its softness. An open middleware such as ROS (Robot Operating System) provides modularity to the project. The system works as follows: A parallel module define the initial and target conditions which are processed by the core instruction to define the path. Once an obstacle has been found along the way, the path is interpolated to an adjacent alternative and then published to the following instruction. Results show how given different initial and final conditions, the trajectory adapts to them. The structure of the system has been disposed in an intuitive manner so further advancement can be applied to the system. Other improvements may include taking into account the geometry of the vehicle when elaborating the path.

Dedication and acknowledgements

I want to dedicate this project to my family, especially to my mum and her supportive calls that make me stay on track. To Pablo Marin Plaza that guided me throughout the project and Eugenio Marinetto for selflessly helping me.

Acronyms

ROS	Robot Operating System
RVIZ	ROS visualization
NHSA	National Highway Traffic Safety Administration
RNDF	Road Network Description File
WHO	World Health Organization
AVs	Autonomous Vehicles
DGT	Dirección General de Tráfico
SDC	Self Driving Car
SDCND	Self Driving Car Nanodegree
LIDAR	LIght Detection And Ranging
SONAR	SOund Navigation And Ranging
RADAR	RAdio Detection And Ranging
Qt	Q toolkit
GPS	Global Positioning Sytem

Table of Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

The goal of this chapter is to describe the Autonomous Vehicles Market (AVM) and to show its current challenges in order to bring a background on the researched content. The Legal Framework has been studied around the globe to bring an idea of what the situation is in every localization. The steps followed to reach the results are introduced in the methodology and to conclude, the time and economic expenses.

1.1 Historical background

In 1925 the inventor Francis P. Houdina built up the first-considered "self-driving" car and demonstrated its functioning at New York City's public screens. Houdina installed an antenna in the car and used a remote control from a second car. As the New York Times reported back then, the vehicle started its engine and rotated through its various gears "as if a phantom hand were at the wheel"[8].

Later in 1950, electrical impulses were used for remote driving in order to detect the location and velocity of other vehicles as well as to provide information to autonomous cars. The problem was that they need receivers as they were built showing confidence that their implementation in roadways will become soon available and unfortunately, they did not. It was claimed that the technology could prevent 40 percent of road accidents, nevertheless, electronic roadways never became popular. [9]

Afterward, Cruise control was implemented in 1958 which allow the vehicle to maintain its speed without driver [10]. Anti-Lock Braking System (ABS) was implemented in cars after being used in aircraft since 1929.

The Sure-Brake System analyses the data from the wheel speed measurements to detect skidding and relay commands to a hydraulic modulator. All these automated techniques belong

to the "Level 1" of the NHTSA [1] classification. The NHTSA is an organization that in the 1980s created a new legislation promising cars cheaper to repair, safer and less polluting [11].

At a later time, one of the pioneers of the modern autonomous driving was the Stanford Cart during the 60s and 70s. It was conceived to be a moon rover project, a vehicle to patrol the moon. It consisted of a remote controlled TV-equiped robot [12]. His main advantage was the video processing technology. It processed images for ten to fifteen minutes each time it moved one meter.

Ernst Dickmanns, another of the pioneers of the self driving car and a professor at Bundeswehr University tunned a van to process visual information from cameras in the 70s. He extracted the edges from the images and analyzed them to extract information.

Dicksmanns also participated on the EUREKA PROMETHEUS project, in which the US government invested to hold the potential for safer and more lawful roadways.

The Stanley robot was the first place winner in the DARPA Grand Challenge in 2005. It was designed by Stanford University and Volkswagen reusing the technology achieved by the Stanford Cart. [13] Sebastian Thrun was the leader of the Stanford team and a visionary in this technology. He also led the development of the Google self-driving car.

After that, DARPA released another challenge to promote the advancement in the autonomous cars field, it was called the Darpa Urban Challenge. Some of the rules of the challenge were difficult to follow, to ease the situation all the teams were given a digital map of the environment. This map came in the shape of a RNDF (Road Network Description File), containing geometric information on lanes, lane marking, stop signs, parking lots, and special checkpoints. Teams were given aswell a high-resolution aerial image of the area, they were enabled to start the RNDF before the event. There were a number of vehicles carrying out the missions in the same environment at the same time, what was a factor for the vehicles to bear in mind. The new entry for this contest coming again from the Stanford University is called "Junior". It took its decisions based on a distributed software pipeline that integrates perception, planning, and control [29].

1.2 Socio-economic context

According to the World Health Organization deaths by car crashes is one of the leading causes of global deaths. More than 1.25 million people die each year as a result of road traffic crashes.

The new target in the 2030 Agenda[2] for Sustainable Development has set as a goal to half the number of global deaths from road traffic crashes by 2020. Without supported action and resources, the forecast states traffic crashes will become the seventh cause of death by 2030.

Road traffic crashes cost around the 3 percent of gross domestic product to most of the countries. Approximately half of those dying on the world's roads are “vulnerable road users”:

[1] National Highway Traffic Safety Administration: They promote vehicle safety innovations and set safety standards.

[2] A plan of action that seeks prosperity carried out by the WHO.

pedestrians, cyclists, and motorcyclists. Ninety percent of the world's mortality on the roads occur in low- and middle-income countries still, these countries have approximately 54 percent of the world's vehicles. The leading cause of death in people which ages are comprised between 15 and 29 years is Road traffic injuries.

People from 15 until 44 years old are most likely to die by a car crash accident accounting 48 percent of the global road traffic deaths. These numbers represent that males are more likely to be involved in road traffic crashes than females. Around three-quarters of the deaths occur among young males under 25 years [7].

Some of the risk factors are:

- Speeding

The increase of the speed is directly related to how likely is an accident to occur. An increase of 1km/h in mean speed results in an increase of 3 percent in the incidence of crashes. From the point of view of a pedestrian, the risk of dying is less than 20 percent if beaten down by a car at 50 km/h and almost 60 percent if hit at 80 Km/h.

- The effect of alcohol and other substances

In the case being drunk once driving, the risk of having a crash starts at low levels of blood alcohol concentration (BAC) and goes up significantly when the driver's BAC is > 0.04 g/dl [14].

- Other causes

As distracted driving, unsafe roads and unsafe vehicles [14].

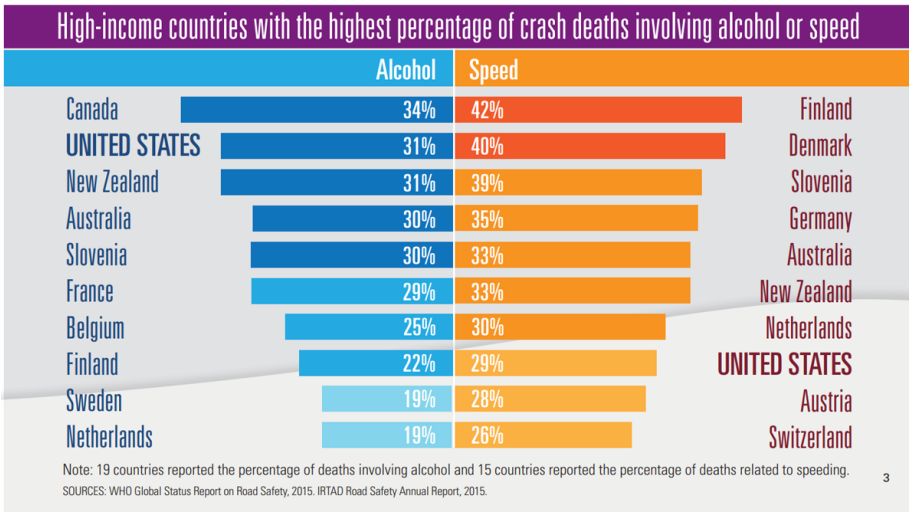

FIGURE 1.1. Graph of deaths

[15]

1.3 Legal framework

Autonomous driving has been a fast evolving field over the last fifty years. Due to this rapid change, some governments are finding it difficult to take measures fitted to the technology advancement. To learn more about the legal limitations in every territory each country should be studied one by one.

Lately in Spain there has been some efforts to regulate the autonomous tendency. DGT [3] and Mobileye, a company specialized in ADAS [4], have partnered to improve driving safety and to get ready to autonomous vehicles. They are part of Vision Zero, a Swedish initiative which target is to achieve no deaths or serious injuries [16]. In terms of the law, the legislation is very plain. The only sentence the Spanish legislation states is: "Only manufacturers and institutions can test vehicles without driver and only in certain ways" [17].

With respect to other countries of the European Union, Germany is the first country able to regulate Autonomous Vehicles within a legal framework. The German government has taken some measurements within the legislation on June 21st 2017. In case of a car accident, they ask for the black box record from the journey underway which has registered whether the cause of the accident was the human factor or the car's self-piloting system [18].

The legislation has created requirements to high-level autonomous systems. However, the driver must be ready to take over again if and when required. Despite this, this new approach does not allow autonomous driving when all of the occupants are merely passengers. As stated at the beginning of the section, there exist different regulations according to each country. That is the reason why some companies are striving to create an internationally harmonized legal framework.

At the international level there are many agreements that conclude a legal plan to improve each nation's legislation. It must be noted the one carried out in Vienna in 1968, although in this year automated systems have not been developed yet and consequently, no framework was defined accordingly. In that time it was considered that vehicles were only controlled by human drivers. ADAS has been allowed since a change in the law in 2016 but autonomous driving is not yet possible. [19]

The legislation to promote safety of driverless cars is to be ready by 2021 by the Law Commission in United Kingdom [20]. As the road minister Jesse Norman states, they are world leaders in self-driving research and developments. They point out that one of the portions of the challenge is that the vehicles can operate with safety and efficiency in all weather conditions [21].

Outside Europe, American automakers have a great success thanks to their regulatory framework "Unlike in Germany and Europe, legislation in the United States already permits highly automated vehicles to be used in a commercial context" [22]. Each state in the US is also

[3] Dirección General de Tráfico. It is an autonomous body under the Ministry of the Interior of Spain responsible for the execution of the road policy.

[4] Advanced driver-assistance systems: These acronyms have been adopted by automakers to refer to systems that help the driver within driving decisions.

responsible for it own legislation in terms of Autonomous Driving. California is the American top-ranked state in the acceptance of this technology, its regulations were presented in September 2014 which required a driver in the vehicle just in case he was required to take the control. Recently California announced SDC are no longer required to operate with a driver on its public road.

In Asia, Singapore is expected to be the first Asian country to adopt SDC as a whole. This is due to their high density of population, issue why the government is forced to take action. In 2017 they approved a legislation saying that motor vehicles don't require humans commanding it. South Korea is the most aggressive country when it comes to investment in Robotics Cars. The country is building an entire artificial town for autonomous vehicle testing called "K-City" [23]. To see the whole world situation at a glance, a map showing the limitations per country has been attached in Figure 1.2.

1.4 Methodology

As a first approach to the Autonomous Driving field, the Self-Driving Car Nanodegree[5] developed by Udacity was the first tool that brought the basic knowledge into the area. It allowed to perceived the advancements achieved regarding Robotics and Artificial Intelligence in the background of Robotics Cars. It contributed to the elaboration of the State of Art in Chapter 3 about Path Planners.

A bibliographical review through IEEExplore concluded that Bezier curves may be an effective alternative to design waypoints paths. To check its behavior a first algorithm was tested with Matlab offering a preview of its benefits and limitations. As this algorithm should be implemented in complex systems, we came up with the idea of using ROS to ease the integration process. Because of ROS only supports Python and C++ programming languages, C++ was chosen due to it is computationally efficient.

Once the system has been developed, it has been tested using Rviz and maps generated by images. To refine the results, some parameters were tuned in the code and tested again. To elaborate this thesis, LaTeX framework has been chosen due to it is a complex yet powerful tool to edit text documents.

1.5 Time planning

As for the prosecution of this project, a detailed storyline has been followed. The biggest workload has been carried right before and right after the completion of the Self Driving Car Nanodegree program due to the program was a high demanding key source of knowledge for the project. The Gantt diagram can be consulted in Figure1.3 It is a chart that describes the project schedule.

[5] An online credential program built with industry partners to bring you Autonomous Driving skills

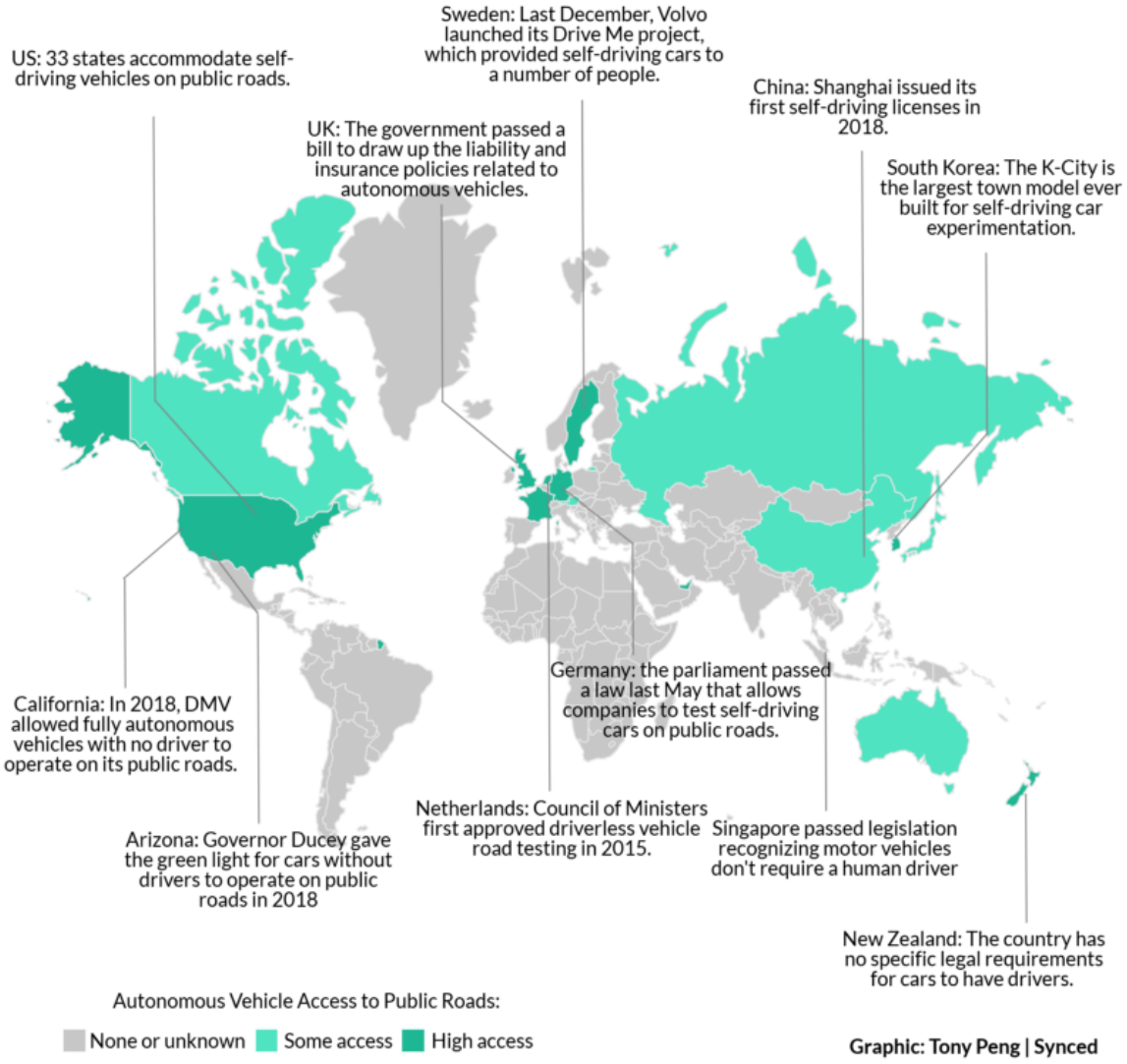

FIGURE 1.2. SDC World Legal situation. Figure taken from [23]

1.6 Budget

All the expenses associated with the project are described in this section. The final cost accounts 1970.53€

1.6.1 Personnel expenses

As defined in "XVI Convenio colectivo nacional de empresas de ingeniería y oficinas de estudios técnicos" the base annual engineering salary for an Engineering Bachelor holder in Spain is as minimum as 17,038.62€ per year with a maximum of 1800 working hours. Then the minimum salary is calculated as 17038.62€/1800h = 9.47€/h [24]. The salary accounted in this project can be calculated as 9.47€/h * 120h = 1136.40€

1.6.2 Software cost

The majority of the software licenses are established as open source software including Ubuntu, an open source Linux distribution; ROS, a set of software libraries; Qt Creator, the open source version; RVIZ, 3D visualization tool for ROS with a BSD license. Other software tools as Matlab has an annual cost of 800€, and Overleaf, the cost of the license used for the development of this thesis in LaTeX of 10.50€ per month.

1.6.3 Hardware cost

As the project prototype has not been tested in a real vehicle, hardware expenses more than computational devices cannot be considered. Taking into account that the advancement of the work has been performed in two different computers, both expenses have been noted. The Laptop cost 800€ in 2012. The desktop and its peripherals cost 1100€ in 2017. Estimating the linear depreciation of 4 years for the laptop and 8 months for the desktop, the cost has been established as 12.08€ for the Laptop and 1.05€ for the desktop. The table 3.1 has been drawn to summarize all the expenses.

Item	Cost
Personnel expenses	
Salary for a Junior Engineer	1136.40€
Software cost	
Matlab annual cost	800€
Overleaf monthly subscription * 2 months	21€
Hardware cost	
Laptop depreciation (4 years)	12.08€
Desktop depreciation (8 monts)	1.05€
Total cost	
	1970.53€

Table 1.1: Project cost

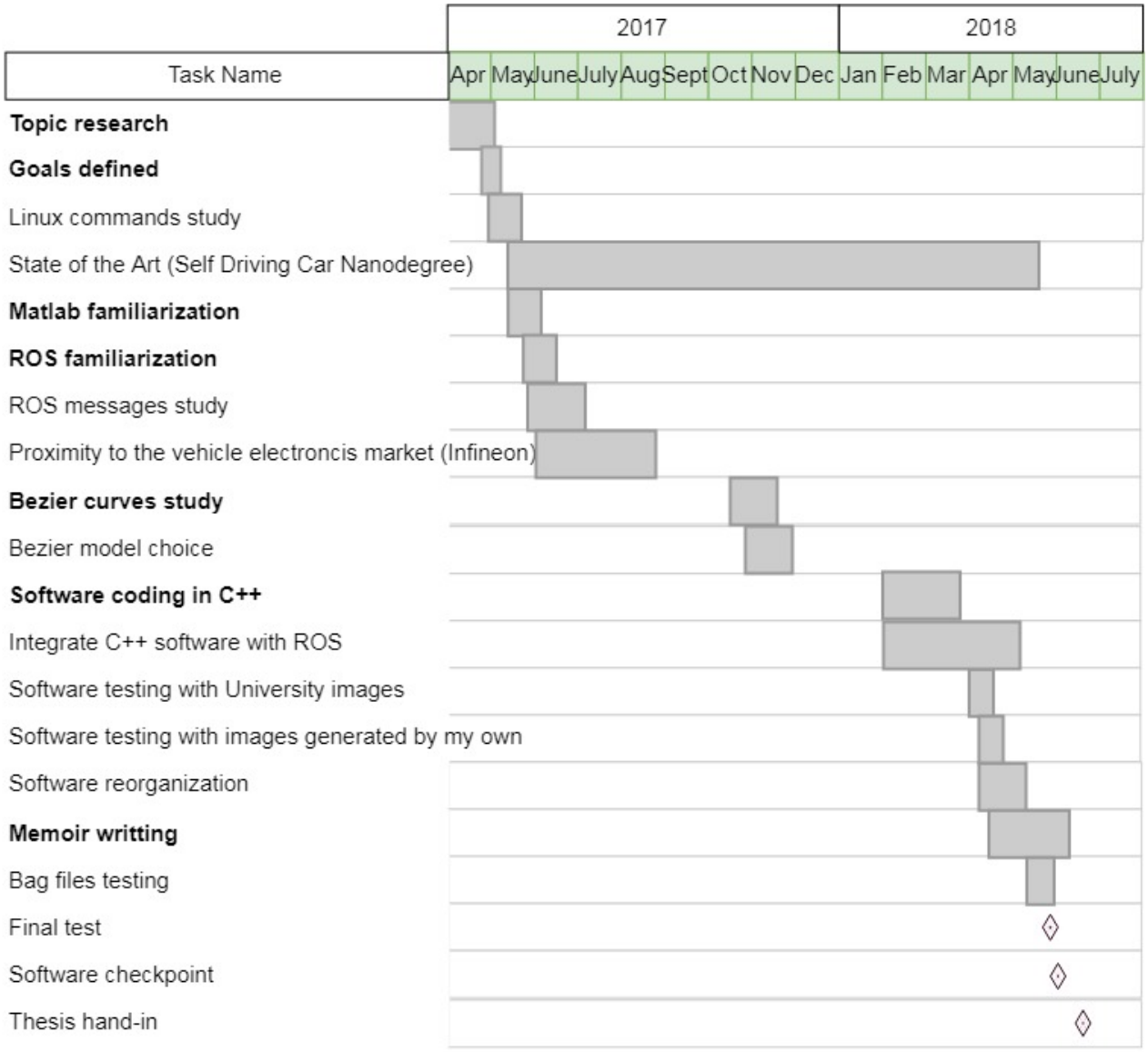

FIGURE 1.3. Gantt diagram

CHAPTER 2

MOTIVATION AND OBJECTIVES

2.1 Motivation

Traditionally the only individual in charge of the car could be a human. Driver assistance systems have been evolving during time starting from technologies like Adaptative light control to the more advanced systems such as ADAS [1]. Those systems, in combination with robotics may allow to accurately measure the current state of the vehicle and systems as well as predict its behavior.

There exist many planners like A* and Dijkstra that have been implemented in many programming languages, however only a few are properly integrated with ROS. Usually, these planners are designed for discrete situations, like grid maps or other means of discretization. In addition, some changing scenarios can assume constant capture of the obstacles that may lead to a higher computational cost. A modern vehicle concentrates a set of different sensors which needs to be supervised. A fully connected system working in parallel may not be efficient enough since some tasks only require certain data. The best case scenario would be a system in which the driver could pay attention to what information is interesting to you without the need of collecting the data from all the sensors for the analysis. All comes down to a local planner system with low-cost computational efforts that can be planned in ROS. Some Universities are demanding an Open Source Path planner [2] framework to integrate into its robots and start constructing over it and to adequate to its particular requirements. The hardest step is to create a universal environment which capabilities are not limited by the system design and where to start building. That is why

[1] Advanced driver assistance-system are mechanisms to help the driver with driving tasks

[2] It is a term used in robotics for the mechanism of dividing a route into small problems to solve

the aforementioned planner has been designed with an environment-friendly intention and easy to break down structure what brings endless possibilities.

2.2 Objectives

The purpose of this project is to develop a framework of a Local Path-planning system in ROS based on Bézier curves that is capable of detecting obstacles and avoid them offering other path alternatives. Specific objectives can be listed as follows:

- Analyze and identify the requirements of the needed path-planning system.
- Analyze and identify the structure of a ROS system.
- Elaborate a framework in ROS to publish Bézier curves.
- Create a Node[3] that publishes a trajectory given an initial and a target point.
- Adequate that Node to different reference frames using the ROS Transforms library
- Design a working prototype for the Obstacle Avoidance System and arrange it so it can be further extended in future studies.
- Test the prototype to check its functionality.

[3]It is a process that performs a computation in ROS

CHAPTER 3

STATE OF THE ART

Autonomous vehicles advancement was very limited before the 90's because companies and governments were not putting much money into it. The evolution of the information technology, especially Artificial Intelligence in the last decade has sparked more interest in the subject. The main focus of this section is to explain all the research already done in the field of Path Planning in Intelligent Vehicles.

Path planning in mobile robotics has been a subject of study for the last decades. Most of the authors divide the problem into Global and Local planning. A great number of navigation tools have been taken from mobile robotics and modified to face the challenges of road networks and driving rules. These planning techniques were classified into four groups, according to their implementation in automated driving: **graph search, sampling, interpolating and numerical optimization**. By the end of this chapter, some graphic examples have been disposed to ease the process of understanding each planner [25] .

3.1 Graph Search Based Planners

When traveling from A to B we can represent the space as a grid made out of cells, it is called Occupancy Grid. A Graph Searching algorithm visits each of the cells to give one solution to the path planning problem. The next algorithms are going to be discussed in the following subsections: Dijkstra's algorithm, A* algorithm and State Lattices.

3.1.1 Dijkstra's algorithm

Dijkstra was conceived as an algorithm to find the shortest way from a source vertex to all other vertices. It is a breadth-first-search algorithm because it explores all the neighbor nodes before

moving to the next depth level. The vertices are processed in increasing order of the distance from the source. It uses the linkdistance, a convention that measures the shortest path between two vertices [1].

The main idea from Dijkstra's algorithm is to note the vertices in expanding order of their separation from the source while building the shortest path tree peak by peak. At each step, it is added a new vertex following the construction of the shortest path to the current new vertex [1].

All the edges are given a distance value. A distance is the sum of all the edge weights of a path between our starting point and wherever vertex we are on. At the end of the algorithm, this distance will be the distance of the shortest path. The distance we start with is infinity which is just a placeholder value. The node we are starting with will have a distance of zero. The technique used is called "min priority queue" where the element with minimum priority is removed until reaching our goal point.

3.1.2 A* algorithm

The A-Star calculation can be utilized to discover an ideal way from the source to the destination goal effectively. It utilizes heuristics to decide and it performs superior to the Dijkstra calculation as for time. The A-Star calculation is a best-first search calculation that finds the ideal path from source to goal.

The A-Star calculation utilizes an expected cost function $f(n)$ to decide the request in which the search visits the nodes in the tree. The cost function $f(n)$ is the evaluated cost of the least expensive route through node n as takes after:

$$f(n) = g(n) + h(n) \tag{3.1}$$

where $g(n)$ is the cost from the source node to the current node n, and $h(n)$ is the heuristic cost of the path from the node n to the destination node. Among the candidate nodes, the node n with the smallest $f(n)$ is the first to be selected to check [2].

3.1.3 State Lattices

The algorithm uses a discrete representation of the planning area with a grid of states (often a hyper-dimensional one). This grid is referred as state lattice over of which the motion planning search is applied. The path search in this algorithm is based in local queries from a set of lattices or primitives containing all feasible features, allowing vehicles to travel from an initial state to several others. A cost function decides the best path between the precomputed lattices. A node search algorithm is applied in different implementations.

3.2 Sampling Based Planners

The goal of these planners is to solve timing constraints. It randomly samples the configuration space looking for connectivity inside it. One of the consequences of these planners is that the solution is suboptimal.

The most commonly used methods in robotics are the Probabilistic Roadmap Method (PRM) and the Rapidly-exploring Random Tree (RRT).

3.2.1 Probabilistic Roadmap Method

The basic idea is to take random samples. The aim consists of capturing the connections of a free space with a number of segments. To be accurate, it is about constructing a graph whose edges belongs to segments within the free of collisions configuration space.

The simple method to construct a probabilistic roadmap is formed by three parts. The first one refers to draw milestones in the configuration space at random, according to a uniform distribution. The edges are created along the second part by searching for a particular milestone just in case there exist milestones close enough that can be linked with it by a path in the free space.

There is a risk to obtain several nonconnected sub-graphs if too few milestones have been sampled near or inside the narrow passages. The objective of the third phase, named as expansion phase, is to localize this problem [26].

3.2.2 Rapidly-exploring Random Tree

The main idea from the RRT algorithm is to expand the leaf nodes of a tree through random search from an initial node in the state space of the system. When the goal has been reached by the node, the search is stopped and an extended random tree is generated [4].

3.3 Time Elastic Bands

In this method, the path is treated as an elastic line what means that it is able to modify its shape. It consists of calculating the force acting on the elastic band by means of taking discrete points and calculating its gradient of the potential energy. These forces come from obstacles located in the surroundings and they can be classified into three different types as shown in Figure 3.1 [27].

3.4 Interpolating curve planners

Once the trace to follow is defined, a solution to do it in a smooth way should be adopted. Through interpolation new data is created given a range of points. This process generates a new set of

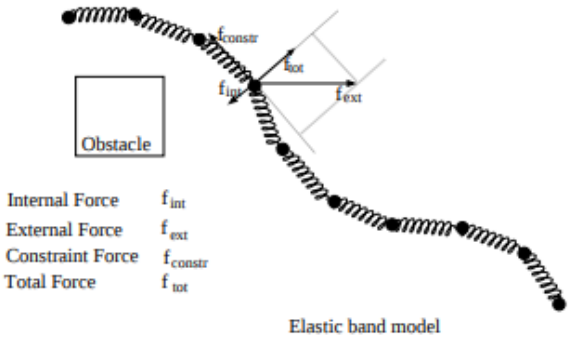

FIGURE 3.1. Elastic Band forces. Figure taken from [27]

dots that, in case they cross obstacles in the vicinity, it is enough to iterate (as it has been done along this project) some of the points to other points that do not collide with anything [25].

A table that summarizes all the interpolating curve planners has been attached to offer a small description from each one.

Technique	Description
Line and Circle	Given known way-points the road may be represented by straights and circular forms.
Clothoid Curves	Useful to design trajectories with linear changes in curvature because that curvature is the same as the arc-length. It allows to smooth trajectories between straight segments to curve ones.
Polynomial curves	They fit the position, angle and curvature to the constraints needed.
Splines	A parametric curve split into intervals that can be considered polynomials.
Bézier curves	This technique is the focus of this project. A parametric line that models smooth curves.

Table 3.1: Interpolating curve planners

Figure 3.2: Dijkstra algorithm [1].

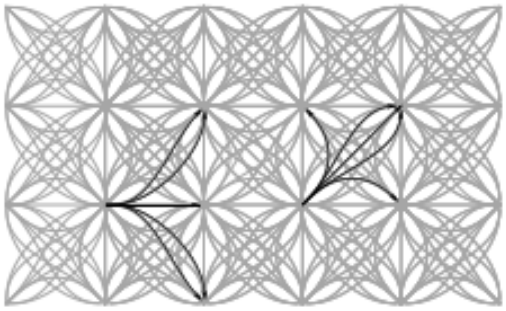

Figure 3.3: Lattice algorithm [2]

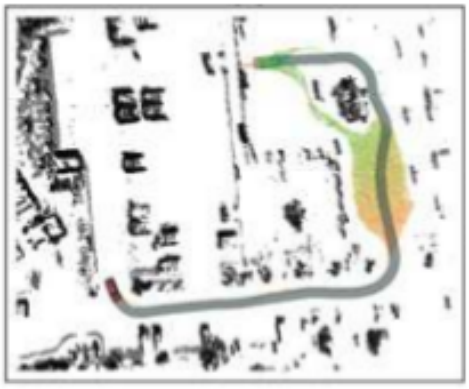

Figure 3.4: A star algorithm [3].

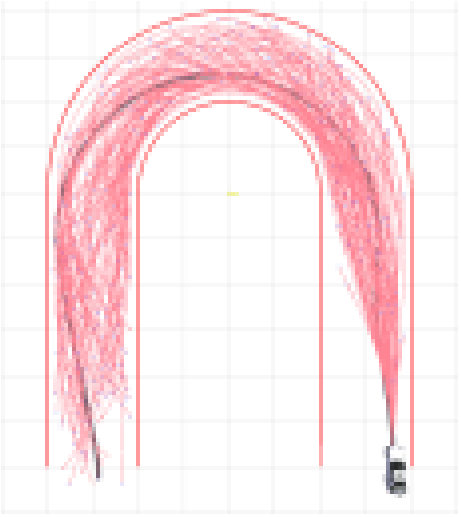

Figure 3.5: Rapid-exploring Random Tree [4]

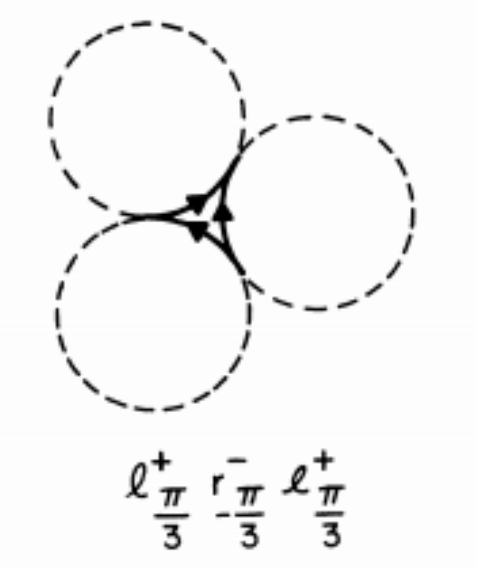

Figure 3.6: Line and circle

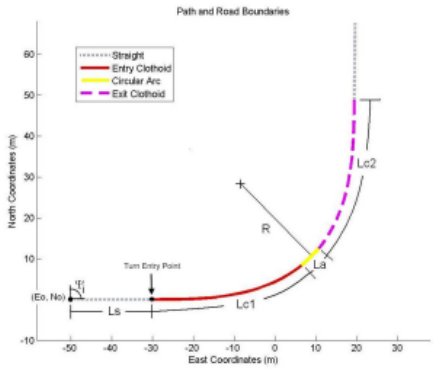

Figure 3.7: Clothoids [5]

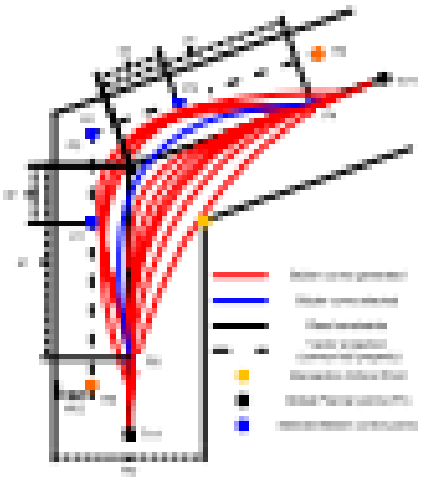

Figure 3.8: Bézier curves [6]

Figure 3.9: Spline [7]

CHAPTER 4

SYSTEM DESIGN

It is going to be demonstrated how the system has been designed with modularity and easy to grasp explanations for other readers and researchers to understand. The steps to adequate each part of the system to the project are noted using the Systems Engineering approach. As mentioned in the Motivation Section 2.1, the system has been designed with the help of ROS. It allows communicating every part of the system independently without the risk of depending on a broken piece or to design the system as a whole.

Furthermore, ROS license is under the standard three-clause BSD license. It is a very permissive open license which allows you to reuse apart from open source projects, a reuse in commercial and closed source products. As it is very standardized it presents the opportunity to connect and collaborate with other roboticists. It has been designed to be as distributed and modular as possible. There are over 3,000 packages in the ROS ecosystem, covering everything from proof-of-concept implementations of new algorithms to industrial-quality drivers and capabilities [28]. These packages go in loyalty, covering everything from confirmation of-idea usage of new calculations to mechanical quality drivers and abilities. The ROS client network expands over a common foundation to give a joining point that offers access to equipment drivers, non-specific robot capacities, improvement tools, helpful outer libraries, and that's only the tip of the iceberg. [29]

4.1 System Requirements

The process to establish the requirements of the model must be the first step since it allows to foresee the time and the material resources to define the cost and schedule the work.

In this section the requirements that directly apply to my path planer are presented and next to them a brief description of them:

Requirements	
Modularity	A posterior implementation of new modules is opened to further development.
Scalability	It can be implemented and adapted without the lose of quality.
Independence	The planner shall work independently of the robot .
Integration	The planner shall be self-contained within the robot.
Obstacle detection	The planner shall identify the obstacles within its environment.
Collision avoidance	Once the obstacle is detected a proper path shall be thrown to deviate from it.
Communication with the actuators	The actuators shall perform the movement indicated by the planner.
Communication with the sensors	The sensors shall provide the information regarding the robot's localization and environment situation.
Visual accessibility	The output from the planner shall be accessible from a visual environment such as RVIZ.
Human commanded	In the end, a human shall be allowed to start and stop the operation of the planner in case of need.
Computational efficiency	The system shall perform the calculations within minimum processing requirements.
Path optimization	The planner should prioritize small distances within the limitations of the environment.
Adjustment to the robot's kinematics	The movements shall be performed by the robot's geometry.
Unplanned events	The robot shall know how to respond to unforeseen events

Table 4.1: Requirements table

The table contains the requirements as a guideline for the implementation stage. Some of the requirements mentioned above have been written during the System Implementation Chapter 5, thus the table has been modified later.

4.2 Flowchart analysis

A flowchart is a logical diagram, an intermediate step to define a work-flow or an algorithm. It has to represent an easy way to understand the system as a purpose. It allows to identify different stages of the problem and design a feasible implementation to solve it. The outcome is an easy way to perceive the problems and the implemented solutions.

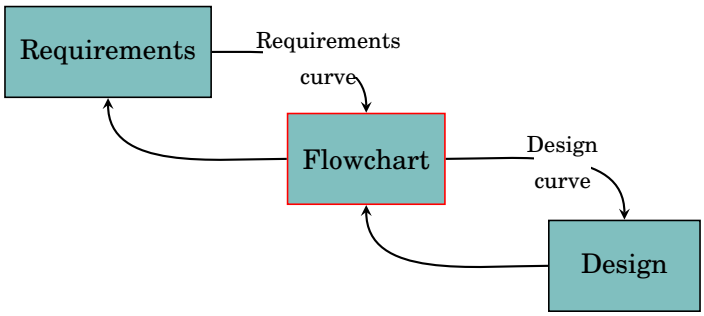

Figure 4.1: The Logical Decomposition phase

4.2.1 Functional design

During the Flowchart Analysis, the system has been decomposed to a logical high-level view. This representation aims to create the work-flow (Figure 5.3) which establish connections with the requirements and the final system design. Following an iterative process during the design, any incongruence should be fixed and labeled to be implemented in the flowchart.

4.2.2 Flowchart decomposition

As the requirements of the system have been already displayed, the next step is to represent in a sequential manner the different tasks and its mission during the performance of the planner. To that goal, a Flowchart diagram is used. According to the Merriam-Webster dictionary: "a diagram that shows step-by-step progression through a procedure or system especially using connecting lines and a set of conventional symbols" [30]. Each of the different shaped blocks defines a function. Each block may have different inputs and outputs and are connected by lines and directional arrows.

There are two shapes: those with rounded ends represent the start and end points of the process and rectangles are used to show the interim steps. The diamond symbolizes that a decision needs to be made. If the rectangle shows two lines on its sides means that the process has been performed inside the node being the opposite in any other way.

The first flowchart represents the tasks performed by the main node. Its goal is to receive the information from other topics and once it has it, publish a path in the given circumstances:

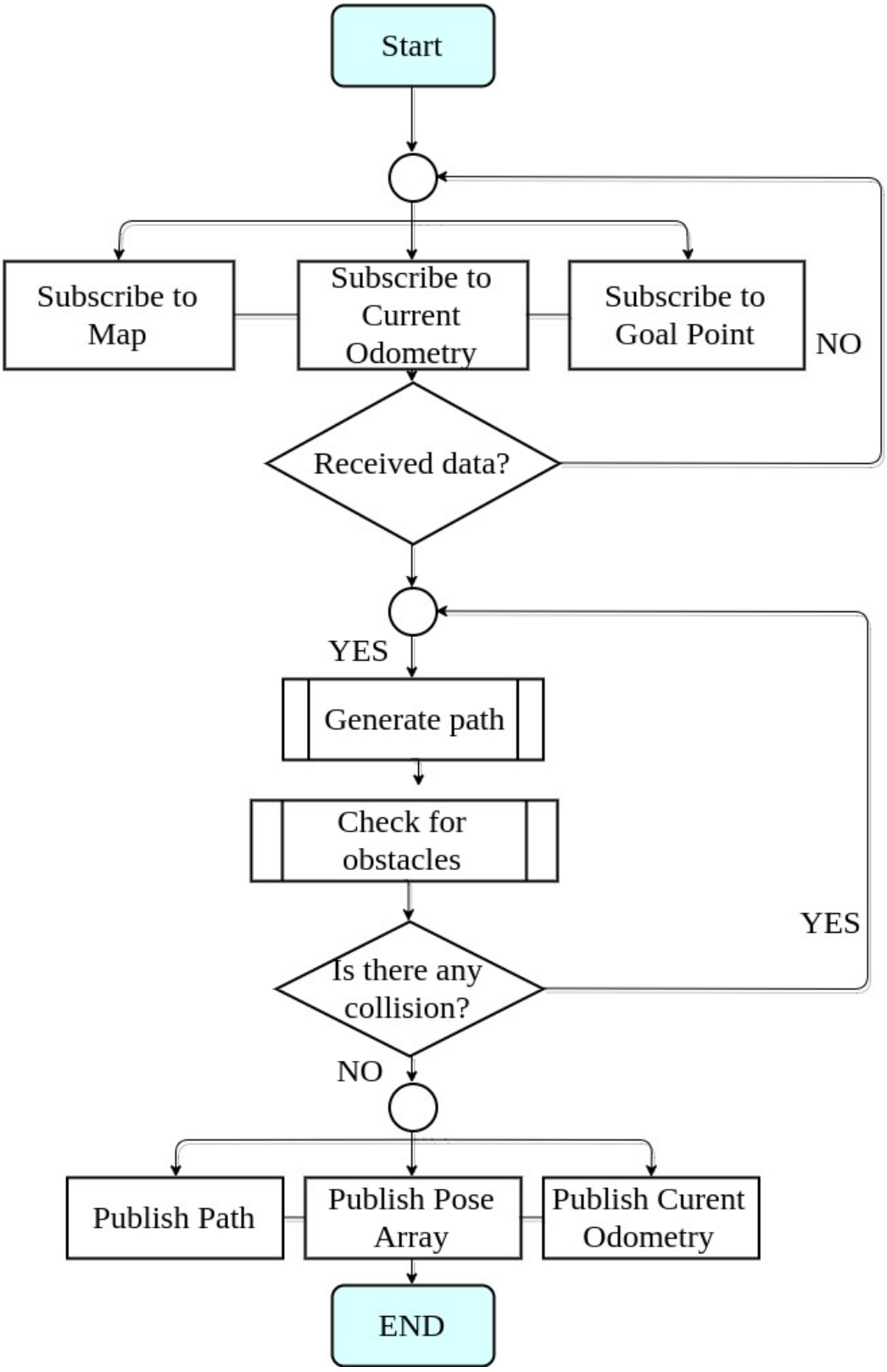

FIGURE 4.2. Main flowchart

4.3 ROS organization

As commented in Chapter 3, "Nodes" are processes that perform computation and "Topics" are buses where nodes exchange messages. The next workload must be represented in terms of ROS Computation Graph Level:

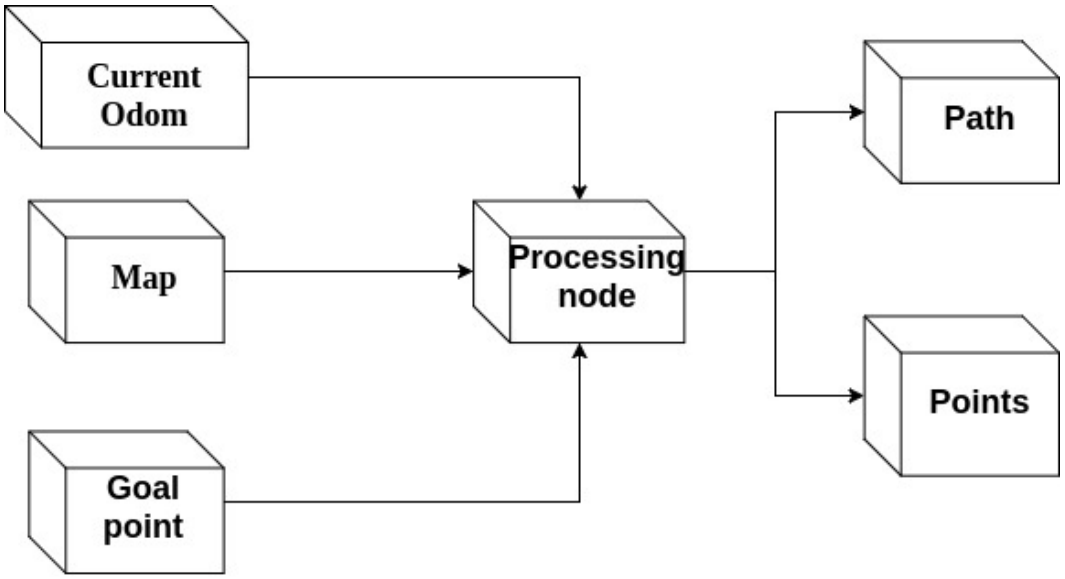

FIGURE 4.3. Graph level scheme

- Current odom: It describes the current position of the robot with respects to the starting location. In either case, it provides the localization in the Euclidean space as the orientation expressed in quaternion form. This block represents a message so it must be plotted as a "topic".

- Map: It is a ".png" image which empty spaces are represented by white colors and the obstacles colored with black or gray color scheme. It is a "topic" as well.

- Goal point: It must be the final location of the vehicle. It has to be reached with a certain position and orientation. The "Goal Point" represents the third topic received by the node.

- Processing node: Represents the computational workload so it is defined as a node. It has to deal with different types of messages thus some type of conversions are a prerequisite. It depends on the topics described above and the final result is published on the "Path" and "Points" topics.

- Path: The place where it is published a special type of message in ROS that defines the way to travel. It symbolizes the different locations and orientations the vehicle is going to tackle along the way.

- Points: It embodies the same as the message above but the points are represented differently without the need of forming a path.

4.3.1 ROS Master

Apart from the usual nodes, there is another type of node that must be executed at the very beginning of the software execution called "Master", thus, another kind of graph that shows its connections with the rest of the blocks must be plotted. The function of the Master is to allow individual ROS node to allocate and communicate with each other. A normal sequence of events will start with one of the nodes of the system, for example, the one that tracks the current odometry notifying the master that it wants to publish a localization and orientation on the topic "current odom". At the beginning nobody may have subscribed to that topic yet no data is really sent. Let's imagine that now "Planner node" wants to subscribe to the topic "current odom", then the master notifies both nodes about each other so they can start transferring data.

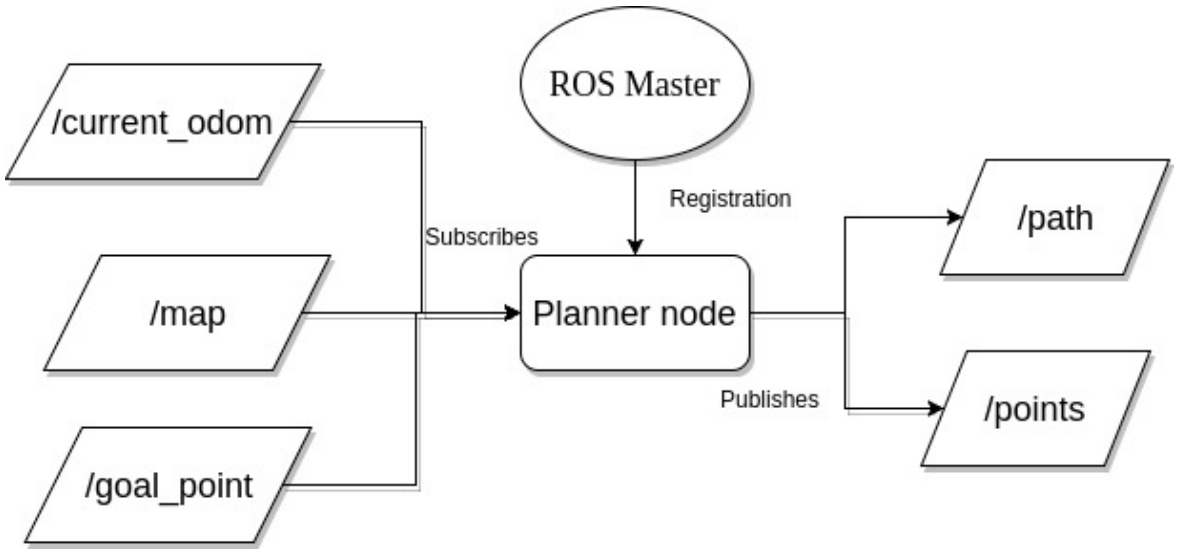

FIGURE 4.4. Graph level scheme

4.4 Coordinate frames and Transforms

A frame can be defined as a coordinate system within the ROS environment. Coordinate systems in general are right handed and in 3 dimensions, thus ROS copies the idea with X forward, Y left, and Z up.

The relationship between two frames is represented by a 6 DOF relative pose, a translation followed by a rotation. If W and A are two frames, the pose of A in W is given by the translation from W's origin to A's origin, and the rotation of A's coordinate axes in W [31]. A rotation matrix has no tf type; instead, tf represents rotations via tf::Quaternion, equivalent to btQuaternion. The bullet quaternion class has tools for constructing quaternions from rotation matrices and vice versa.

The following figure represents the whole tf's configuration in the system where the path planner is going to be tested:

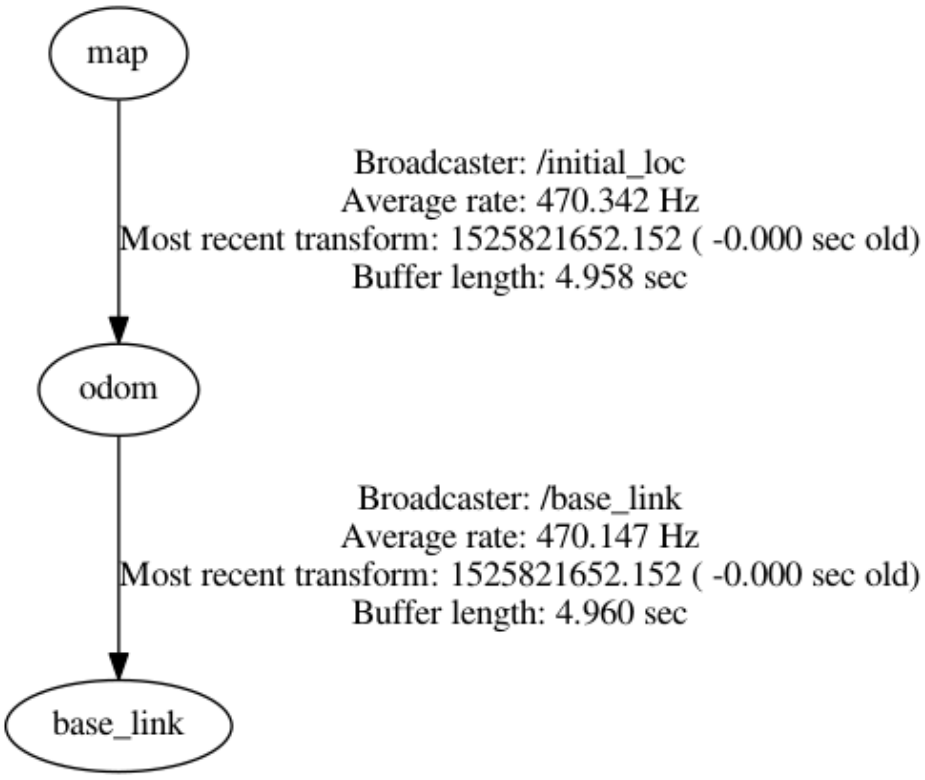

FIGURE 4.5. Tf transform tree

The Broadcasters (initial localization and base link) have been designed within a launch file. The average rate tells about the frequency those transforms are being published and then the most recent transform when the caption was taken. The buffer length informs about how many

seconds the data is available.

- map: Refers to the Global Reference Frame. It alludes to where the coordinates system is located into the map
- odom: Refers to Odometry. The position of the robot from its starting position, in this case from the map position. This transform is obtained by means of sensors disposed along the robot's body.
- base link: Reference frame of the robot itself.

To have a better understanding of where does "base_link" comes from, the following figure tries to schematize the localization of each of the reference frames from each of the robot's sensors in the system where the planner is integrated, the iCab. The disposition is represented in Figures 4.6 and 4.7

4.4.1 ROS Transform Library

When the robot is sensing the environment around itself, it can get some idea of what it looks like and where things are located, but it must bear in mind all that data is referenced to the location of its particular sensor. The same occurs when the path to follow has been computed, it must be referenced to a particular localization within the robot and that localization referenced to a point within the map. To eradicate all these problems it was came up with the idea of creating a library that keeps track of the different transformations and localizations. It has the aim of providing a standard method for monitoring the different coordinate frames. With respects to the iCab Path Planner, the Transform library will be in charge of drawing the path not from the "map" reference frame but from the "base_link" as explained in Chapter 5.

4.5 Type of messages used in ROS

ROS uses a language to simplify the way in which data values are described, also known as messages. It makes it easier for ROS tools to automatically create source code for the message type in different desired languages. A message is a simple data structure composed of different typed fields. Some of the standard types that are supported are integer, floating point and boolean. The powerful side of messages in ROS is that they can include nested structures and arrays [32].

4.5.1 Header field

Some of the messages that have been implemented include a special type of message called 'Header'. It includes some metadata fields such as timestamp and frame ID. It can be distinguished three fields in a usual 'Header' message.

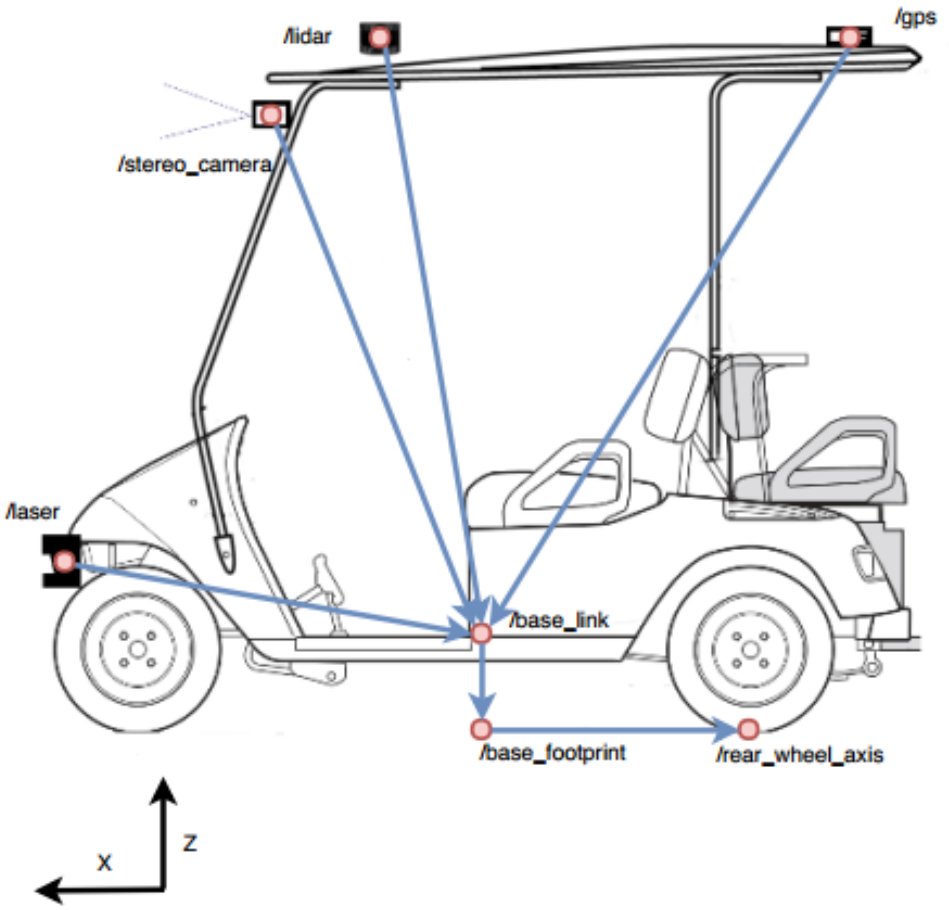

FIGURE 4.6. iCab side view

- The 'seq' field refers to an id that is increased automatically whenever a message is being sent from a client.

- The 'stamp' field saves time information. It must be related to data in a message. In the case that, for example, an image has been taken, the 'stamp' may refer to the time in which the image was captured.

- The 'frame id' saves information about, as the name already explains, the frame information that must be associated with data in a message. Going back to the example of the photography, it refers to the frame in which the photo has been taken.

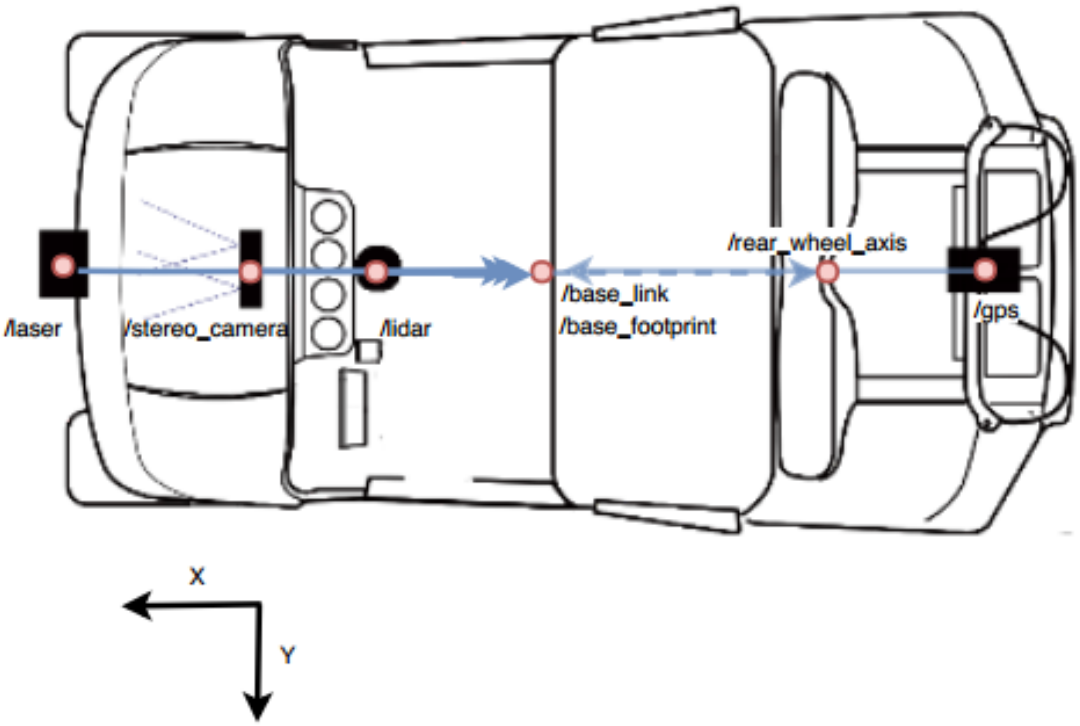

FIGURE 4.7. iCab top view

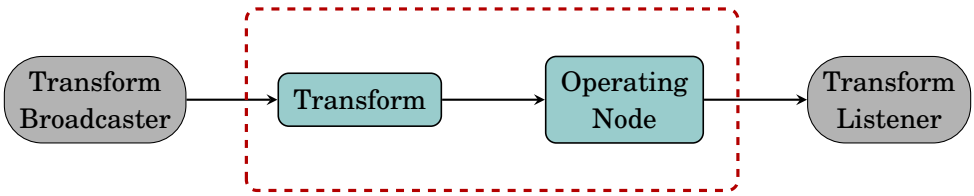

Figure 4.8: Transform Broadcaster and Listener

4.5.2 Navigation messages

Navigation messages, in ROS defined as 'nav_msgs' defines the common messages to interact with the navigation stack. The navigation stack is a package that gathers information from odometry, sensor streams and target pose and returns some safe velocity commands which are sent to the robot. Specifically for this project, it is of particular interest the 'nav_msgs/Path' message because it has to deal with a series of positions and orientations that are going to be used to describe the path from the initial to the target goal . The definition for the raw message can be said as *an array of poses that represent a Path for a robot to follow*.

The diagram showed in the figure 4.9 shows the dependence of the images between them. Note that navigation message is a nested structure that depends upon other nested structures, thus when a subordinate structure is repeated twice as it is the case of 'Header', it hasn't been extended more than once. The rest of the types of messages included in this structure are explained below.

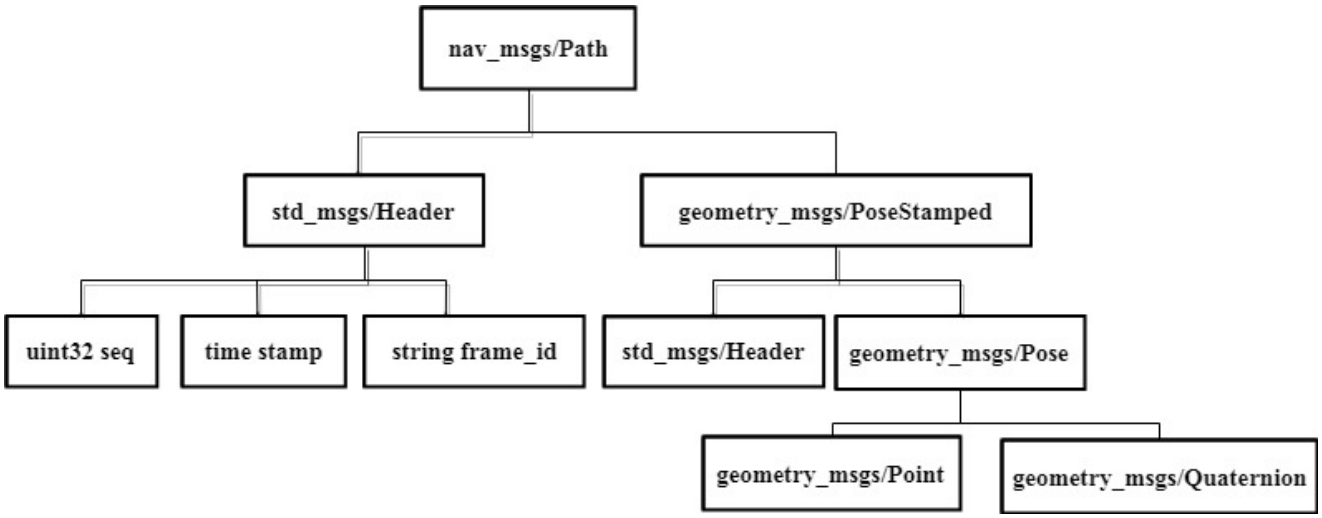

FIGURE 4.9. Navigation messages chart

The other type of navigation messages implemented is 'Odometry'. It is used to represent an estimate of a position and a velocity in the free space. The pose must be specified in the coordinate given by the header. The twist must be specified in the one given by the child frame id. For an easy understanding of how is the Odometry structure distributed, a diagram has been attached aswell in the figure 4.10.

4.5.3 Geometry messages

The geometry messages 'geometry_msgs' package provides messages for geometric primitives that need to be described such as points, vectors and poses. It facilitates the interoperability throughout the system and provides a common data type [33]. The geometry messages package contains other structures that have been used: Point, PoseStamped, PoseArray, Quaternion and Transform. The relationship among these elements can be observed in figure 4.12.

Some of the messages have not been shown, in particular, those that depend on Point and Quaternion because they have been considered to be redundant. Those messages represent floating points for the position and orientation in three dimensions. [1]

[1] x, y and z for the position and x, y, z and w for the orientation.

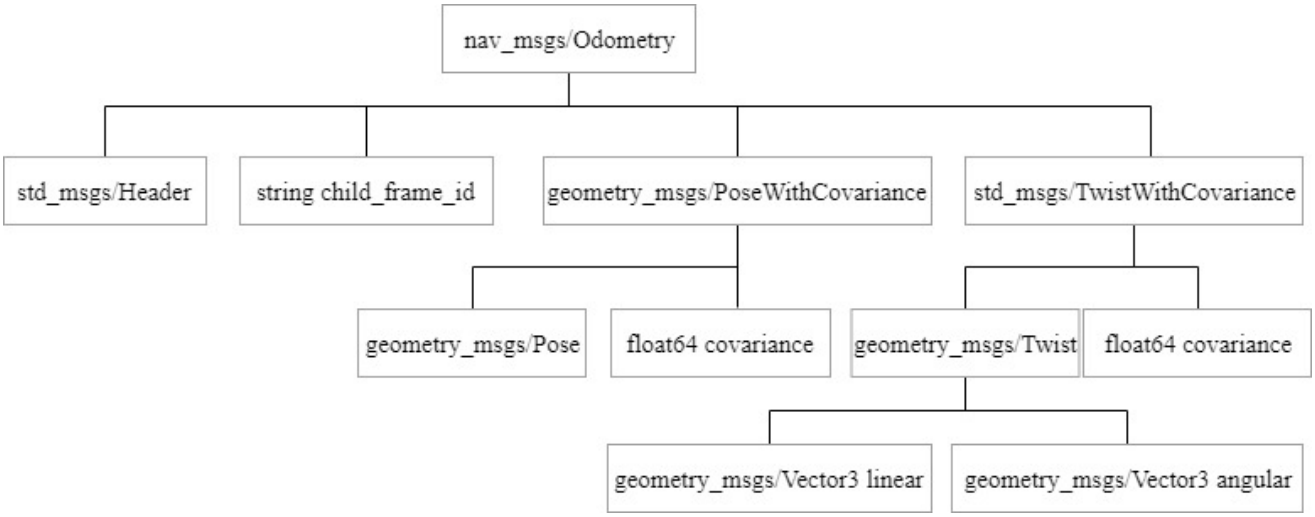

FIGURE 4.10. Odometry messages chart

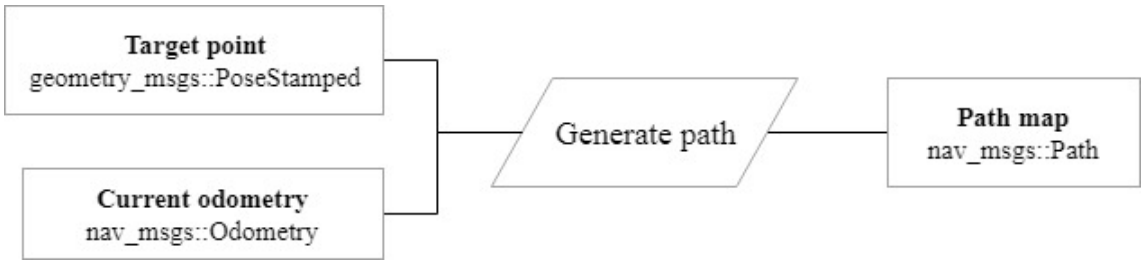

FIGURE 4.11. Generate path system

4.5.4 Visualization messages

Some high-level packages such as RVIZ allows representing graphic elements in the three-dimensional space. That makes easier for humans to understand how the model is working at a glance. In this context, a type of package that works with visualization data, Visualization messages (visualization_msgs) has been designed particularly. The type of message from this package that has been used the most along this project is the Marker message. It allows visualizing a set of elements in the shape of boxes, arrows, spheres etc.

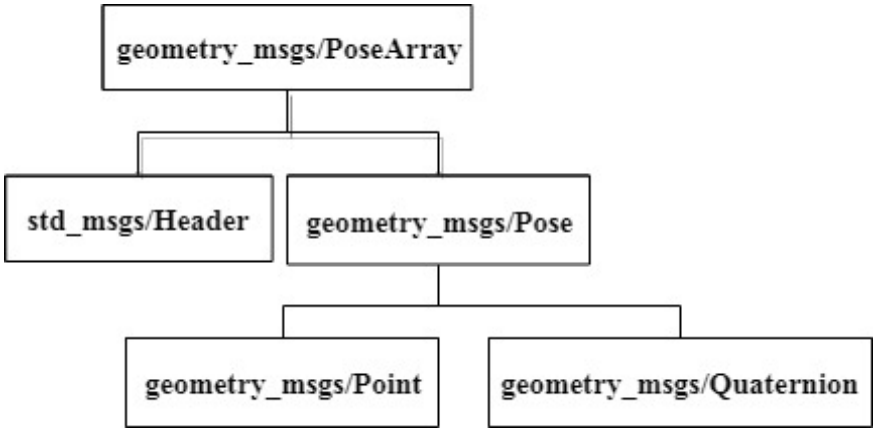

FIGURE 4.12. PoseArray messages chart

CHAPTER 5

SYSTEM IMPLEMENTATION

The actual realization of the system is going to be described in this section. The process that has been followed for the planer will be explained in chronological order of the development starting with the main idea and highlighting the different ways of approaching the problem. All the software design is being explained step by step in the shape of "pseudocode". As some authors state: "My point is that you should use the right tool for the job. For design documents intended for technical and nontechnical review, use pseudocode predominantly". [34] Pseudocode is used to make sure everyone can understand the behavior of a system, it summarizes a program's flow, but excludes underlying details. It uses short terms or simple English language syntaxes to write code for programs before it is actually converted into a specific programming language.

In every stage different programming languages have been chosen, they will be detailed next to its correspondent pseudocode. Although the final results are going to be exposed in the next chapter, the partial results and its conclusions will be attached next to its method.

5.1 Early stage

5.1.1 Matlab Bezier curves script

As a first adoption, it was needed to check if a Bézier curve was a feasible tool to work with. For doing so it was decided to search for a script to check its operation and to get in touch with the field.

This algorithm is developed with the aim of a tool called in Matlab "The Kron function" [35]. It gives us all of the possible products of the elements of two arrays. The reason to use this is that the function is going to be expanded to a higher order combination of three or four points.

Algorithm 1 Bezier script

procedure MATLAB PROCEDURE
    $pt1 \leftarrow$ coordinates of point 1
    $pt2 \leftarrow$ coordinates of point 2
    $pt3 \leftarrow$ coordinates of point 3
Define a quadratic Bezier curve:
    $function \leftarrow (1-t)^2 * pt_1 + 2(1-t)t * pt_2 + t^2 * pt_3$
    $t \leftarrow$ *(0-1)*
Plot:
    plot $\leftarrow$ pt1, pt2, pt3

With the "plot" function the result can be then showcased:

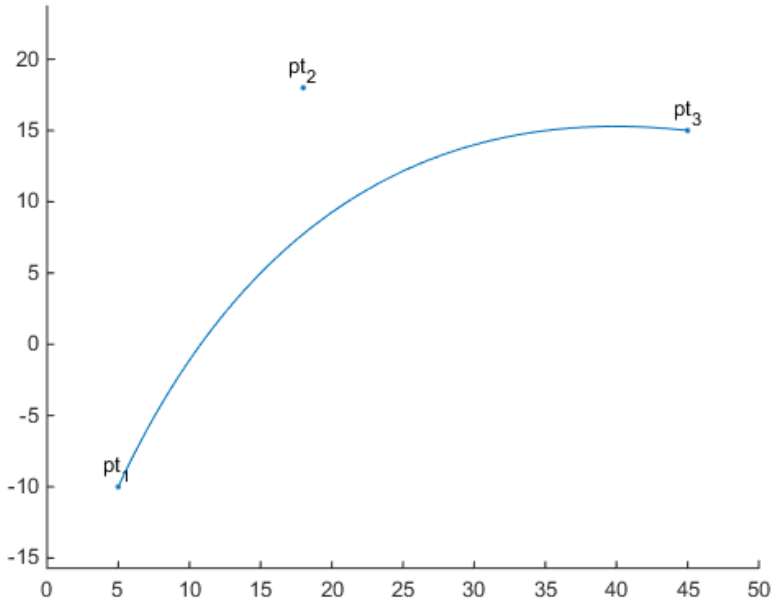

FIGURE 5.1. Quadratic Bézier curve in Matlab.

Bear in mind that the line obtained starts at $pt1$ and ends at $pt3$. In the middle, it goes to, but doesn't reach, $pt2$. As the function represented above, functions of different degrees can be plotted.

5.2 Development in C++

5.2.1 Bezier function

As it was already checked the algorithm worked in Matlab, now it is time to test it in C++. For the first trial, the following algorithm was taken as a referral. To get started, a function that returns a series of factorial numbers depending on what number you choose as an input.

Algorithm 2 Factorial

```
1: procedure C++ PROCEDURE
2:     factorialLookup ← vector from a list of 33 factorial numbers
3:     i ← number to get a factorial from
4: function:
5:     if i < 0 then return error
6:     if i > 32 then return error
7:     else
8:         factorialLookup() ← i.
9:     return factorialNumber.
```

It is needed to calculate the Binomial $\binom{n}{i} = \frac{n!}{i!(n-i)!}$
The code above is used to return the factorial from each number of the numerator and the denominator. Now it is time to implement those numbers in the actual binomial.

Algorithm 3 Binomial

```
1: procedure C++ PROCEDURE
2:     a1 ← factorial(n)
3:     a2 ← factorial(i)
4:     a3 ← factorial(n-i)
5: function:
6:     Ni ← a1/(a2*a3)
7:     return Ni.
```

Now that we have the binomial part, all the elements required to calculate the Bernstein polynomial are ready. As a reminder from the theory, Bernstein polynomial has the following shape:

$$\vec{x}(s) = \sum_{i=1}^{n} \vec{b}_i B_i^n(s) \tag{5.1}$$

Then now the missing part is $t^i * (1-t)^{(n-i)}$

The factorial algorithm previously defined is the basis for the next Bernstein Polynomial.

Algorithm 4 Bernstein

```
1: procedure C++ PROCEDURE
2:     if t = 0 and i = 0 then return 1
3:     else t^i
4:     if n = i and t = 1 then return 1
5:     else (1 − t)^(n−i)
6: call to Binomial:
7:     binomial() ← i, n.
8:     return (binomial ∗ ti ∗ tni)
```

Then the next function takes in the algorithm defined earlier for the final calculation of the Bezier curve. It calculates the points on the curve that define its shape, those points will be used later to check how does the curve look like.

Algorithm 5 Bezier

```
1: procedure C++ PROCEDURE
2:     for i1 != cpts do
3:         p[icount] ← 0.0
4:         p[icount+1] ← 0.0
5:         for i != npts do
6:             basis ← Bernstein(npts-1, i, t)
7:             p[icount] ← basis*b[jcount]
8:             p[icount+1] ← basis*b[jcount+1]
9:             jcount ← jcount+2
10:        icount ← icount+2
```

5.2.2 Implementation in ROS

At the beginning, it is defined a Bézier path that takes in an initial and a goal point. Both the initial and goal point are defined inside the main function without getting them from a topic. I need to select beforehand the size of the init and goal vectors. The points in goal will be the number of points that ROS will use for drawing the path. The Bézier node subscribes to map topic and publish the path in the map path topic.

There is a type of function in ROS called "callback function". The function is defined by yourself but it is not called by yourself. The intuitive manner of working with callback functions is passing a pointer to another component of the code that will call to your function whenever it is appropriate. As ROS works connecting different parts of the system between each other, most of the time these functions work as messages handler. A message handler function is defined to subscribe to the message. This function is never called by yourself, in any case, what happens is

that when a message arrives the function is automatically called and the message it is passed to it.

There are three plain functions which goal is to only receive the message and store the content of the pointer. Those functions are: **goalPointCallback, mapCallback and odomCallback**

- goalPointCallback: It receives the goal point in the shape of a "posestamped" message. It contains the position and the orientation of the point where to go.

- mapCallback: It receives the map where to move in the shape of an "occupancy grid" message. This represents a 2-D grid map in which each cell represents the probability of occupancy.

- odomCallback: It receives the current odometry of the robot, in this case the autonomous vehicle. The type of the message is "odometry". This represents an estimate of a position and velocity in free space. The odometry is the use of data from motion sensors to estimate change in position over time.

With the aid of the Bézier function, the **generate path** function creates the waypoints that define the path. There are two arrays that contain the points that will serve as a requirement to generate the trajectory, "input" and "output". Again with the "posestamped" message each of the points generated by the Bézier function is stored one by one and then published in two forms, as points and as a path. The type of message is navigation message, path message. It contains an array of the poses and a header with ID information.

As the Bézier algorithm only outputs the points but not the orientation they have to follow, it has to be calculated afterward.
The approach I have followed to calculate the orientation is as next:

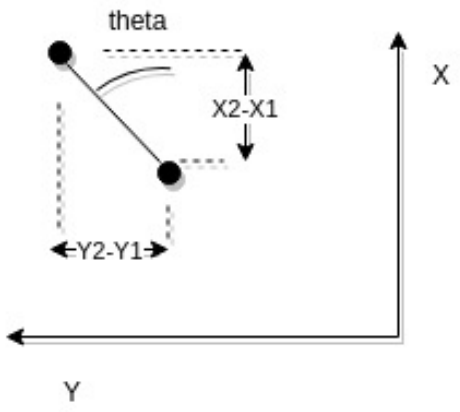

FIGURE 5.2. Orientation calculation.

In the figure above it can be seen two different points. A line between them has been drawn to facilitate the visualization. It is easy to calculate the hypotenuse with a square root:

$$hyp = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2} \tag{5.2}$$

And then apply trigonometry to calculate the angle that line draws with respect to the X axis:

$$\theta = arcos(\frac{X2 - X1}{hyp}) \tag{5.3}$$

Using the aid of the quaternions, each of the orientations is transformed into Quaternions. In our case we are only rotating along the **z axis** because the plane where the vehicle is moving relies on the x and y axis, thus the **x** and **y axis** are null:

$x = 0 * sin(\theta * 0.5)$

$y = 0 * cos(\theta * 0.5)$

$z = 1 * sin(\theta * 0.5)$

$w = cos(\theta * 0.5)$

This way the orientations are drawn as in Figure 6.9

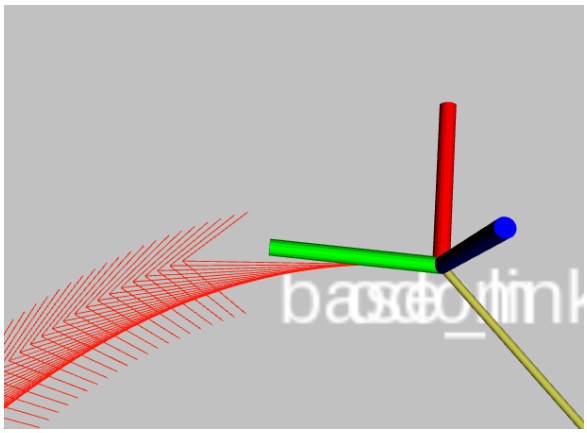

FIGURE 5.3. Orientations in RVIZ.

Each of the arrows represents a line tangent to each point of the path, it should be the orientation of the vehicle at every stage of the trajectory. It must be noted that the vehicle can only take certain positions and orientations as it is an Ackermann [1] steering based robot.

There is also a transform process. The transform is taken from *map* ⇒ *baselink*

[1]The Ackermann model is a steering geometry. This kind of robots cannot turn in place due to its wheeled configuration

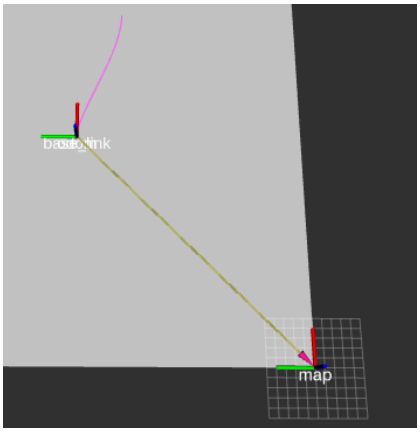

FIGURE 5.4. Transform from map to base link in RVIZ.

So the path starts publishing in base link. The pseudocode that shows the development of the calculations is shown in the next algorithm.

Algorithm 6 Generate path

procedure C++ PROCEDURE
    inputs[] ← current_odom, goal_point
    outputs[] ← Bezier(inputs)
    for i < number_of_points **do**
        pose.pose.position ← outputs
        pose_map.pose.position ← pose.pose.position + transform
        pose_map.pose.orientation.x ← 0
        pose_map.pose.orientation.y ← 0
        pose_map.pose.orientation.z ← $1 * sin(\theta * 0.5)$
        pose_map.pose.orientation.w ← $cos(\theta * 0.5)$
        path_map ← pose_map
    return *path_map*

There is another type of message called "Pose array". It is an array of poses with a header for global reference The behavior is similar to the path but in this case, instead of showing the whole path the vehicle is about to follow, it shows each of the poses that have been calculated, that is why it was so important to calculate the orientations previously. The shape of the code is similar to the one of the paths except that it returns a different type of message.

Algorithm 7 Generate array

procedure C++ PROCEDURE
    inputs[] ← current_odom, goal_point
    outputs[] ← Bezier(inputs)
    for i < number_of_points **do**
        pose.pose.position ← outputs
        pose_array.pose.position ← pose.pose.position + transform
        pose_array.pose.orientation.x ← 0
        pose_array.pose.orientation.y ← 0
        pose_array.pose.orientation.z ← $1 * sin(\theta * 0.5)$
        pose_array.pose.orientation.w ← $cos(\theta * 0.5)$
        path_array ← pose_map
    return $array\_map$

Once the path is defined it is needed to detect obstacles in order to be able to avoid them. Then a correlation between the points in the path and those on the map must be established. With the aid of the map resolution, the square a coordinate belongs to is calculated. It returns the index of the square. Following the figure 5.5 the explanation is easier to follow.

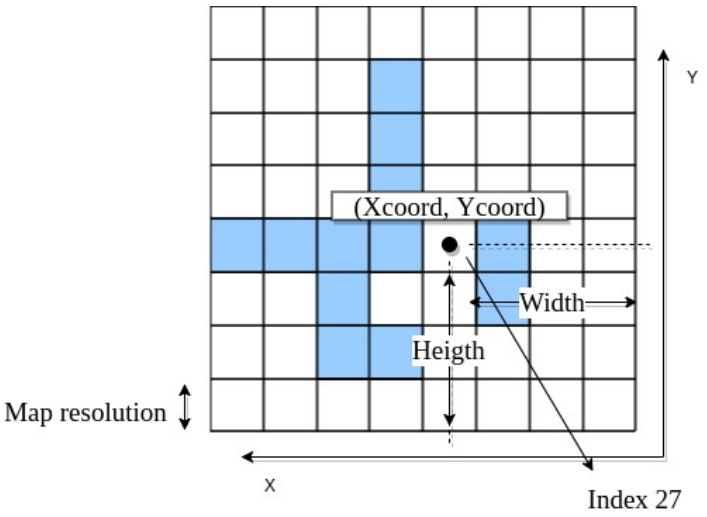

FIGURE 5.5. Parametrization into map grid.

Taking into account the map message is one dimensional, the map grid must be taken as a one dimension array aswell. The sum of all the squares until reaching the point gives the index from the map grid array.

$$height = y\_coordinate/map\_resolution \tag{5.4}$$

$$width = x\_coordinate/map\_resolution \tag{5.5}$$

(5.6) $$index = map\_width * height + width$$

Once the index of a particular point from the path is found, it must be checked if there is an obstacle on it. In case there is, an iteration along the perpendicular from the line that joins the previous and the next point is given. An increment is fixed and then it is summed or subtracted from the perpendicular. In the figure 5.6 the middle point hits an obstacle, thus it is shifted along the perpendicular to the place where no obstacle is found.

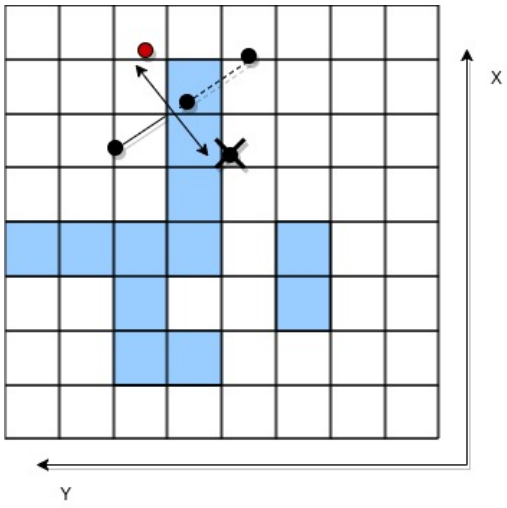

FIGURE 5.6. Obstacle avoidance.

To calculate the localization of the new point it is needed again to obtain the angle θ. The image 5.7 shows where to find that angle in the new situation.

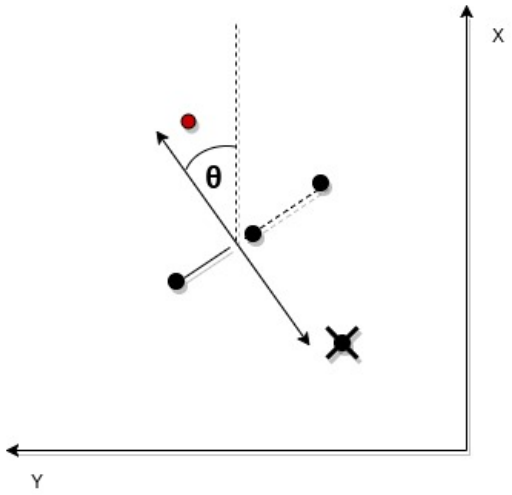

FIGURE 5.7. Obstacle avoidance abstraction.

$$tan(\theta) = \frac{y\_coordinate2 - y\_coordinate1}{x\_coordinate2 - x\_coordinate1} \tag{5.7}$$

$$pose\_x = cos(\theta) * increment + \frac{(x\_coordinate2 - x\_coordinate1)}{2} + x\_coordinate1 \tag{5.8}$$

$$pose\_y = sin(\theta) * increment + \frac{(y\_coordinate2 - y\_coordinate1)}{2} + y\_coordinate1 \tag{5.9}$$

With the angle obtained with the aid of trigonometry it is possible to locate the grid cell where that point lies on. With the use of θ the calculus of the index is trivial. It is needed to take both x and y coordinates from each of the two adjacent points and obtain the mediatrix. Then extend the increment along it in both coordinates. As the number of points that shape the curve is not a fixed quantity, they can be parametrized. That number will control the accuracy of the curve and the computational cost wasted for generating each path. Keep in mind this algorithm is meant to be working on a real time system, thus computational efficiency should be a parameter to care about.

Algorithm 8 Check obstacles

```
1:  procedure C++ PROCEDURE c
2:      increment ← 20
3:      toggle ← 1
4:      for i < number_of_points do
5:          x_coordinate ← trajectory.position.x
6:          y_coordinate ← trajectory.position.y
7:          height ← y_coordinate / map.resoulution
8:          width ← x_coordinate / map.resoulution
9:          index ← map.width * height + width
10:         if map.data[index] ≠ 0 then
11:             x_coordinate_1 ← trajectory[i-1].position.x
12:             x_coordinate_2 ← trajectory[i+1].position.x
13:             y_coordinate_1 ← trajectory[i-1].position.y
14:             y_coordinate_2 ← trajectory[i+1].position.y
15:             tangent ← (y_coordinate_2 - y_coordinate_1) / x_coordinate_2 - x_coordinate_1
16:             θ = arctan(tangent)
17:             do
18:                 aux_pose.position.x ← toggle*cos(theta)*increment + (x_coordinate2 -x_coordinate1)/2 + x
19:                 aux_pose.position.y ← toggle*sin(theta)*increment + (y_coordinate2 -y_coordinate1)/2 + y
20:                 height ← aux_pose.position.y / map.resolution
21:                 width ← aux_pose.position.x / map.resolution
22:                 index ← map.width * height + width
23:                 toggle ← toggle * -1
24:             while map.data[index] ≠ 0
25:         return true
```

Then when a new point that avoids the obstacle has been created, the **generate path** function is called to shape the curve again. Once the proper route has been found, it is required to publish it on a topic. Taking the advantage of a ROS Publisher [2] previously defined with the adequate message type, the message is taken outside the node and published on a topic where the launch file will be subscribed later. As there are three kinds of messages aimed to be published, three different types of publishers must be defined.

The instructions within the **launch file** defines not only the transforms among reference frames, but it also publishes the map contained in the "maps folder" and it orders the launch of RVIZ with the configuration stored in the **global plan** file. Once in RVIZ, the points from the path can be observed in a graphic manner so it is easier for humans to understand its meaning. The path is represented as a unified line, the pose arrays as a set of positions with a defined orientation and the points, as marks that shape the route.

[2] A Publisher is a tool from ROS that allows publishing a message from a Node to a Topic. The type of the publisher requires to be defined before it gets the message to publish

CHAPTER 6

TESTING AND RESULTS

In this section the results from the design described in Chapter 5 are discussed. All the tests are presented from RVIZ and they try to bring clarity to the explanations showed along this project. In some cases, a schematic figure has been included to represent what is aimed to see in RVIZ. The different points that make up the trajectories have been specified next to each graph as the reference frame.

6.1 Bézier curves in ROS

The main issue when drawing Bézier curves in ROS was transforming the points into ROS messages. As a test a simple curve has been drawn from the origin reference frame, in this case, the "map" reference frame. The curve is defined by 2048 points, the point in the origin is (0,0) and the goal point is variable. As it is shown in figure 6.1 the goal point orientation is perpendicular to the horizontal thus, the orientation of the vehicle stays the same in Figure 6.1.

6.1.1 Transform the curve from map to odom

With the aid of the transform package from ROS, the curve with respect to the map reference frame has been transformed to the odom reference frame. Not the positions nor the orientations have been changed from the code. The transform matrix has been added to the previous position. The number of points that defines the curve as the goal positions and orientations are the same as in section 6.1. To see it more clearly the scheme in figure 6.3

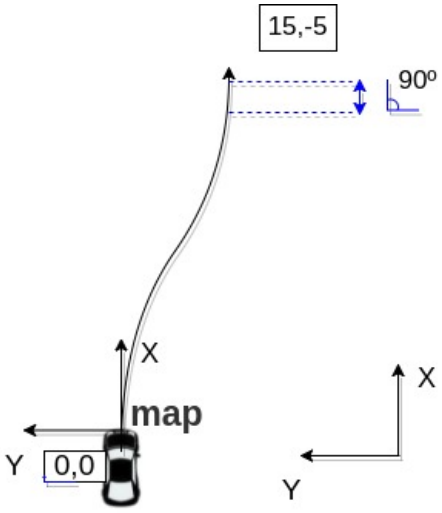

Figure 6.1: Bezier curve from map.

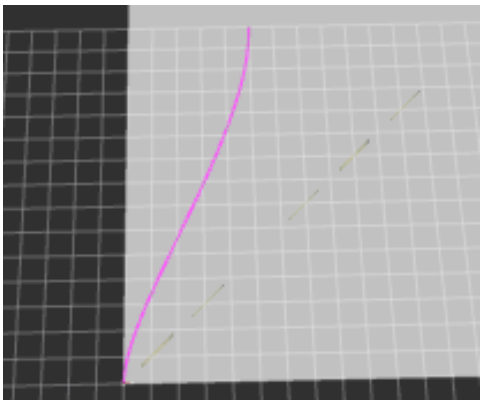

Figure 6.2: Bezier curve in RVIZ

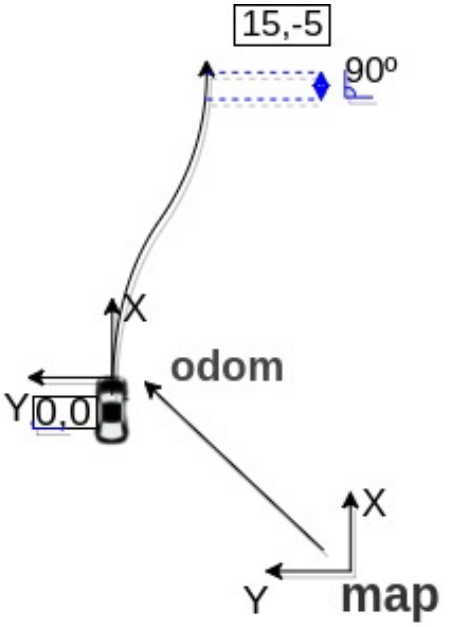

Figure 6.3: Bezier transform to odom.

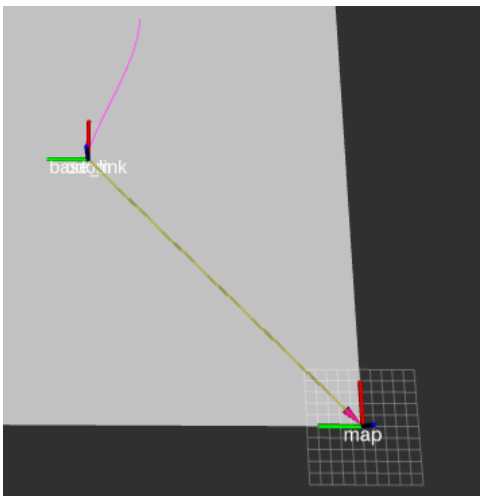

Figure 6.4: Bezier transform in RVIZ

6.1.2 Behavior of the routes depending on different orientations and positions

In the figure 6.5 a set of routes are disposed to explain how different orientations at the goal point shape the whole path. On the left side, three different paths with different goal points are shown from the RVIZ view. As those paths are composed of points with different positions and orientations, a zoom to the origin is disposed in the right side of the images, figure 6.6 where each

of the points can be noted.

Figure 6.5: Set of different paths with different orientations.

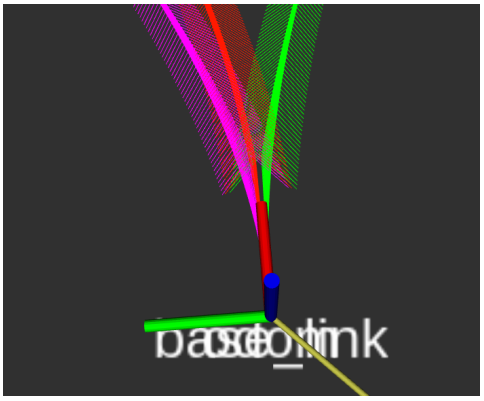

Figure 6.6: Set of poses from each path. This view comes from a "PoseArray" message.

The next set of images are a sample of how the path observed from RVIZ is modeled. To note the tendency of the curve different parameters, the initial and target position and the initial and target orientation have been tuned as showed in Table 6.1.

Group	Initial orientation	Final orientation	Final position
A (pink)	90º	90º	(20, 15)
A (blue)	90º	90º	(10, 5)
A (green)	90º	90º	(20, -10)
B (pink)	90º	45º	(20, 15)
B (blue)	90º	45º	(10, 5)
B (green)	90º	45º	(20, -10)
C (pink)	135º	45º	(20, 15)
C (blue)	135º	45º	(10, 5)
C (green)	135º	45º	(20, -10)

Table 6.1: Parameters from each curve

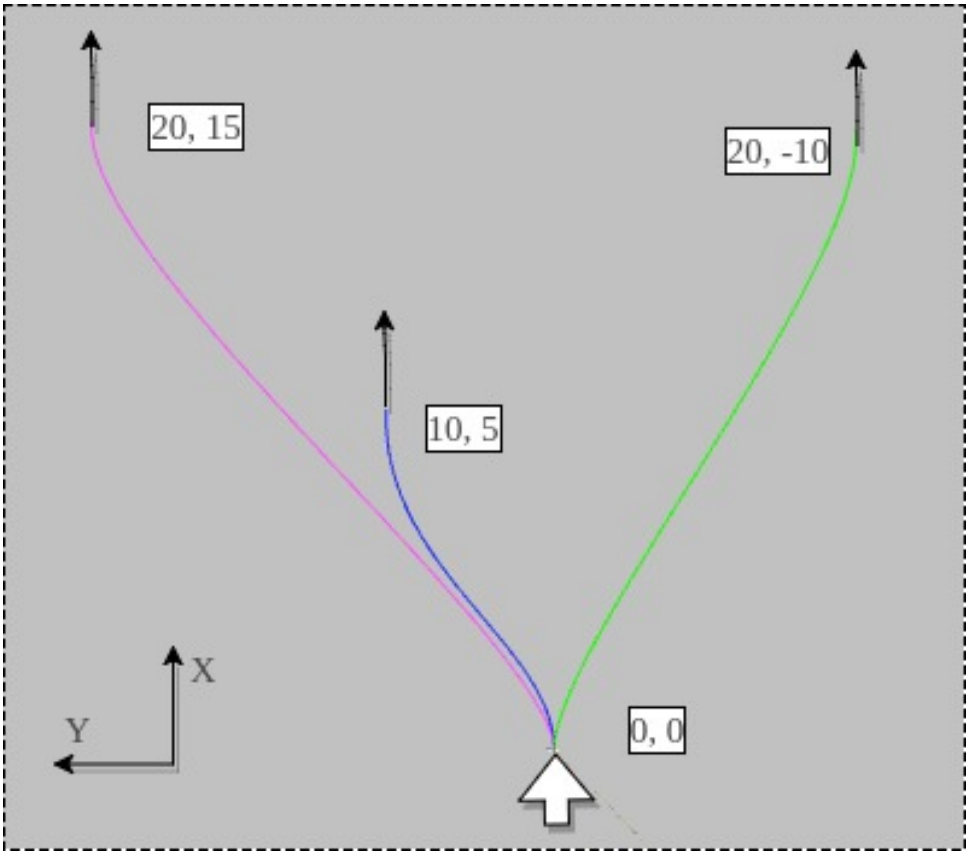

FIGURE 6.7. Group A curves in RVIZ.

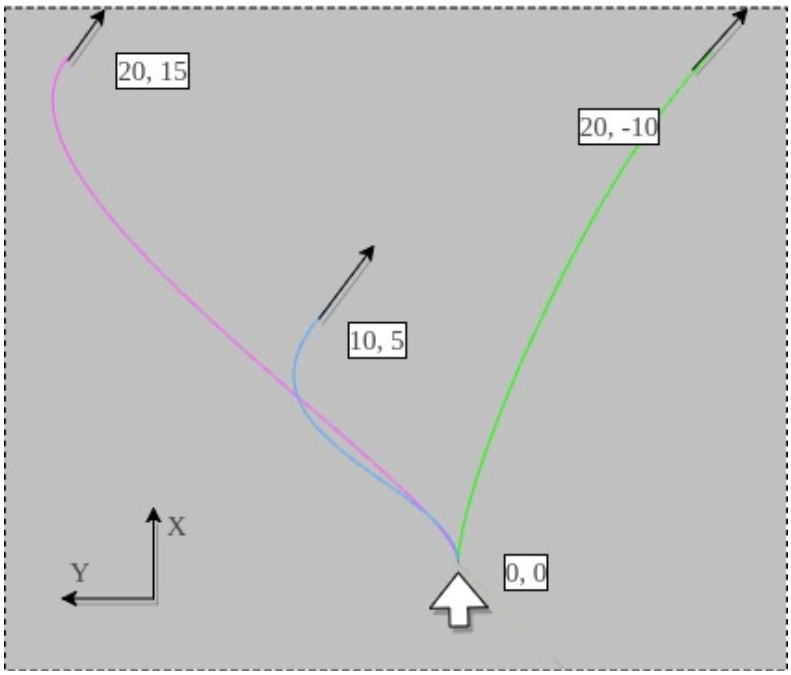

FIGURE 6.8. Group B curves in RVIZ.

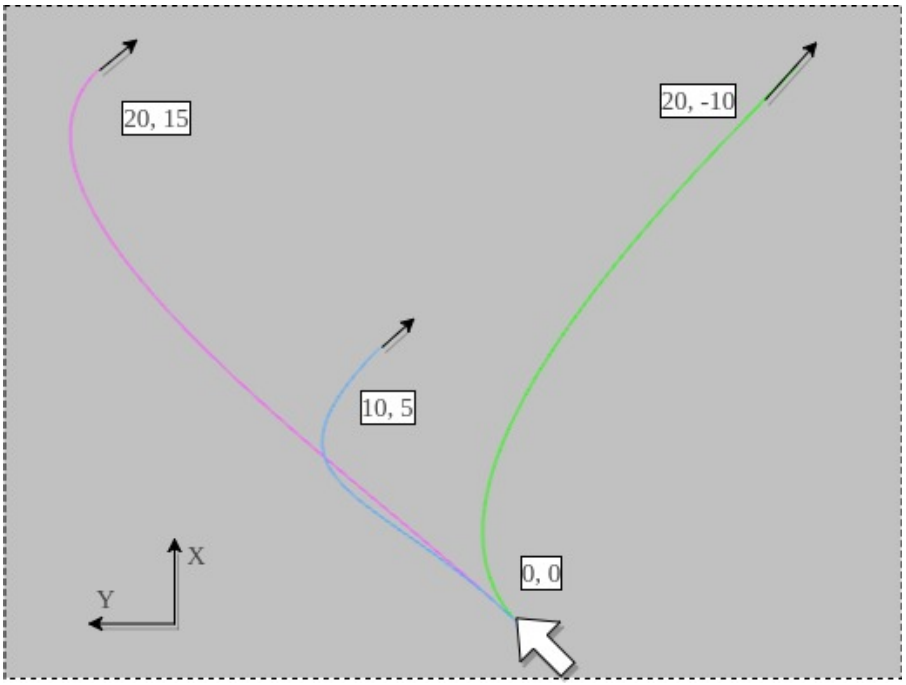

FIGURE 6.9. Group C curves in RVIZ.

6.2 Avoiding obstacles in ROS

Different dispositions of obstacles within the map are received by the node. It analyzes them and provides a solution. In the first image the car should turn right to avoid the wall. Note the car always begins with an orientation parallel to the horizon. The algorithm begins scanning the map in the x coordinate (upwards with respect to image 6.10) as if the map consists of a one-dimensional array as explained in Chapter 5. It parametrizes each of the positions of the path generated to compare them with the map grid. When the "Obstacles Detection" function bumps into an obstacle pixel, it strives to find the first hole within the perpendicular to the obstacle, first in the left sense and then in the right one. Once it finds an empty hole, that point is taken as an initial parameter requirement and the path is recalculated. The loop keeps going until the function does not complain about obstacles anymore.

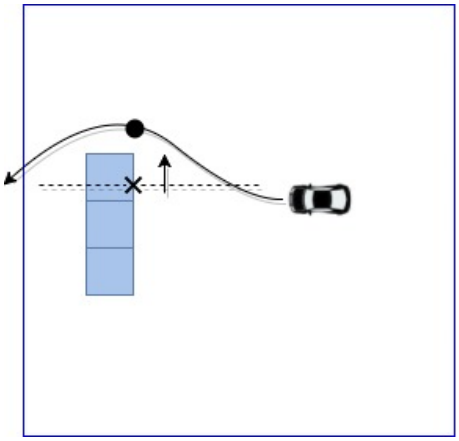

Figure 6.10: One obstacle avoidance abstraction

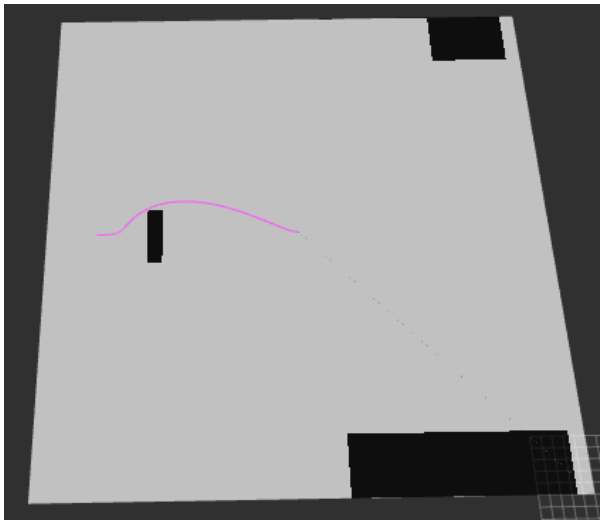

Figure 6.11: One obstacle avoidance in RVIZ

One of the complications that may arise working with the avoidance algorithm is that the remaining holes that are not treated, get stuck at the end of the path forcing the curve to adopt a non-smooth shape in the final section. The vehicle, in this case the iCab [1] has a fixed minimum radius which limits the path to only work with smooth ways. Furthermore, there should be a tolerated range around the obstacles because the iCab cannot be treated as a line-wide object. As it can be appreciated in Figure 6.11 the car drives very close to the top obstacle corner. Depending on the dimensions of the vehicle it may cause a collision with the obstacle even if the path does not look to do so.

As a more complex proposal, a map in which the road is blocked by two obstacles is analyzed. In this case, it is provided two extra empty points, let's call them holes which memory has to

[1]Vehicle from Universidad Carlos III de Madrid used for studies in the field of Autonomous Driving

be allocated. Once the location of the obstacle has been found, the algorithm starts to iterate positions until obstacles are not detected any longer. The tests can be appreciated in Figure 6.13.

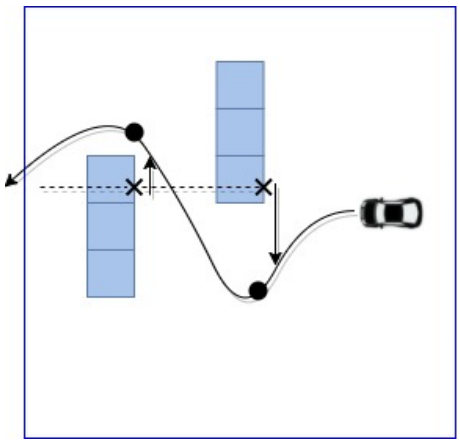

Figure 6.12: Two obstacle avoidance abstraction

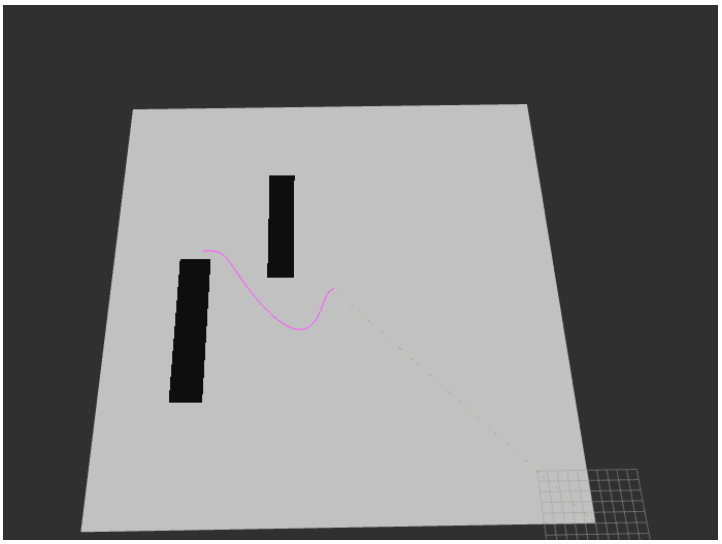

Figure 6.13: Two obstacle avoidance in RVIZ

As in this last escenario one more hole is occupied by the second point, the curve softens in the final stretch allowing the vehicle to pass by that section with fewer difficulties. Again the dimensions of the car should be taken into account in later studies as it can not be concluded if it touches the obstacles at this level.

CHAPTER 7

CONCLUSIONS AND FURTHER IMPROVEMENT

The goal of this dissertation was to design a local path planner in order to make autonomous driving more secure and flexible using Bézier curves. In this chapter first the "Contributions" sections gathers all the achievements of this project. Then some conclusions the author has learned have been summarized. In the end some ideas for further improvement have been suggested.

7.1 Contributions summary

The main content of this project describes all the contributions to the Autonomous Driving field (Chapters 4 5). The bullet points have been summarized in the next points to have a quick view of it:

- A Bézier curves algorithm written in C++ has been integrated into the ROS environment. It is able to process the goal point and the current Odometry from the robot in this case, an Ackerman robot and provide a path defined by a number of points. That number can be modified to change the accuracy and improve the processing capacity of the system. It takes into account not only the localization but the orientation from the current and local position aswell to create a path that fits into a vehicle.

- A ROS node has been created. It adapts the Bezier algorithm to convert the positions and orientations that it returns into points, poses and paths. It is capable of transforming each of the points of the path into any of the reference frames from the components of the robot. It also receives data from other topics to define the "init" and "goal point".

- A working prototype for avoiding obstacles has been designed and has been tested in a number of maps with different blocks. It follows the guidelines proposed in the objectives section. A tool for detecting obstacles and correlate the coordinates from a physical system with map grid cells have been implemented. It locates the obstacle and proposes a solution as an alternative path. Then it checks if that alternative path is feasible, otherwise it generates another iteration and repeats the process.

7.2 Conclusions

- Designing an Engineering System is a complex process that depends upon a big set of variables and tools. Those variables provide a huge amount of data that is difficult to select and track. The libraries and tools provided by ROS turn out to be a very useful solution to control the flow of data and select which of the sources are effective to our project.

- Several methods have been studied both for local and global Path Planning. There is more than a solution when it comes to developing a local planner and a collision avoidance system. Despite in this thesis the algorithm chosen has been Bézier, there is already a lot of work established with other planners as explained in Chapter 3.

- To coordinate a system there must be an agreement about the type of data sent and the flow rate of it. When connecting different nodes between each other there must be noted there may be time delays between the data emitted and the data received.

7.3 Future work

The design of a full path planner system can be hardly completed within the span of a Bachelor's Thesis. Thus the dissertation has been focused on designing a prototype on which to elaborate a later study to, for example, implement the system into the working Autonomous System iCab from Universidad Carlos III de Madrid. A number of ideas have emerged during the elaboration of the thesis but they haven't been further advanced:

- Implement the planner in a real-time environment so the node is continuously receiving data from the topics and it publishes a new path at a certain rate.

- In the real-time system mentioned above: Deploy an obstacle detection process where the map is being updated and new obstacles show up without being expected.

- Test the path planner in a simulator such as the one from Udacity Self Driving Car nanodegree where a particular vehicle must bear in mind the localization from the other vehicles and decide if, depending on the path generated, it is possible to overtake them without collision

- Split the node created during this project into different nodes. Each of them should provide a different type of path and receive different classes of data.

APPENDIX A

APPENDIX: THE SOURCE CODE

Some of the files were too big to be attached to this document. That is the reason why a GitHub repository has been created. The following link redirects to the repository: Click here

BIBLIOGRAPHY

[1] S. A. Fadzli, S. I. Abdulkadir, M. Makhtar, and A. A. Jamal, "Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries," in *Information Science and Security (ICISS), 2015 2nd International Conference on*. IEEE, 2015, pp. 1–4.

[2] X. Ji, L. Liu, P. Zhao, and D. Wang, "A-star algorithm based on-demand routing protocol for hierarchical leo/meo satellite networks," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1545–1549.

[3] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[4] J. hwan Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 188–193.

[5] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Bessler, and B. Huhnke, "Up to the limits: Autonomous audi tts," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 541–547.

[6] D. J. Shiach. Bezier cruves. [Online]. Available: https://www.youtube.com/watch?v=2HvH9cmHbG4

[7] R. T. Farouki, *Pythagorean—hodograph Curves*. Springer, 2008.

[8] L. Dormehl and S. Edelstein. Sit back, relax, and enjoy a ride through the history of self-driving cars. [Online]. Available: https://www.digitaltrends.com/cars/history-of-self-driving-cars-milestones/

[9] J. U. The road to driverless cars: 1925 - 2025. [Online]. Available: https://www.engineering.com/DesignerEdge/DesignerEdgeArticles/ArticleID/12665/The-Road-to-Driverless-Cars-1925--2025.aspx

[10] E. Benedito, A. Doria-Cerezo, C. Kunusch, and J. M. Olm, "Traffic flow-oriented design and analysis of an adaptive cruise control system," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–5.

[11] J. Claybrook, "Safety: The view from nhtsa: New legislation promises to make cars safer, cheaper to repair, and less polluting during the 1980s," *IEEE spectrum*, vol. 14, no. 11, pp. 87–88, 1977.

[12] L. Wos, F. Pereira, R. Hong, R. S. Boyer, J. S. Moore, W. Bledsoe, L. Henschen, B. G. Buchanan, G. Wrightson, and C. Green, "An overview of automated reasoning and related fields," *Journal of Automated Reasoning*, vol. 1, no. 1, pp. 5–48, 1985.

[13] Stanley, el vehículo sin conductor de la universidad de stanford, gana la carrera de 'autos-robot'. [Online]. Available: https://elpais.com/tecnologia/2005/10/10/actualidad/1128932879_850215.html

[14] Road traffic injuries. [Online]. Available: http://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

[15] Motor vehicle crash deaths. [Online]. Available: https://www.cdc.gov/vitalsigns/motor-vehicle-safety/infographic.html

[16] R. Future. España, pionera en la regulación de los vehículos autónomos. [Online]. Available: https://future.inese.es/espana-pionera-en-la-regulacion-de-los-vehiculos-autonomos/

[17] N. López. (2017) Legislación sobre el coche autónomo en españa: queda mucho por hacer. [Online]. Available: https://www.autobild.es/reportajes/legislacion-coche-autonomo-espana-queda-mucho-hacer-179660

[18] L. Thomas and J. Irish. (2017) Germany adopts self-driving vehicles law. [Online]. Available: https://www.reuters.com/article/us-germany-autos-self-driving/germany-adopts-self-driving-vehicles-law-idUSKBN1881HY

[19] Automated and autonomous driving. legal framework. [Online]. Available: https://www.daimler.com/innovation/case/autonomous/legal-framework.html

[20] O. Bowcott. Laws for safe use of driverless cars to be ready by 2021. [Online]. Available: https://www.theguardian.com/law/2017/dec/14/laws-safe-use-driverless-cars-ready-2021-law-commission

[21] N. Lomas. (2018) Uk kicks off driverless car law review to get tech on the road by 2021. [Online]. Available: https://techcrunch.com/2018/03/07/uk-kicks-off-driverless-car-law-review-to-get-tech-on-the-road-by-2021/?guccounter=1

[22] M. Mittereder. Automated vehicles: Legal framework will decide which automakers win the race to full automation. [Online]. Available: https://www.rolandberger.com/en/Media/Automated-Vehicles-Legal-Framework-Will-Decide-Which-Automakers-Win-the-Race-to.html?country=null

[23] T. Peng. (2018) Global survey of autonomous vehicle regulations. [Online]. Available: https://medium.com/syncedreview/global-survey-of-autonomous-vehicle-regulations-6b8608f205f9

[24] Á. Melgosa Pascual, "Uav planning, autonomous tracking, and obstacle identification and avoidance," 2016.

[25] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[26] É. Simonin and J. Diard, "Bbprm: A behavior-based probabilistic roadmap method," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*. IEEE, 2008, pp. 1719–1724.

[27] S. K. Gehrig and F. J. Stein, "Collision avoidance using elastic bands for an autonomous car."

[28] Ros introduction. [Online]. Available: https://blog.robotiq.com/

[29] Is ros for me? [Online]. Available: http://www.ros.org/is-ros-for-me/

[30] Definition of flowchart. [Online]. Available: https://www.merriam-webster.com/dictionary/flowchart?utm_campaign=sd&utm_medium=serp&utm_source=jsonld

[31] Coordinate frames, transforms, and tf. [Online]. Available: http://wiki.ros.org/tf/Overview/Transformations

[32] D. Kurzaj. Ros messages. [Online]. Available: http://wiki.ros.org/msg

[33] T. Foote. Geometry messages. [Online]. Available: http://wiki.ros.org/geometry_msgs

[34] P. Ranger, "Comments, with reply, on'structured flowcharts outperform pseudocode: an experimental comparison'by da scanlan," *IEEE Software*, vol. 7, no. 2, p. 6, 1990.

[35] Bézier curves and kronecker's tensor product. [Online]. Available: https://blogs.mathworks.com/images/graphics/bezierpost.html

[36] About ros. [Online]. Available: http://www.ros.org/about-ros/

[37] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1879–1884.

[38] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2061–2067.

[39] R. W. Farebrother, J. Groß, and S.-O. Troschke, "Matrix representation of quaternions," *Linear algebra and its applications*, vol. 362, pp. 251–255, 2003.

[40] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[41] S. Upadhyay and A. Ratnoo, "Continuous-curvature path planning with obstacle avoidance using four parameter logistic curves," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 609–616, 2016.

[42] Q. Tan and J. G. Balchen, "General quaternion transformation representation for robotic application," in *Systems, Man and Cybernetics, 1993.'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on*, vol. 3. IEEE, 1993, pp. 319–324.

[43] D. Tromba, L. Munteanu, V. Schneider, and F. Holzapfel, "Approach trajectory generation using bezier curves," in *Aerospace Electronics and Remote Sensing Technology (ICARES), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–6.

[44] Y. Zhang, H. Xiao, D. Chen, S. Ding, and P. Chen, "Link from zd control to pascal's triangle illustrated via multiple-integrator systems," in *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on*. IEEE, 2016, pp. 2127–2131.

[45] [Online]. Available: https://www.youtube.com/watch?v=2HvH9cmHbG4

[46] N. Montes, M. C. Mora, and J. Tornero, "Trajectory generation based on rational bezier curves as clothoids," in *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE, 2007, pp. 505–510.

[47] J. Z. Vidaković, M. P. Lazarević, V. M. Kvrgić, Z. Z. Dančuo, and G. Z. Ferenc, "Advanced quaternion forward kinematics algorithm including overview of different methods for robot kinematics," *FME Transactions*, vol. 42, no. 3, pp. 189–199, 2014.