Treball de Fi de Màster

Master's degree in Robotics and Control Systems

Study of path following algorithms for LIDAR obstacle detection and collision avoidance

MEMÒRIA

Autor:	Noelia Llamazares Álvarez
Director:	Bernardo Morcego Seix
Convocatòria:	June 2018

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

This master thesis is aimed at developing a path following algorithm compatible obstacle detection and collision avoidance for an AGV (autonomous ground vehicle) configured with different hardware elements using 2D Lidar sensors. This hardware configuration will be taken into account as part of a future work of the thesis.

Therefore, the thesis has a technological viewpoint, in which LIDAR signals have to be processed and analyzed in order to determine where is located the free space; and a theoretical viewpoint, the study of the more appropriate path planner and its escape and obstacle avoidance part when a collision probability is detected.

The main objective of this study is to implement into the AGV the more suitable path planning algorithm. It will be discussed through the study of different path planner algorithms and the development of the obstacle avoidance analysis to find the appropriate method to apply in.

The project is focused on how LIDAR sensors work in this kind of robots. This work is developed paying attention to how they work and sample the data. Software development operates using ROS simulation through Matlab platform.

In addition, some improvements will be presented according to the first simulation results. Once they have been implemented, the final results will be discussed. In this part, the behaviour and the robot finally obtained performance will be evaluated as a way to summarize the strengths and the weak points of the thesis.

Contents

ETSEIB

FIGURE LIST

TABLE LIST

1. Introduction

In recent years we have experimented how technology has evolved exponentially and how these changes are adapted to our daily life.

In this way, the competencies of AGV systems have hit the roof as software and sensor technology has improved. Nowadays, companies are able to provide more accurate, safer and efficient vehicles than ever before, at the same time that as years passed more promising will be these results, including the AGV industry.

In the next three years it is expected to show off cars able of navigating along city streets at adaptive speeds along predetermined routes fixed before. Nowadays, most of vehicle companies that are increasing their efforts in this field can already handle basic driving at low speeds. Despite of that, the most difficult part is to get people approval after the appearance of some accidents after pilot testing phase. For this reason, it is expected to develop systems able to handle all situations even when suddenly situations that require immediate reaction appear.

When we talk about autonomous ground vehicles, one of the most relevant and challenging point is the confidence level of its navigation trajectory. This means that we expect that the AGV will be able to reach a destination goal from an initial point every time avoiding collisions to each different kind of obstacles that can appear around the vehicle itself.

From the computed planned path defined by a set of waypoints and the navigation mode of operation, the AGV should move from the initial node to the desired goal node with a determined direction and speed.

Along the navigation performance from one waypoint to the next one, the data collected by the sensor is continually analyzed in such a way to define if the path is navigable, or on the contrary it detects an obstacle and therefore, an avoiding action must be performed to continue the navigation in a safe way to evade the obstacles found.

This thesis is focused on the computation of the navigation commands through the right selection of a path planner at the same time collected data from the sensors is filtered and analyzed to estimate the obstacles around and their distance to.

To solve the obstacle avoidance problem there exist many different algorithmic solutions, which most of them implement strategies to detect 2D obstacles. In this master thesis, a 2D obstacle avoidance planner was determined, implemented and simulated supposing an autonomous ground vehicle equipped with a 180 degree 2D Lidar sensor to obtain the data of the environment and then, detect the possible obstacles that can appear around to be able of designing a free collision trajectory.

An AGV with a LiDAR sensor can easily measure the distance range between objects and the vehicle and its orientation through the set of laser pulses that Lidar sensor is transmitting. All this data is compiled, and as a result, creates an environmental map of the operational area which makes feasible the navigation of the AGV without any additional infrastructure.

One of the most relevant reasons of using Lidar sensors, it is that they provide more flexible systems. Despite of being around for last years, the cost of this kind of sensors is dropping, which means that they are more accessible and more and more companies are including them into their projects.

This document summarizes our capabilities of conducting this research. *Chapter 2* states the objectives, framing the reach of our study. Related work is analyzed in *Chapter 3*, our proposed software development is contained in *Chapter 4* and C*hapter 5* describes some enhancements to take into account. A recompilation of experiments along the previous study conditions with the improvement proposed are presented in *Chapter 6*. Final results and conclusions are explained in the last two chapters.

2. Objectives

To obtain the final objective of this work it is necessary to point all the steps and tasks required to develop it. These stages can be classified in the following ones:

- Understand which is the initial point and goal of the problem.
- Study, determine and implement the suitable AGV dynamics model.
- Analysis of different path planning algorithms: its structure, behavior and weaknesses. Determine the right path planner algorithm that can fit with the characteristics of the master thesis problem definition.
- Study of diverse obstacle avoidance techniques: how they are implemented, how they work and under which constraints they experiment bad results. Select the one that can fit with the problem case study.
- Get information about how LIDAR sensors work and how they transform the evidence into data. Be able of reading, filtering and analyzing the data collected by the Lidar sensors to build an occupancy map from the obtained samples where each one has the information related to the AGV distance and position.
- According the data processed from LIDAR sensors, be able to determine and detect the obstacles found around the autonomous ground vehicle in the different maps treated and estimate the probability of collision it has.
- Implement all the information obtained before in the path planner algorithm to calculate the collision free navigation routes for the autonomous ground vehicle presented and in case of obstacle be able to compute a new obstacle free trajectory.
- Simulate the results in different environments as evidence that works under blocked path situations to extensively validate the thesis project.
- State the performance obtained, the problems and situations experimented, and the improvements implemented to obtain the final result.

3. Related work study

To develop this master thesis, it is necessary to understand and to study the theoretical part of the algorithms and systems involved in path following and collision avoidance techniques. Also, they must be taken into account the simulated AGV dynamics model and how the data is obtained from the environment.

For this reason, *Chapter 3* is divided in four main sections in order to explain in detail how the project works from the theoretical point of view.

One of the points of this thesis is LIDAR system application. According to that, it has been considered important to explain how these kind of sensors works. *Section 3.1* collects the main characteristics of this system and how it processes the data from the environment.

To continue, *Section 3.2* contains the required equations to obtain the dynamics model for the autonomous ground vehicle according the characteristics considered.

Finally, the two last sections allude to the detailed analysis of both collision avoidance and path planning algorithms. *Section 3.3* is about the development of the path following algorithm through the idea of the results we want to obtain. Also, it considers the evolution of the algorithm basis till the selected one is found. In Addition, the obstacle avoidance techniques are analysed in *Section 3.4.* Following the steps considered in *Section 3.3,* this section studies the basis of obstacle avoidance algorithms and how their performance have evolved in order to obtain the final algorithm used in this project.

3.1. LIDAR System

As it is detailed in *Chapter* 7, the project has been developed taking as a reference real components. For this reason, the AGV is supposed to count with the A1M8 laser scanner model developed by RoboPeak. The system is omnidirectional, which means that it can perform the scan within 6-meter range distance and an angular resolution of one degree.

LIDAR's scanning frequency reaches 5.5 hz when sampling 360 points each round. And it can be configured up to 10 hz maximum. It contains a range scanner system and a motor system. Once subsystems are powered, Lidar starts rotating and scanning clockwise. In this way the user can get range scan data through the communication interface (Serial port/USB). *Figure 1* summarizes the main LIDAR components.

This Lidar model comes with a speed detection and adaptive system. The frequency of laser scanner is adjusted automatically according the motor speed.

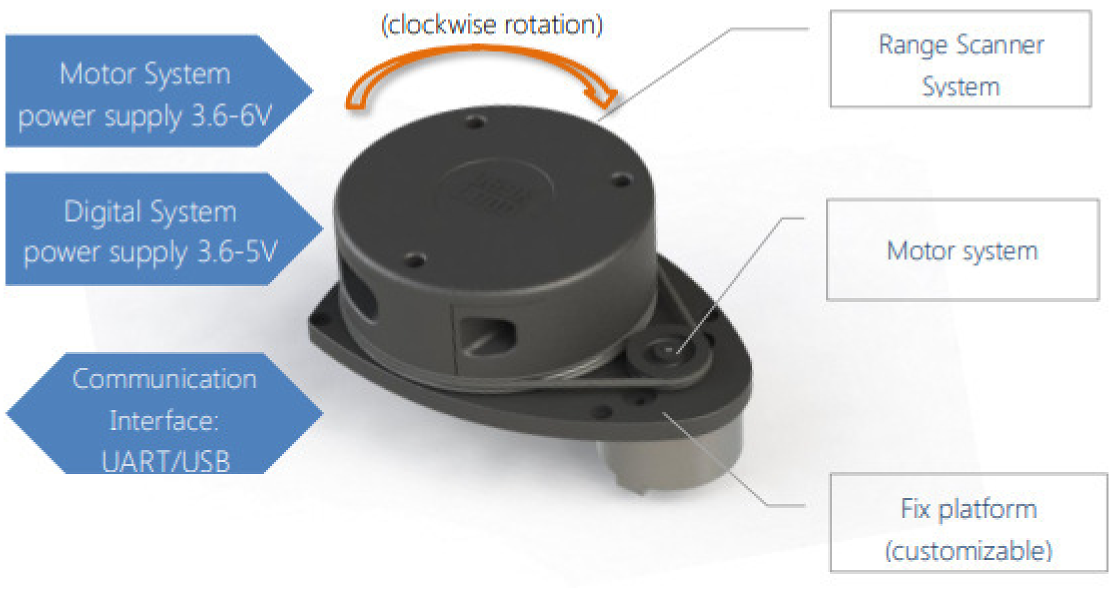

Figure 1*. Lidar system connection*

Its working is based on laser triangulation ranging principle and uses high-speed vision acquisition and processing hardware. The sensor measures with high resolution distance output (less than 1% of the distance) the distance data more than 2000 times per second. In this way, it emits modulated infrared laser signal. Then, the signal is reflected by the object to be detected. *Figure 2* represents the triangulation measurement principle.

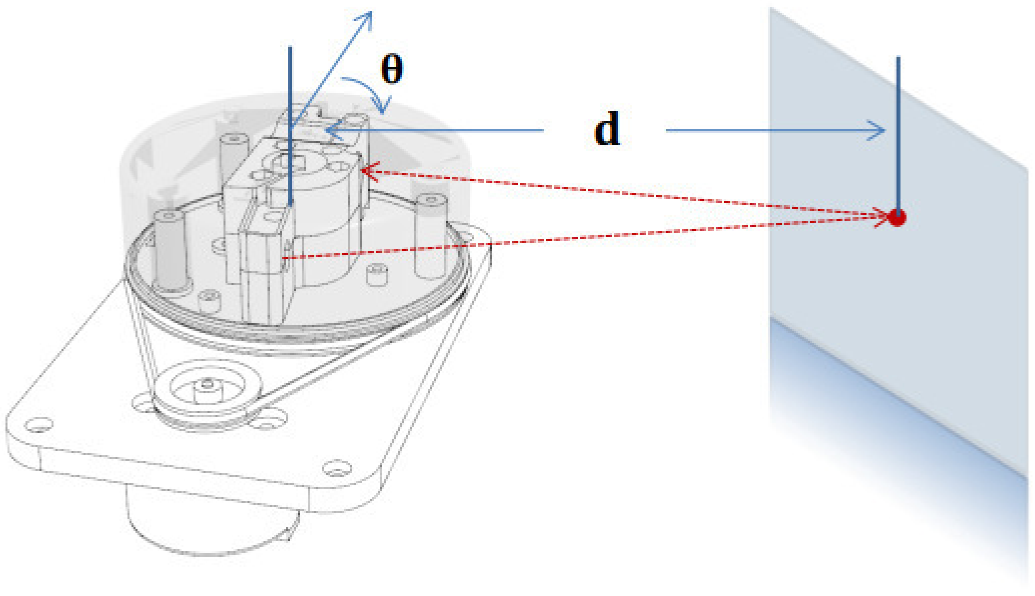

Figure 2. *Laser triangulation measurement on Lidar systems.*

The high-speed ranging scanner system is set up on a spinning rotator with a build-in angular encoding system. During rotating, a 360-degree scan of the current environment will be performed.

Finally, the returning signal is sampled by vision acquisition system and the DSP embedded in the Lidar sensor starts processing the sample data obtained from the scanning period. Through the communication interface the output data obtained is the distance value and the angle between the Lidar and the obstacle detected.

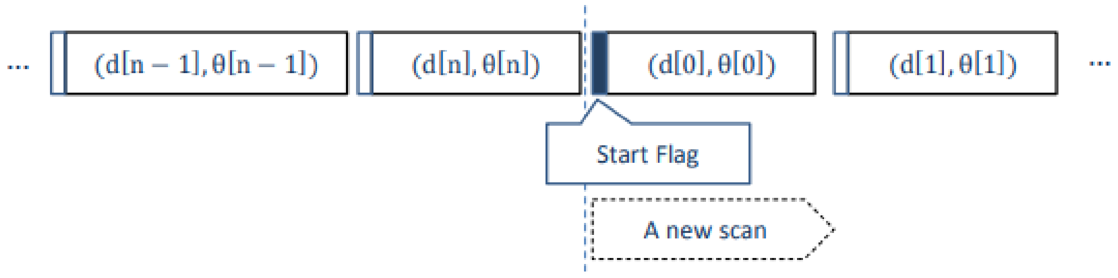

Figure 3. *Output data from Lidar sensor.*

Once the output data is obtained, we need to separate it in two different vectors: orientation angle and distance. Furthermore, we need to apply a filtering method to discard that pair of samples that have been affected by noise or those that are a product of an erroneous measurement. [1]

After computing the operations mentioned before, we can extrapolate the results as a cloud of points to plot them in a map. As the sensor does not estimate the height of the samples, the resulted map obtained will be a 2D graph, being the distance and the angle the orientation and distance value of the position between the obstacle and the robot.

In *Chapter 6*: *Final Results,* will be exposed the figures corresponding with LIDAR sampling and occupancy map resulting from simulation part.

3.2. AGV Dynamics model

To define the kinematics used in this project, the car is considered as a rigid body that is moving along the plane.

According to that, C-space is defined by $C = \mathbb{R}^2 \times \mathbb{S}$. $\mathbb{R}^2$ means *x* and *y*, which refer to the 2D car position coordinates. $\mathbb{S}$ denotes the third term that describes the state of the robot: the orientation.

A configuration is denoted by *q = (x , y, θ)*. The body frame of the car places the origin at the centre of rear axle, and the x-axis points along the longitudinal axis of the car. Let ϕ represent the steering angle, which gets a negative value in the position represented in *Figure 4*. If ϕ is fixed, this steering angle will cause a car circular movement where *ρ* represents the radius of a circle that is traversed by the centre of the rear-axis. L denotes the distance between the front and the rear axles. Let *s* represent the speed of the car (velocity in the x direction of the body frame). [2]

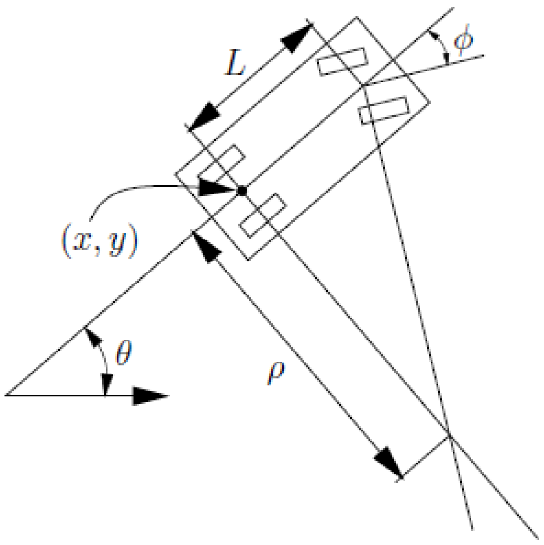

Figure 4. *AGV representation*

According to the notation presented, the objective is to constitute the car motion as a set of equations of the form:

$$\dot{x} = f_1(x, y, \theta, s, \phi) \qquad (3.1)$$

$$\dot{y} = f_2(x, y, \theta, s, \phi) \qquad (3.2)$$

$$\dot{\theta} = f_3(x, y, \theta, s, \phi) \qquad (3.3)$$

In a short time period, Δt, the aim is that the car must move approximately in the direction that the rear wheels are pointing.
In the limit as Δt tends to zero, this implies that $dy/dx = \tan\theta$, which consequently $dy/dx = \dot{y}/\dot{x}$ and $\tan\theta = \sin\theta \,/\, \cos\theta$.
This condition can be written as a Pfaffian constraint as:

$$-\dot{x}\sin\ \theta + \dot{y}\,cos\,\theta = 0 \qquad (3.4)$$

This constraint is always satisfied if $\dot{x} = cos\theta$ and $\dot{y} = sin\,\theta$. Moreover, any scalar multiple of this solution is also a solution where the scaling factor is directly related to the speed *s* of the car. In this way, the two first scalar components of the configuration transition equation are $\dot{x} = \mathrm{s}\cos\theta$ and $\dot{y} = s\,sin\,\theta$.

The next step is to derive the equation for $\dot{\theta}$. Let ω indicate the distance travelled by the car (the integral of speed). Note that dω = ρ $d\theta$ where, according to trigonometry $\rho = L\,/\,tan\,\phi$ involves *Equation 3.5:*

$$d\theta = \frac{tan\phi}{L}d\omega \quad (3.5)$$

Using the fact that $\dot{\omega} = s$ and dividing both sides by dt, *Equation 3.6* is obtained:

$$\dot{\theta} = \frac{s}{L}tan\phi \qquad (3.6)$$

According the previous equations, the motion of the car has been modelled at the same time no action variables have been specified. Let us suppose that the speed s and steering angle ϕ are directly specified by the action variables u_s and u_ϕ , respectively.

The convention of using u variable with the old variable name as a sub-index helps identify the actions in a configuration transition equation. As follows, a two-dimensional action vector $u = (\ u_s\,, u_\phi\,)$ is obtained. In this way, the configuration transition equation for the project car is:

$$\dot{x} = u_s\cos\theta \qquad (3.7)$$

$$\dot{y} = u_s \sin\theta \qquad (3.8)$$

$$\dot{\theta} = \frac{u_s}{L} tan\, u_\phi \quad (3.9)$$

To finish the transition equation we must specify *U* as the set of actions of the form $u = (\ u_s\, , u_\phi\ )$. First suppose that any $u_s \ \epsilon\, \mathbb{R}$ is possible. The interval $[-\pi\,/2, \pi\,/2]$ is sufficiently large for the steering angle u- because any other value is equivalent to one between $-\pi/2$ and $\pi\,/2$. Steering angles of $\pi\,/2$ and $-\,\pi\,/2$ are problematic.

To derive the expressions for $\dot{x}$ and $\dot{y}$, it was assumed that the car moves in the direction that the rear wheels are pointing. If the front wheel is perpendicular to the rear wheels (assigning $u_\phi\ = \pi\,/2$), it means that $\dot{x}\ = \dot{y}\ = 0$ because the center of the rear axle does not translate. This behavior is not allowed for a standard automobile. A car with rear-wheel drive would probably skid the front wheels across the pavement. Therefore, the car should have a maximum steering angle $\phi_{max} < \pi\,/2,$ and $|\phi| \leq \phi_{max.}$ As it is represented in *Figure 4*, a maximum steering angle implies a minimum turning radius $\rho_{min} = L/tan\ \phi\ _{max}$.

Returning back to the speed u_s it seems that the maximum steering angle should reduce at higher speeds. To make the steering angle vary continuously over the time, let u_ω be an action that represents the velocity of the steering angle: $\dot{\phi} = u_\omega$. The result is a four-dimensional state space in which each state is represented as (x, y, θ, ϕ). This yields a continuous-steering car, where two action variables u_ω and u_s exist:

$$\dot{x} = \cos\theta \qquad (3.10)$$

$$\dot{y} = \sin\theta \qquad (3.11)$$

$$\dot{\theta} = \frac{tan\,\phi}{L} \qquad (3.12)$$

$$\dot{\phi} = u_\omega \qquad (3.13)$$

To convert the steering angle a C^1 smooth function of time a second integrator is applied.

Let ω be a state variable and consider u_α as the steering angle angular acceleration.

For this application, the state vector is $(x, y, \theta, \phi, \omega)$, while the state transition equation is:

$$\dot{x} = \cos\theta \qquad (3.14)$$

$$\dot{y} = \sin\theta \qquad (3.15)$$

$$\dot{\theta} = \frac{\tan\phi}{L} \qquad (3.16)$$

$$\dot{\phi} = u_{\omega} \qquad (3.17)$$

$$\dot{\omega} = u_{\alpha} \qquad (3.18)$$

The integrator can be applied as many times as desired to convert the variables smoother. The rate of change in each case can be limited according to the bounds on the phase variable and on the action set. [2]

3.3. Path planning algorithm analysis

Trajectory planning and path planning are decisive topics in the field of robotics automation. According to each different problem, it is necessary to establish the suitable trajectory taking into account that it could be executed at high speed, which means at the same time that there exist robots that can be harmed, in terms of avoiding excessive accelerations of the actuators and vibrations of the mechanical structure. Such a trajectory is defined as smooth.

Path planning algorithms are the tool to generate a geometric path, from an initial point to a final desired point, passing through pre-defined waypoints, while trajectory planning algorithms take a given geometric path and provide it with the time information. For this reason, trajectory planning algorithms are so critical in the scope of robotics, as defining the times of passage at the waypoints bias not only the kinematic properties of the motion, but also the dynamic ones. [3]

For this reason, the problem must be defined as there exists a range of different situations. The problem is classified as static when we have a perfect knowledge and information about the environment. On the other hand, it is considered dynamic if this information is imperfect or changes as the task unfolds.

According to the obstacles position in the space, we could have a time-invariant problem when they are fixed in space or time-variant if the obstacles are changing their position as time passes. If the problem is kinodynamic or differentially constrained, it means that the equations that belong to the vehicle motion act as constraints on the path.

Furthermore, if we go deeper we can continue categorizing the problem also according the AGV shape, its behavior or the kind of environment presented on the problem. But, the problem definition is focused in the two main categories presented at the beginning of the section.

3.3.1. Follow-the-carrot algorithm

Follow-the-carrot algorithm is the simplest idea of the behavior goal desired. From the car initial position, the carrot point (or goal point) is determined to be the point on the path at a look-ahead distance from the intersection point of this segment. The orientation error is one of the critical factors that must be considered. In *Figure 5,* it can be seen that it is established as the angle between the segment from the center of the vehicle coordinate system to the carrot point and the current car heading. The vehicle is pointing precisely towards the goal when the value of orientation error reaches zero.

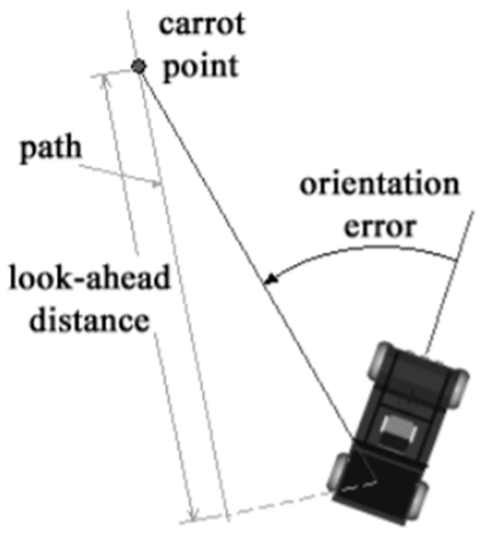

Figure 5. *Follow-the-carrot parameters.*

This algorithm is presented as an easy way to understand the idea of the project. As it is not a complex algorithm to implement, we have seen many weaknesses in its performance. The vehicle naturally inclines to cut the corners as the result of its tendency to try directly to turn towards to the new goal point each time it changes. Which is more, when the look ahead distances have a small value or using higher speeds, the result is that the vehicle oscillates about the path as it is shown in *Figure 6.*

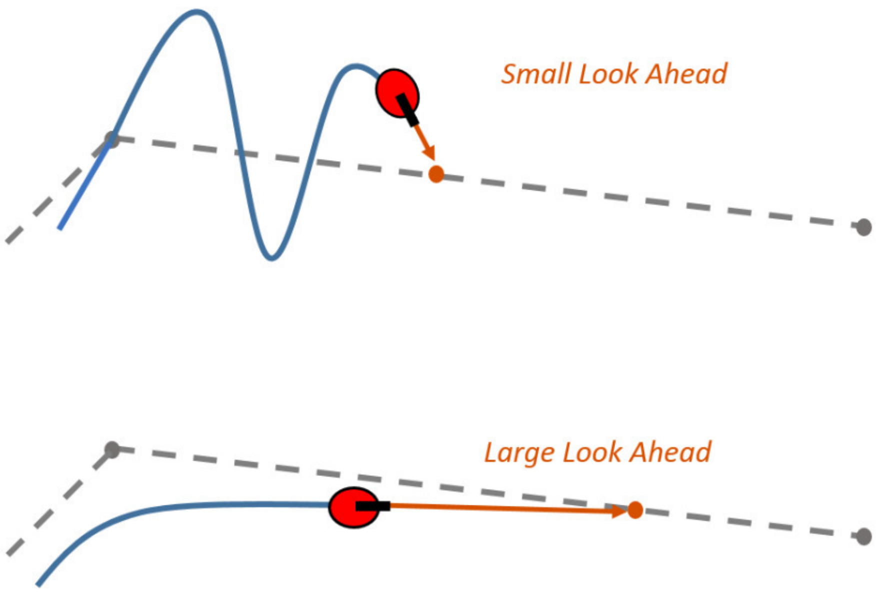

Figure 6. *Difference between small and large look ahead distance*

Despite of the ease of understanding the main idea, this project requires a good tracking performance to obtain the best results of its application.

3.3.2. Pure Pursuit Algorithm

Pure Pursuit algorithm is based on the concept of obtaining the resulting curvature that the vehicle needs to take to reach the goal point from the current position. It is also found in the literature as non-linear guidance law. The desired point is determined in the same way as it has been described in Follow-the-carrot algorithm section.

This name comes from the way of describing the methodology used in the algorithm. The general idea is the vehicle reaching the desired point on the path a defined distance in front of it. For this reason, the vehicle is constantly in pursue of the point, which gives the name to the algorithm.

To obtain the pure pursuit curve it is necessary to estimate an arch that passes through both positions, the vehicle initial and the goal points. Once the circle is obtained, the control parameter adjusts the suitable steering angle to this curvature. [4]

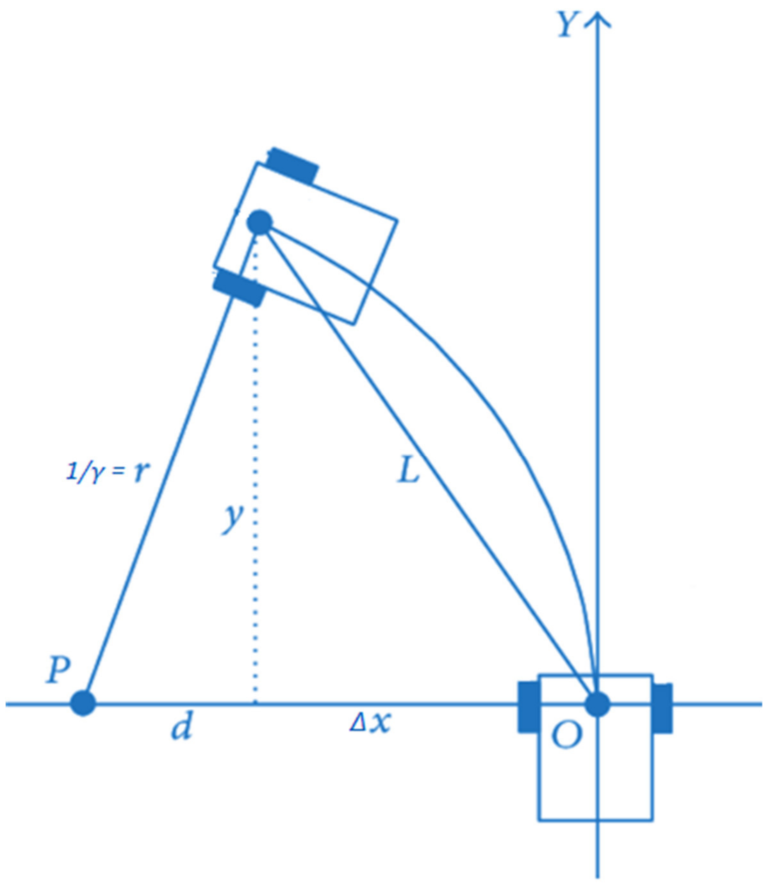

Figure 7. *Pure Pursuit curvature estimation.*

In this way, the robot changes this curvature by the adjust of the circular arcs of this type, always considering and pushing the goal point forward.

The goal and the initial look forward position of the vehicle is given in coordinates. According to that, the coordinate system of the robot is defined where the y-axis is in the forward direction of the vehicle, the x-axis is perpendicular axis to y-axis. The sensors used in this project give information about the x,y position of the vehicle and also its orientation in relation to the x-axis measured in radians.

The parameters represented in *Figure 7* are required to estimate the required value of the vehicle curvature: *Δx* is presented as the offset of the goal coordinate over the x-axis from the initial point, *L* is the length that separates the car current position and the desired point, and *r* is the radius of the circumference that passes by both points, the current position and the goal coordinates. To obtain the required curvature of the

AGV is applied:

$$\gamma = 2\,\Delta x/\,L^2 \qquad \textit{(3.19)}$$

To obtain the previous formula it is just only necessary the development of the two following equations:

$$\Delta x^2 + y^2 = L^2 \qquad \textit{(3.20)}$$

$$\Delta x + d = r \qquad \textit{(3.21)}$$

Equation (3.20) is the application of Pythagoras' theorem over the triangle drawn in *Figure 7* following Δx , *y* and *L* side. On the other hand, *Equation (3.21)* is the definition of the radius as the sum of the different elements that comprise x-axis. To obtain the curvature formula is just enough introducing the second equation into the first one:

$$d = r - \Delta x \qquad \textit{(3.22)}$$

$$(r - \Delta x)^2 + y^2 = r^2 \qquad \textit{(3.23)}$$

$$r^2 - 2r\,\Delta x + \Delta x^2 + y^2 = r^2 \qquad \textit{(3.24)}$$

$$2r\,\Delta x = L^2 \qquad \textit{(3.25)}$$

$$r = L^2/2\Delta x \qquad \textit{(3.26)}$$

Once the radius of the circumference is obtained; next step is to compute the constant curvature which is inversely proportional to its radius:

$$\boldsymbol{\gamma = 2\,\Delta x/\,L^2} \qquad \textit{(3.27)}$$

All the points that shape the path to follow share the same coordinate system and it is not necessary transform any given coordinates. For this reason, the algorithm can be explained following iterative steps as it is resumed in the diagram of *Figure 8:*

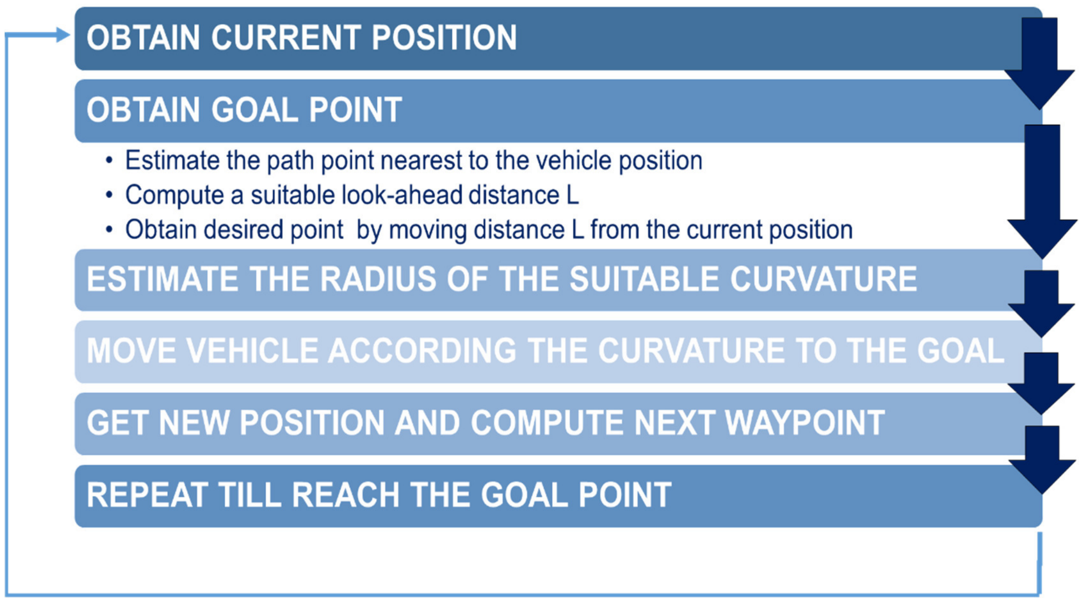
OBTAIN CURRENT POSITION
OBTAIN GOAL POINT
• Estimate the path point nearest to the vehicle position
• Compute a suitable look-ahead distance L
• Obtain desired point by moving distance L from the current position
ESTIMATE THE RADIUS OF THE SUITABLE CURVATURE
MOVE VEHICLE ACCORDING THE CURVATURE TO THE GOAL
GET NEW POSITION AND COMPUTE NEXT WAYPOINT
REPEAT TILL REACH THE GOAL POINT

Figure 8. *Pure Pursuit algorithm steps*

Comparing Pure Pursuit algorithm with the Follow-the-carrot algorithm presented before it is easily perceived some improvements in its performance. About the oscillations presented in Follow-the-carrot algorithm when a short look ahead distance or high speeds were given in Pure Pursuit algorithm are decreased also improving heading angle errors. Another highlighted point is its behavior when corners appear, while Follow-the-carrot algorithm cuts them Pure Pursuit shows a smoother trajectory with better accuracy tracking the curves of the path.

Despite of presenting good results using this algorithm, some improvements have been introduced in this project to obtain better results, which will be discussed later in *Chapter 5: Experiment improvements.*

3.4. Obstacle Avoidance analysis

Nowadays, autonomous vehicles and aerial robots are taking an important role in technology industry till the point that some companies are developing and testing their first models with the idea of introducing them into human daily lives as soon as possible. One of the most important issues in these kind of applications is obstacle avoidance.

Obstacle avoidance is the ability of determining a collision free path through the previous analysis of the input data sent by different sensors that endows the robot the information about where the obstacles are located. For this reason, in this chapter an obstacle avoidance overview to determine the suitable algorithm is presented.

Obstacle avoidance can either be done locally, globally or through a combination of both. The local obstacle avoidance systems use real-time sensor data to identify and avoid obstacles around the robot nearness. On the contrary, global obstacle avoidance systems select the most suitable trail for a robot to follow according to a map of the environment known by the robot. However, there exist systems that combine both local and global obstacle avoidance.

In this Chapter, the study of this field is presented starting on the basis of the Potential Field algorithm. According to that, new algorithms have been built taking into account the previous theoretical base and new improvements. The development of these algorithms till reaching the final selected one is discussed along this Chapter.

3.4.1. Potential Field Algorithms

Potential field algorithms allocate a gradient vector to each different point on the manifold using potential functions. The gradient representation can be interpreted as forces in different directions acting on the robot, where the robot could be a positive charge attracted to the negatively charged goal. Obstacles also could be determined as positive charged, so that between the obstacles and the robot exist repulsive forces. [5]

From this set of forces, the result of a good and balanced performance is obtained where the robot is guided by induction to the goal and moved away from an obstacle in the form of repulsion force. *Figure 9* gives us a graphical view of potential function representation according the principles stated before:

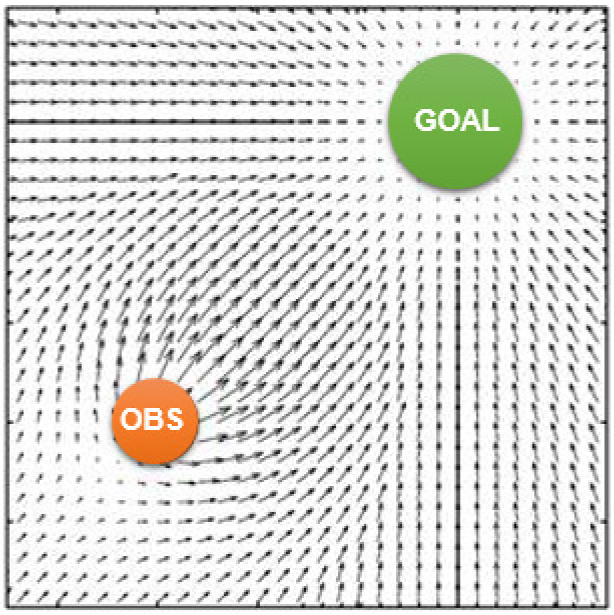

Figure 9. *Potential function representation in potential field algorithms*

Furthermore, potential functions can be described as a surface comprised by different heights and hollows where the robot tends to move from regions with higher values to the lower ones. The path described by the robot following lower values finding the minimum is often referred to as gradient descent. The real problem of this function is that the robot can easily be stuck in a point which belongs to a local minimum. Local minimum points are the areas where the gradient value where the robot is located is lower than all the gradient value points around the robot location.

3.4.2. Virtual Force Field Algorithm

Virtual Force Field algorithm is a real-time obstacle avoidance algorithm created to be implemented in mobile robots. The algorithm applies wall following techniques that drive the robot to find sometimes some local minimum locations. Its performance is composed by a histogram grid, where the obstacles are represented, and the potential fields to navigation actuation [6]. Three stages are highlighted to explain Virtual Force Field algorithm:

- **Histogram grid *C***

Obstacles are represented through a two-dimensional Cartesian histogram grid noted by *C*, where each cell is defined by *(i, j).* Another important factor is the certainty grid value represented in *c (i,j),* which refers to the confidence that an obstacle belongs to a concrete cell. Confidence values are updated by a probability function that takes into account the properties of a given sensor. If an obstacle is detected, it is more likely that this object is located closer to the laser axis of the sensor than to the periphery of the conical field of view. According to this, *C* increases certainty values in cells close to the axis more than certainty values in cells located in the outside.

The mobile robot will stay stationary while it is taking a scan with Lidar sensors. The certainty grid is updated after the application of the probabilistic function to each of the range readings. For each range reading, the cell that lies on the laser axis and corresponds to the measured distance *d* is augmented, increasing the certainty value of the cell.

When some sensor information is used, only one cell's certainty value is increased for each range reading. In order to get a probabilistic distribution, the robot moves to a new position, stops and repeats the process. It must be continuously moving and all its sensors must be sampling information to update all the information from the histogram grid.

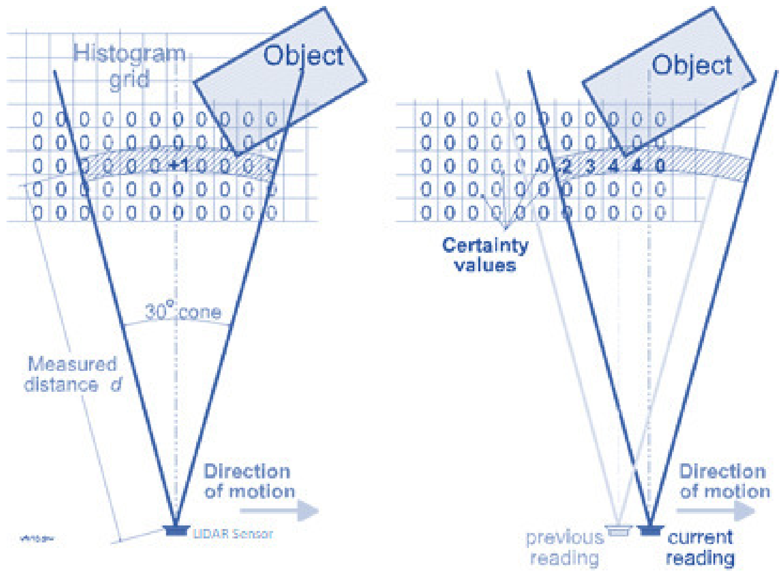

Figure 10. *Projection of the conical field of view of a LIDAR. (J. Borenstein and Y. Koren, "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots")*

- **Potential field Algorithm**

  According to the histogram grid's probabilistic sensor information collected as is represented in *Figure 10*, the robot steering is computed through the potential field algorithm.

  In each moment, there is a region centered on the robot position which moves together as the robot scrolls along the histogram grid. This is known as active window, denoted by *C** within its active cells *c*(i,j).*

  Each active cell experiments a virtual repulsive force *F (i,j)* in the direction of the robot. To compute the magnitude of this force is enough to know that it is proportional to each active cells' certainty value *c*(i,j)* and inversely proportional to d^{x}. The value of *d* is related to the distance that separates the center of the robot from the active cell, while the value of *x* is agreed to have a value of 2 for mobile ground robots by Borenstein & Koren (1989).

  All the repulsive forces from the active cells are represented by a single repulsive force vector F_{r}, at the same time F_t refers to the force vector in charge of leading the robot in goal position direction represented in *Figure 11.*

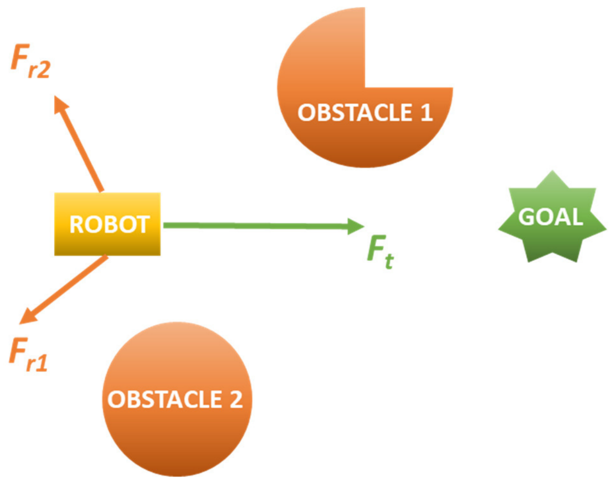

Figure 11. *Forces representation*

  The result of computing the sum of both force vectors is force vector *R,* which will be the vector used to steer the robot to the goal location avoiding obstacles in its path.

- **Combination of previous knowledge**

  The application of the histogram grid and the potential field algorithm in real-time makes possible that sensor data has instant effect on the steering control of the vehicle.

Each sample data is used to update not only the histogram grid, also the resulting force vector *R*. In this way, the vehicle can perform a faster response at the moment that obstacles suddenly appear.

On the other hand, under concrete conditions it can occur that the robot gets stuck as the result of being enclosed in a local minimum. In *Figure 12 it* is represented the situation described before. The robot is following the path attracted by the goal point force according that the obstacle repulsive force is smaller. However, at the point represented below the repulsive force is increasing till the moment that both forces are compensated and resultant vector force is equal to zero or the robot changes 180 degrees the movement direction. If this behavior occurs, the resultant force vector will be varying oscillatory the direction at the same time the robot is moving back and forth towards and away the desired final point.

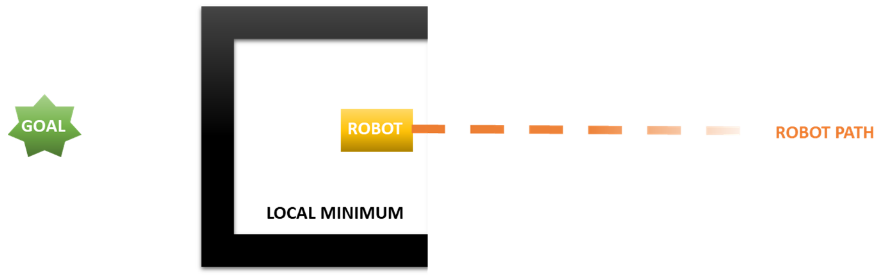

Figure 12. *VFF algorithm application in local minimum scenario*

As a solution to the local minimum problem, it is presented the wall following method, which consists on guiding the robot around the obstacles when a local minimum is detected. Local minimums are detected according the robot will encounter the local minimum problem by the traditional artificial potential field.

Now, the repulsive forces are all contained in a repulsive force vector F_r. Instead of the original attractive force, a new virtual attractive force is estimated adding or subtracting a β angle from or to F_r direction. By the application of these two considerations the robot will be guided to the left or to the right around the obstacle always in a direction parallel to the obstacle limits respecting a fixed distance to the wall. The authors [7] recommend a β angle between 90º-180º, in this way the robot satisfies wall following method until the angle between the goal direction and the robot's direction becomes less than 90º.

What is more, Virtual Force Field algorithm does not enable the robot to pass between two obstacles placed together. In the scenario represented in *Figure 11*, at the same time the robot is attracted to the goal is suffering repulsive forces from the side obstacles. The Wall Following Method presented before would act guiding the robot around one of the obstacles to reach the goal instead of letting it pass between their free spaces.

One of the most common issues observed using this algorithm is the application in large and narrow aisles, where the vehicle performance is not good enough due to the repulsive forces from the nearest wall.

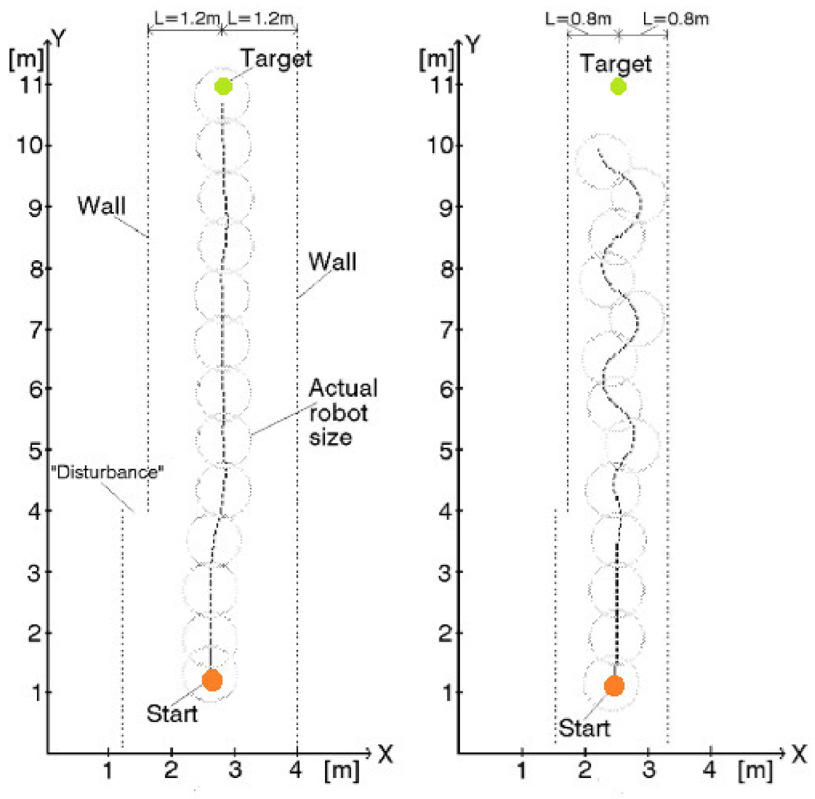

Figure 13. *Robot performance according the corridor width*

As it is observed in *Figure 13,* the change of the repulsive force intensity from one wall to the other makes the vehicle oscillate continuously from side-to-side, which makes it difficult obtain good results.

3.4.3. Vector Field Histogram Algorithm

Vector Field Histogram (VFH) algorithm is the improvement of Virtual Force Field algorithm developed by Borenstein & Koren [8]. Whereas the VFF algorithm uses a single-step data reduction process, VFH operates in a two-stage data reduction process and highlights three layers to represent data:

- **Highest level**
  To represent and describe the vehicle's environment a two-dimensional Cartesian histogram grid *C* is needed. Through the range sensors of the robot all the sample data is obtained to create and develop a real-time model of the robot's world. *Figure 14* suggests an example of the environment to explain the steps of the algorithm in a visual form.

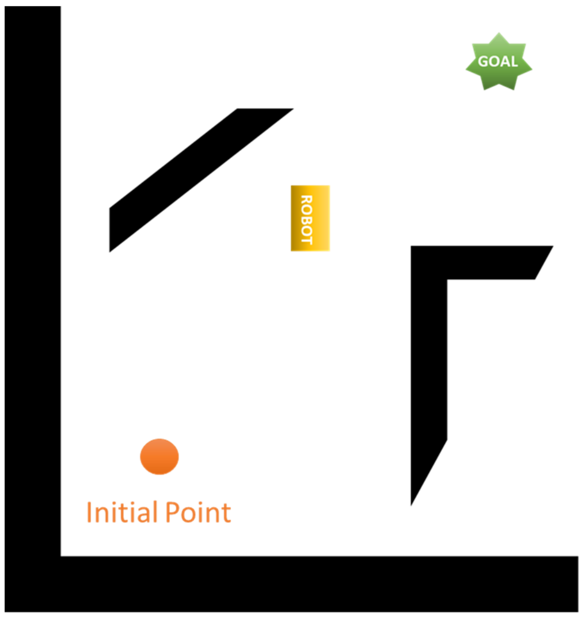

Figure 14. *Robot and obstacles positions*

Cells with a higher chance of an obstacle are represented with greater certainty values, which are denoted with different intensity according the grey scale as in *Figure 15:*

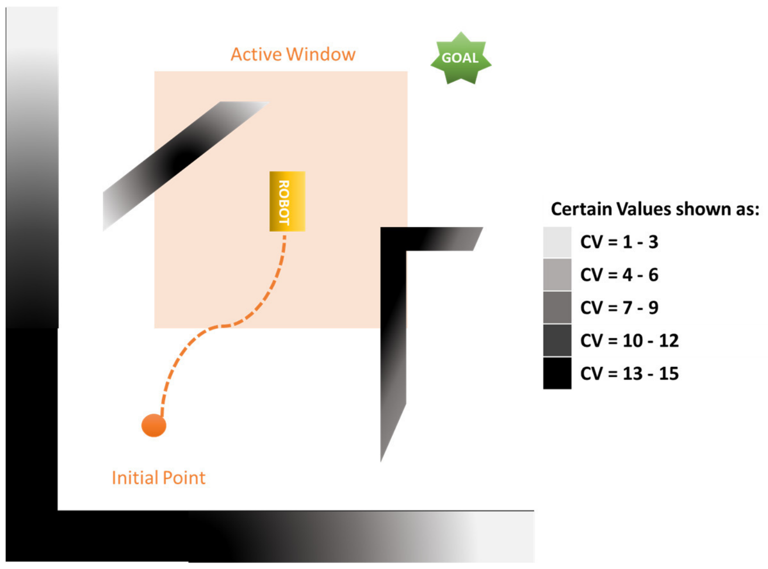

Figure 15. *Histogram grid representation of robot's environment.*

- **Intermediate level**

  To build the one-dimensional polar histogram *H*, the active region *C** is needed, which is centered on the robot and lies on the two-dimensional Cartesian histogram *C* estimated in previous stage. The information obtained from the certainty cells located in the active region is used to build *H.*

  This polar histogram is developed around the robot's instantaneous position and is composed of an *n* sectors division with β angle range. In this manner, each sector has a particular value that is representative of the polar obstacle density (POD) in the direction of that specific sector.

  In *Figure 16* it can be observed the intermediate step to build the polar histogram:

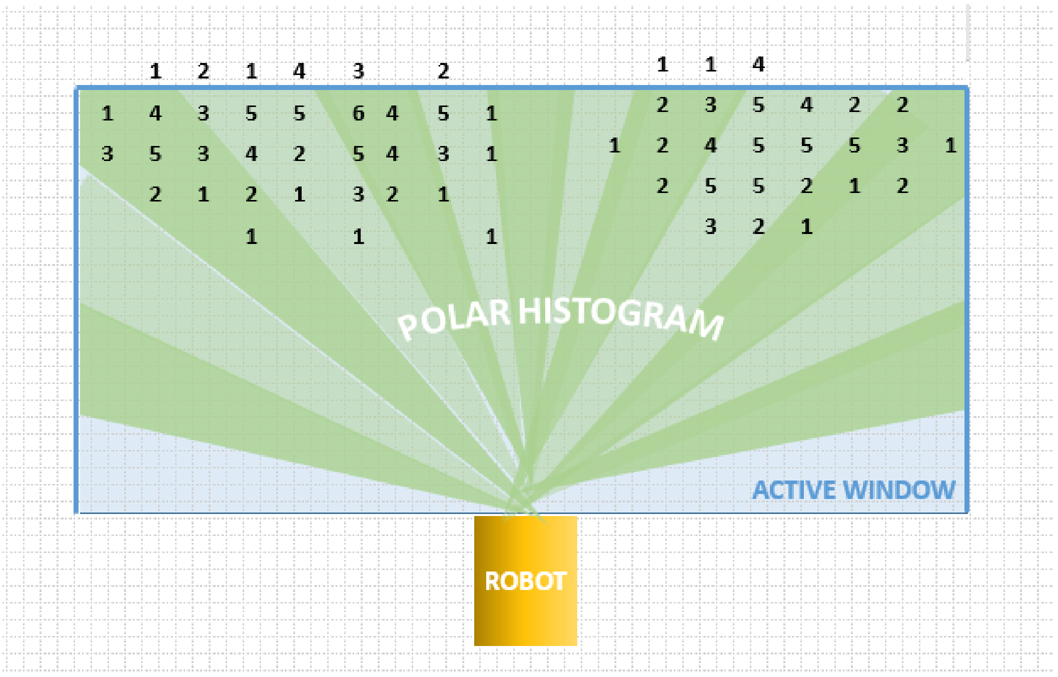

***Figure 16**. Polar histogram construction*

So, the polar histogram represents the polar obstacle densities, where the sectors form a 360-degree circle around the robot. These obstacle densities can also be attached in a histogram, *Figure 17.*

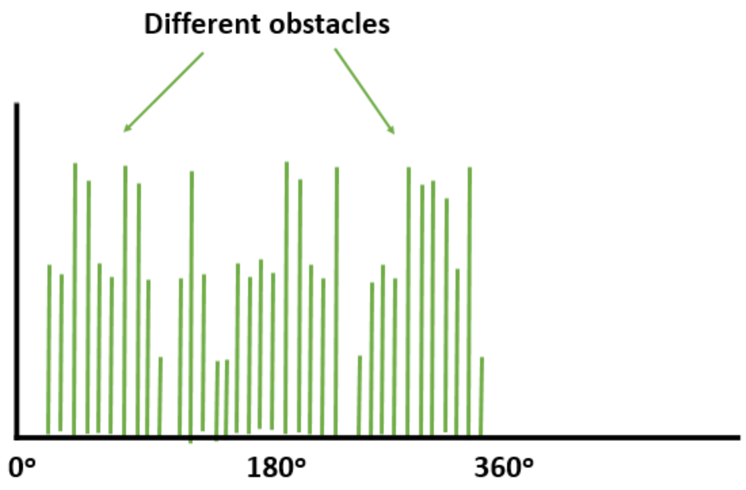

***Figure 17.** Obstacle histogram by sectors using thresholds*

- **Lowest level**
  Finally, at this stage the reference values for the robot's drive and steering controllers are obtained. From the polar histogram grid, the open sectors can be differentiated as candidate directions to guide the robot, while the closed sectors represent the unsafe directions to follow. So, the new direction of motion is determined according to the candidate direction that better guides the robot towards the goal stablished.

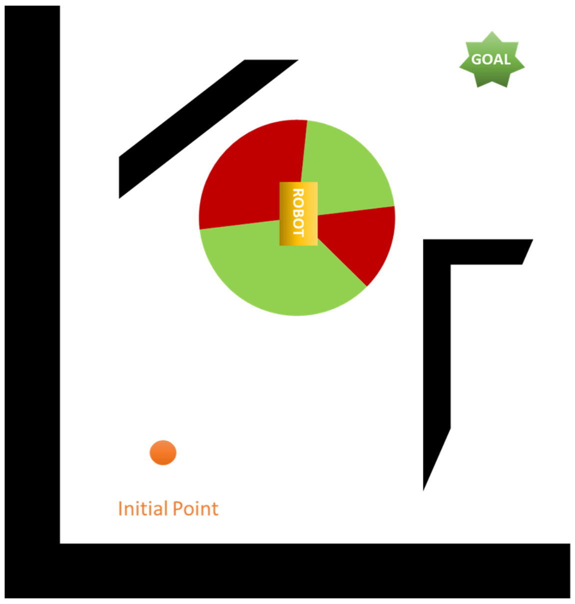

Figure 18. *Candidate steering directions. Polar histogram*

As it is shown in *Figure 18*, green colored sectors represent the free-obstacle and safe candidate directions, while the red filled sectors denote the unsafe movement directions.

3.4.4. Vector Field Histogram + Algorithm

VFH + algorithm introduces an enhancement to VFH algorithm: it takes into account the robot's radius or width along with the accessible trajectories. [9] Vector Field Histogram + increase from the two-stage data reduction process used in VFH to four-stage data reduction process:

- **First stage**
  From the two-dimensional Cartesian histogram grid, VFH + algorithm builds a one-dimensional primary polar histogram that takes into account the radius

of the robot. Using the polar histogram crated before each obstacle cell in the active windows is enlarged to allow that it counts for the robot's width. In *Figure 19,* it can be checked how to enlarge each obstacle cell not only is considered the robot's radius r_{robot}. The algorithm also contemplates an additional safety distance, r_{safety}. For this reason, the final safety region around the possible obstacles is the sum of both r_{safety} and r_{robot}.

$$r_{s+r} = r_{safety} + r_{robot} \qquad (3.28)$$

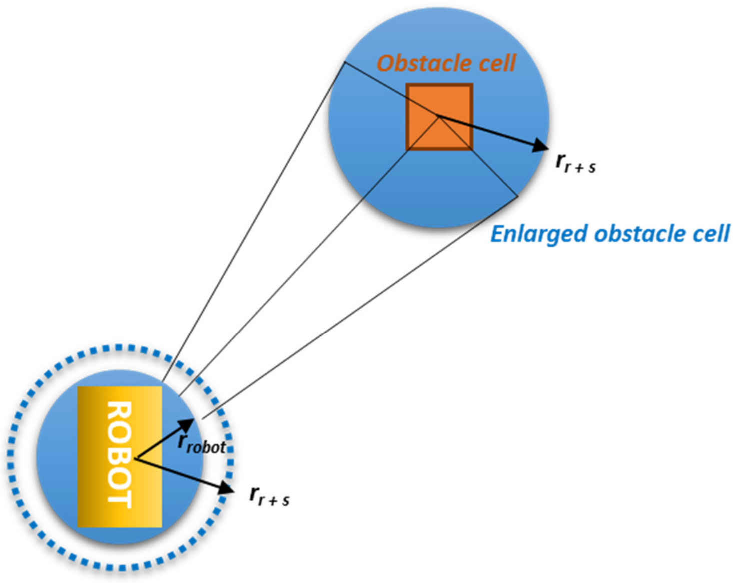

Figure 19. *Obstacle enlargement according robot width and safety distance*

- **Second stage**

  At this level, the primary polar histogram computed before is converted using thresholds to transform the different sectors either open space or blocked according the polar obstacle densities, to binary polar histogram form. *Figure 20* represents the primary polar histogram. It can be compared with *Figure 21*, to see how it changes after threshold application:

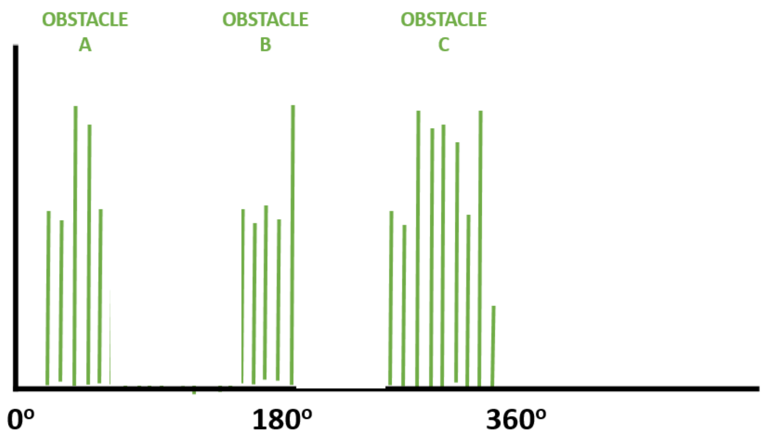

Figure 20. *Primary polar histogram representation*

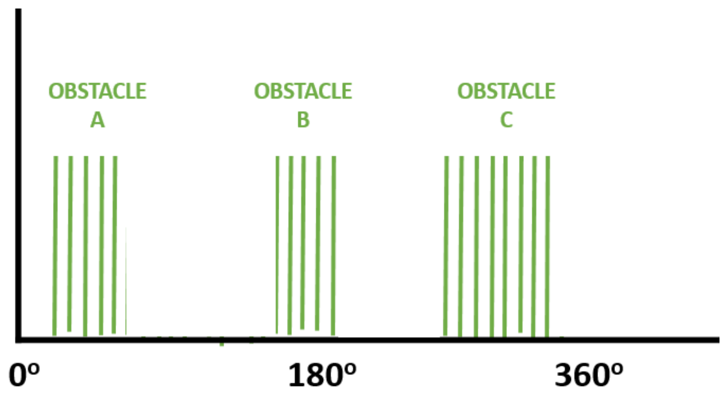

Figure 21. *Binary polar histogram representation of Figure 20*

- **Third stage**
  To continue with the second stage, a mask is applied to the binary polar histogram taking into account the different trajectories that the robot can follow. This step is applied taking into account that most robots are not able to change the movement direction in an instant way (*Figure 22 (1)).* They are subject to turn circles or trajectories according robot's configuration, speed and dynamics (*Figure 22 (2))*. Under this constraint, the available steering directions the algorithm can select are reduced.

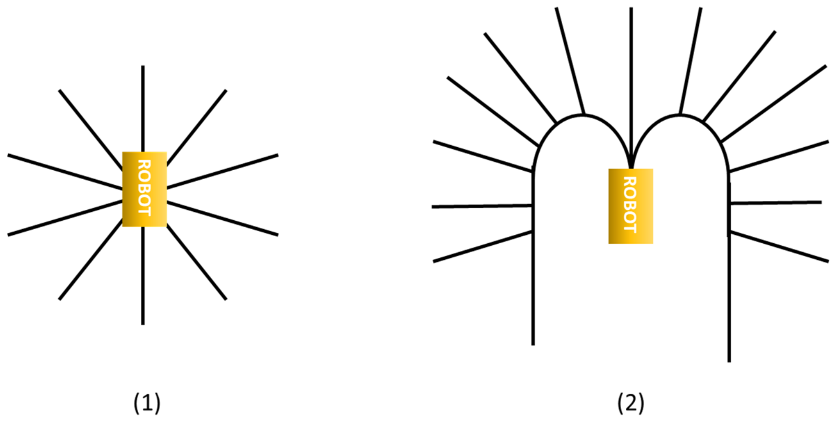

Figure 22. *Robot trajectories*

- **Fourth stage**

  Last stage corresponds to last step to finish VFH + algorithm process: the algorithm determines the best direction to develop the robot is steering based on the possible directions available determined along all these stages.

4. Software development

To explain the part related to software development, it has been stablished to structure *Chapter 4* in different sections in order to describe more in detail the topics as: the software platform, the code structure, the different element definition and the outputs selected to show if it works properly.

As we have exposed in previous chapters, software development is through Matlab platform, where we would create a ROS simulation to represent and check the behaviour of our project.

4.1. Environment

The goal of this project is to make a study of the different path following algorithms and collision avoidance techniques. From this point, the robot will have distinct behaviours under different environments. This occurs according that each of the parameters will be updated following the robot current conditions.

For this reason, a set of maps has been created to check how the robot performance is, and if there exist some differences between them. For instance, the robot should not have the same difficulty if it is surrounded by many obstacles or if it is in front of a free area.

The maps have been created starting from a simple environment. This map has been considered as a reference to add different kind of obstacles in. *Figure 23* represents the simplest possible map, an obstacle-free area where is only delimited by its own borders.

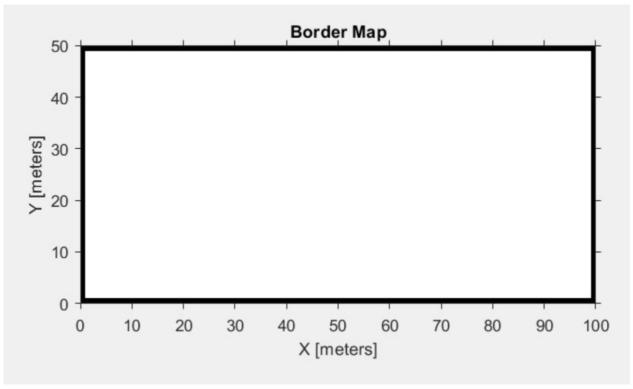

Figure 23. *Border map representation.*

Figure 24 and *Figure 25* represent two maps delimited by horizontal and vertical corridors in order to evaluate the robot's performance.

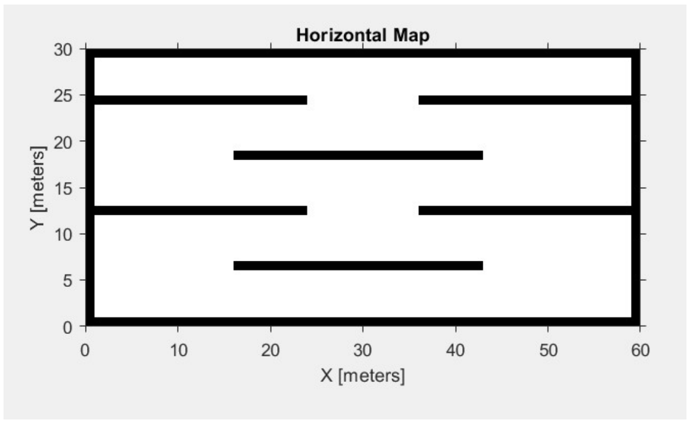

Figure 24. *Horizontal map representation*

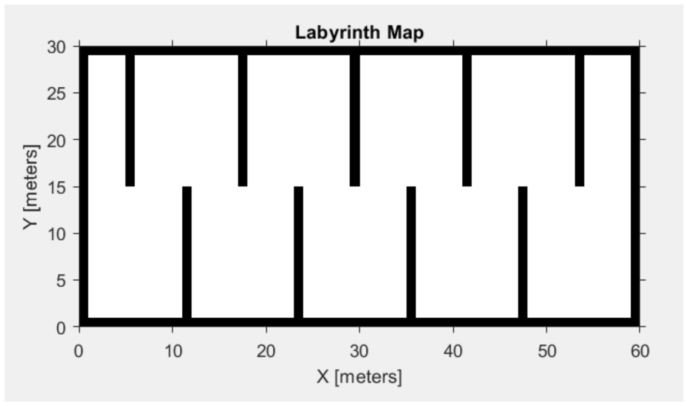

Figure 25. *Vertical Labyrinth map representation*

Squares, branches or polygons have been considered also as obstacles in other kind of maps as they are represented in *Figure 26, Figure 27* or *Figure 28.* Maps also have been developed varying their dimensions, in order to offer totally different environment conditions.

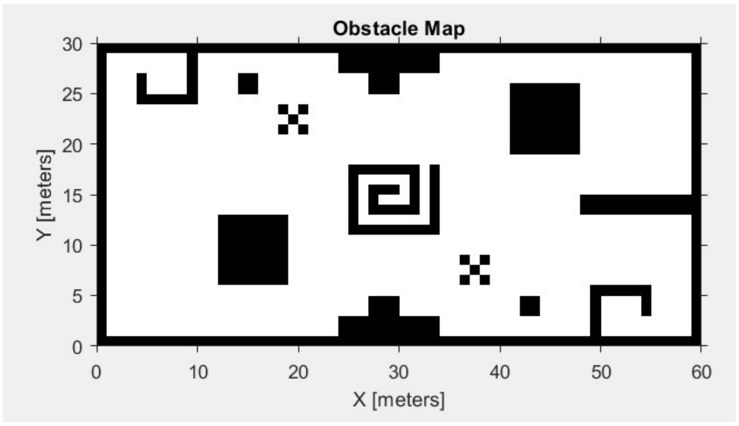

Figure 26. *Obstacle map representation*

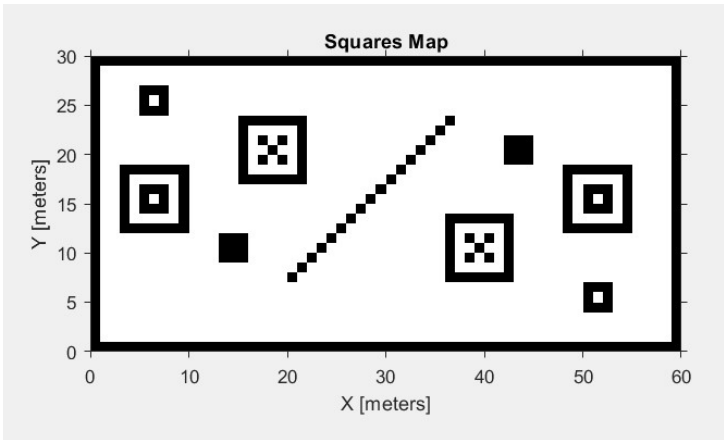

Figure 27. *Squares map representation*

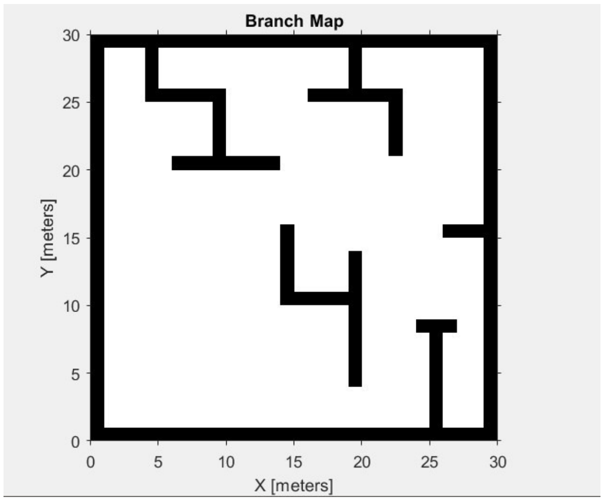

Figure 28. *Branches map representation*

All the maps have been designed individually according the robot performance situations we want to observe. They have been implemented setting different values in an occupancy grid map, with a resolution value of 2 cells per meter. They are organized under the same structural variable in order to keep them together. In this way, it is easier to specify which map we want to simulate from the main code.

The characteristics of the environment are unknown for the robot. It will be discovering the obstacles at the same time it moves, and LIDAR sensors are sampling all the information around it.

4.2. Simulator Structure

As it has been seen in previous chapters, the project consists on a Matlab software simulated using ROS platform. ROS will simulate the LIDAR sensor providing all the information related to the scan measurements and the robot position. To develop this part, we need to create subscribers to receive the data from ROS topics, and publishers to send the results in a message to the simulator.

To explain this blocks, this chapter has been divided as follows to offer a clear idea of how these connections work.

4.2.1. Inputs

To develop this part, two ROS subscribers have been considered: one to receive the data related to LIDAR sampling, and the other one to obtain the current position of the robot.

The first subscriber obtains and reads the messages sent on the */scan_samples* topic. The LIDAR Scan message is computed in order to get the scan range and its angles. Scan_Ranges is a vector which represents the distance measurements in the direction collected in Scan_Angles vector to robot's body frame.

The second subscriber is necessary to get the coordinate position (x,y) of the robot, and its yaw orientation. Through the messages received from */robot_pose* topic, yaw orientation is estimated converting quaternions to Euler angles. In this way, Robot_Pose is the vector with the current robot's position and orientation.

In addition, the waypoints are considered as system inputs.

4.2.2. Speed Configuration

In this part, we must take into account two different stages. Pure Pursuit configuration aimed to compute linear and angular velocities in order to follow the current path according the AGV dynamics model.

In case that an obstacle exists in the target direction, Vector Field Histogram + algorithm recalculates a new linear and angular velocities. In this way, the robot has a new target direction that makes feasible to reach the goal.

4.2.3. Outputs

Once we have the linear and angular velocities that allow the robot movement, it is necessary to establish the publishers to send all the information to ROS simulator. The velocities computed using the path following algorithm according the AGV model seen in *Section 3.2* are added to the velocity adjustments estimated using VFH algorithm. This allows to modify the speed in case of obstacle. Final velocities are set on *geometry_msgs/Twist* message and published on the topic *mobile_robot/commands/velocity.*

When a new LIDAR message is received, the subsystem is activated. In this way, only when the new sensor data is available, the speed command is published. The aim of this subsystem performance is to avoid the robot from crashing with the obstacles in case of data delay.

4.3. Occupancy map

As it has been explained before in *Section 4.1*, the robot has not information about the environment is moving through. However, as time passes and the LIDAR sensor is sending data, the robot has clues about the obstacles around. To offer a clear view of the obstacles and walls the robot now can see, an occupancy map has been developed.

When the sensors send information about the position and the distance of the object, the occupancy map cells are coloured following the gray scale. Cells with a higher probability of an obstacle are represented with greater certainty values, which are denoted with different intensity according the grey scale as it has been explained in *Section 3.4.3.*

So, if an obstacle is identified many times it will have a darker representation than if only the robot detects it once or twice. In this manner, we can have an idea of the total amount of the map that the robot has detected. *Figure 29* shows an example from the robot's performance over the vertical labyrinth map:

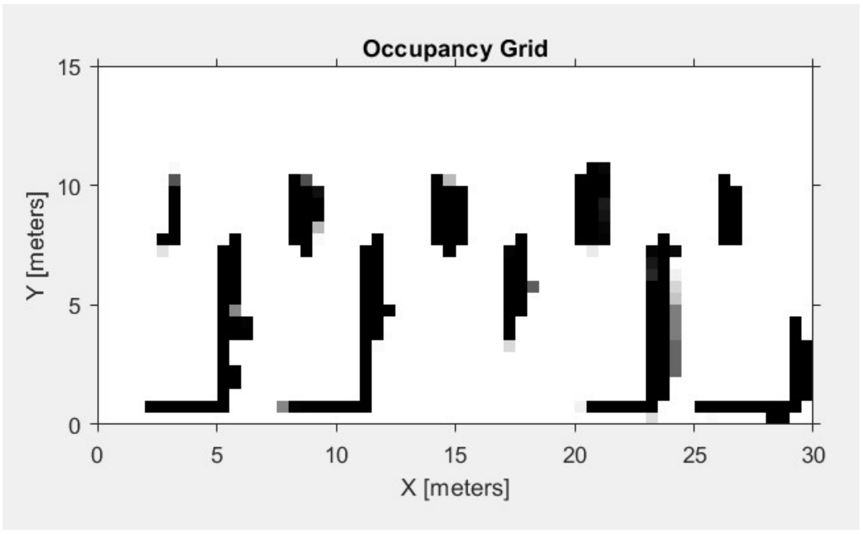

Figure 29. *Occupancy grid from Vertical Labyrinth map performance*

To implement the occupancy map in Matlab, a function that checks if the code has been executed previously has been created. If it is the first time, a map is created, whose dimensions and resolution are the same that the environment map selected as in *Section 4.1.* From this point, each LIDAR ray data is represented. In this way, according the samples rises, the occupancy grid represents more obstacle information.

4.4. PRM Map

As it is explained in the next chapter, after the observation of the resulting performance, we decided to improve the software.

When the robot found an obstacle that blocks or is located near of its trajectory, it moves away to try to avoid it. However, we sometimes found that the computation time increases considerably as a result of robot's exploration. To enhance the behaviour and to offer a better result, it has been considered to include a probabilistic roadmap planner. All this information is explained in detail in *Chapter 5.*

In this way, it has been implemented a function that works in case of the robot detects an obstacle close enough of its trajectory. If this occurs, the PRM planner would evaluate a set of samples to determine the best set of waypoints till reaching the goal. In this way, the new waypoints are computed and updated to offer a new free-obstacle path. If new obstacles appear, the waypoints will be recomputed again.

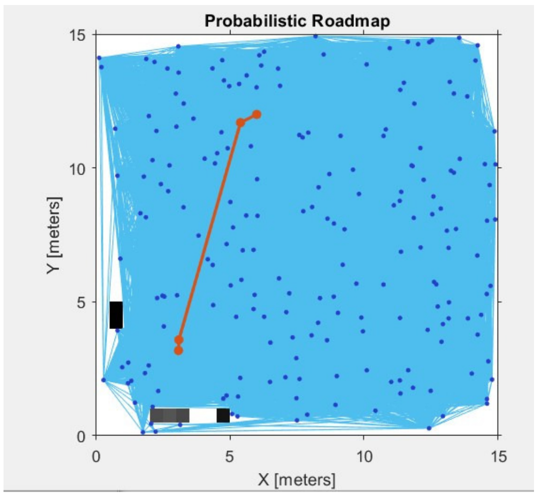

Figure 30. *Probabilistic Roadmap example*

The main idea is to optimize the trajectory till the final goal. So, this process will be repeated as many times as necessary till the robot gets the final goal. In *Figure 30* the iteration when objects are detected close enough is represented. As it can be seen the cells that belong to the elements detected by the robot are coloured in black.

4.5. Main structure

Along this chapter it has been described the most relevant parts of the software development. However, it has been considered to specify more in detail how the code implementation works.

In addition to the previous functions presented along this *Chapter 4*, the software takes into account the established configuration to set the simulator parameters. This configuration refers to the maximum linear and maximum angular velocity of the robot, robot's initial pose, robot's bounding radius, or to enable Laser sensors between other characteristics.

From the main script, we are allowed to determine which is the desired goal and which map we want to upload. Once it is working, the main structure is activated.

The simulation starts as it has been configured previous, and the unknown environment map for the robot has been selected. From this point, the information from the laser scan samples is received from the subscribers as it has been explained in *Chapter 4.2.* According to that, the robot's position and orientation are obtained. Also, from LIDAR sensors, the range distance measurements and their angles are extracted.

The map is printed showing the desired trajectory and the robot position. From the centre of the robot, there is represented a set of laser rays covering a 180-degree area in order to symbolize the LIDAR range. At the same time the occupancy grid plot will be displayed. At the beginning, the occupancy grid is supposed to be empty as the robot has not information about its surrounding.

The speed of the robot is computed according the AGV dynamics model implemented in Pure Pursuit algorithm. The information computed will be sent in a message through the publisher to be updated at the simulator.

As the number of LIDAR samples increase, the process repeats in the same way. However, the different values from the LIDAR rays will update the occupancy grid as it has been described previously in *Chapter 4.3.*

When an obstacle is close enough, VFH+ algorithm detects it and computes a new steering direction in order to avoid it. To improve this situation, PRM algorithm has been implemented. When the desired trajectory is crossing an object detected by LIDAR, PRM computes a new trajectory.

To check if the current desired trajectory is crossing an obstacle, next steps are followed:

- The line equation between one waypoint and the following one is computed.
- 100 samples (x and y pairs) are taken from each equation.
- For each of the computed samples, it is tested if it belongs to an object detected by LIDAR.
- If one of the samples belongs to an obstacle, PRM recalculates the path giving a new set of waypoints.

According both situations, finally the robot reaches the desired goal.

5. Experiment improvements

Along this chapter are exposed some changes introduced to improve the performance of the different algorithms. The reason to develop these modifications is because once compared the results obtained and detailed along the document, we noticed some lack of performance that we consider to implement.

So, it consists on determine the new waypoints of the path each time an obstacle is detected in a small distance. In order to compute them, a Probabilistic Roadmap Method is applied to determine which are the feasible points of the map to reach the goal from the new current position. In *Chapter 5.1* it is discussed how this planner works to know more in detail the improved version of the algorithm performance.

5.1. Probabilistic Road Map planner

The probabilistic roadmap method is a new approach to motion planning. It seems to work efficiently, be easy to implement, and appropriate for a large range of motion planning problems. The basic motion planning problems ask for computing a collision-free, feasible movement for the robot from an initial given point in an environment with obstacles around. [10]

From a general view, PRM methodology works sampling the configuration space for collision-free locations. The configuration space is denoted as the space of all possible placements for the moving body. Where a collision-free space is found, these are added as nodes to the roadmap graph. Pairs of favorable nodes are selected in the graph and a simple local motion planner is used to try to connect these nodes with a path. The process is repeated until the graph covers all the possible connections of the space. Practical implementation of PRM method consist on two stages: preprocessing and query phase. [11]

This kind of problems are normally expressed in terms of the configuration space *C*. Each dimension of the configuration space corresponds to a degree of freedom of the object. All the obstacles in the configuration space *C*, have been transformed from obstacles in the workspace where the robot moves. They constitute the obstacle part of the configuration space, denoted as C_{obs}.

The path for the moving robot corresponds to a curve in the configuration space joining the initial and goal configuration. The condition for a path to be collision free is determined by the location of its associated curve. This means that the curve does not intersect C_{obs} and consequently, is located in the free area of the configuration space C_{free}.

The task of the PRM planner is sampling the configuration space looking for free configurations. Additionally, it tries to join these configurations into a feasible movement path roadmap.

Despite of existing different versions of PRM, all of them perform under the same theoretical principle. The main idea of this planner is to collect a range of configurations in the free configuration space. Free configuration samples create the nodes of the graph *G = (V, E),* where *V* refers to vertices and *E* to the edges.

At roadmap construction stage, random nodes or configurations of the robot are generated over the configuration space. From a simple local motion planner, the path is created trying to connect all these configurations. One of the characteristics of the local planner is that it must be quite fast, but it is allowed to fail under certain difficult cases.

To develop the interconnection, a connection distance is determined in the configuration space. In this way the local planner only tries to link the nodes within a predefined distance. Each of the successful connections yields an edge on the roadmap.

Once a large number of nodes is generated, the narrow parts of the configuration space are recognized heuristically and more nodes are placed near these zones. The purpose of this improvement step is to ease the information of roadmap components that correspond to the elements of configuration free space. In *Table 1* is represented the pseudo algorithm steps of the Road Map Method.

```
Let: $V \leftarrow 0$ ; $E \leftarrow 0$ ;
1:  loop
2:      $c \leftarrow$ a (useful) configuration in $C_{free}$
3:      $V \leftarrow V \cup \{ c \}$
4:      $N_c \leftarrow$ a set of (useful) nodes chosen from $V$
5:      for all $c' \in N_c$, in order of increasing distance from $c$ do
6:          If $c'$ and $c$ are not connected in $G$ then
7:              If the local planner finds a path between $c'$ and $c$ then
8:                  Add the edge $c'$ to $E$
```

Table 1. *Build Road Map pseudo algorithm.*

Once the computed graph verifies the connectivity of C_{free} it can be used to answer motion planning queries.

Path planning query is used to specify the initial and goal configuration of the robot. Through this step, both are included to the graph using the local planner. The objective is to find a motion between the start and the goal configurations. PRM first connects the initial configuration to a roadmap node by attempting to connect it as a first choice with the nearest roadmap nodes. When the connection is established, the goal configuration is tried to link to the same component. The new nodes correspond to the nodes which initial and goal configurations get connected respectively. From this point, the search of the roadmap can build the final path between these two nodes. [12]

In *Table 1,* the pseudo code is represented to obtain a clear vision of the steps needed to build the Road Map.

According to the robot and the workspace, PRM requires the tuning of some parameters.

In this project, the number of nodes and the connection distance have been considered. Using a connection distance allows us to set a threshold for distance in order to obtain the connections between all feasible points whose paths are not blocked.

In addition, it has been also considered how the performance changes increasing or decreasing the number of nodes. A higher number of nodes means a higher efficiency computing more feasible paths. Conversely, a higher number of nodes increases the computation time. We have experiment some changes in the selection of the number of nodes as a consequence that it slows down the final performance of the entire algorithm developed. Finally, the number of nodes established in this project is between 100 and 200.

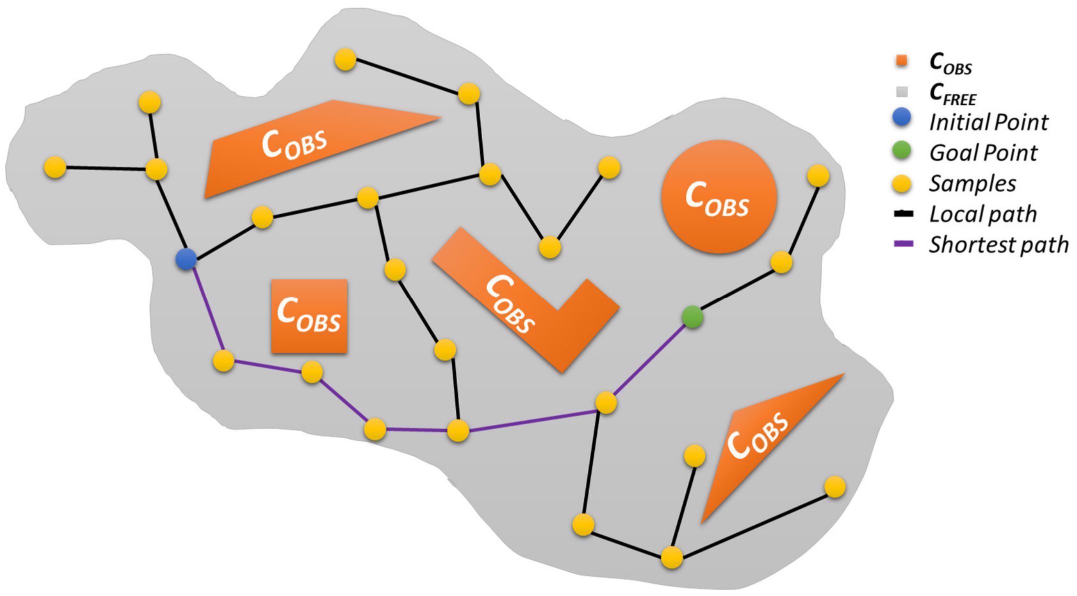

Figure 31. *PRM planner representation*

In *Figure 31* is described the different components of the PRM planner are explained before. We can observe in grey the free configuration space C_{free}, while the orange areas denote the obstacle configuration space representation C_{obs}. From the blue initial point, to the green goal point, many different yellow samples have been estimated covering all the configuration free space.

According all the possible connections, the local path is represented in black. However, the shortest path that connects the blue point with the green point is the purple path, which corresponds to the final trajectory.

After this application, we have seen an improvement in the computation time. PRM addition allows to get a new trajectory instead wasting time exploring the environment.

However, after some simulations it has been experimented that when the robot moves close to an object, PRM was continuously executed. So, it sometimes spends some extra time computing each time that minimum distance was satisfied.

As a solution to regulate this state, some changes have been applied. The goal is to reduce the computation time each time the robot considers that an object is quite near. To avoid recalculating the PRM new trajectory, the robot will check if the current desired trajectory is crossing an obstacle. In *Figure 32*, the purple line represents the current trajectory. However, the yellow circle shows the exact moment when the robot detects an obstacle that interrupts the path. After this moment, the PRM will compute a new set of waypoints to establish the new trajectory to follow.

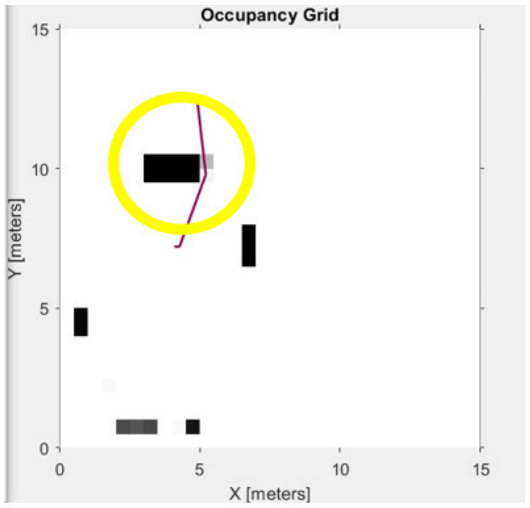

Figure 32. *PRM trajectory crossed by an obstacle*

If this condition is satisfied, the line equation between one waypoint and the following one is computed. Then, 100 samples (x and y pairs) are taken from each equation. For each of the computed samples, it is tested if it belongs to an object detected by LIDAR. Finally, if one of the samples belongs to an obstacle, PRM recalculates the path giving a new set of waypoints. In this way, the new trajectory to the goal is computed faster.

6. Final Results

In this chapter the results obtained from the diverse map performances will be evaluated. As the objective of this thesis is to develop a high quality robot performance, the results will be evaluated once all the enhancements have been included.

To start from the simplest map, the border map will be analyzed. For a given goal, the algorithm computes the Pure Pursuit trajectory from an initial point. As there are not obstacles around, the robot reaches the goal easily and the occupancy grid map is nearly empty. In *Figure 33* the results obtained from border map environment are represented.

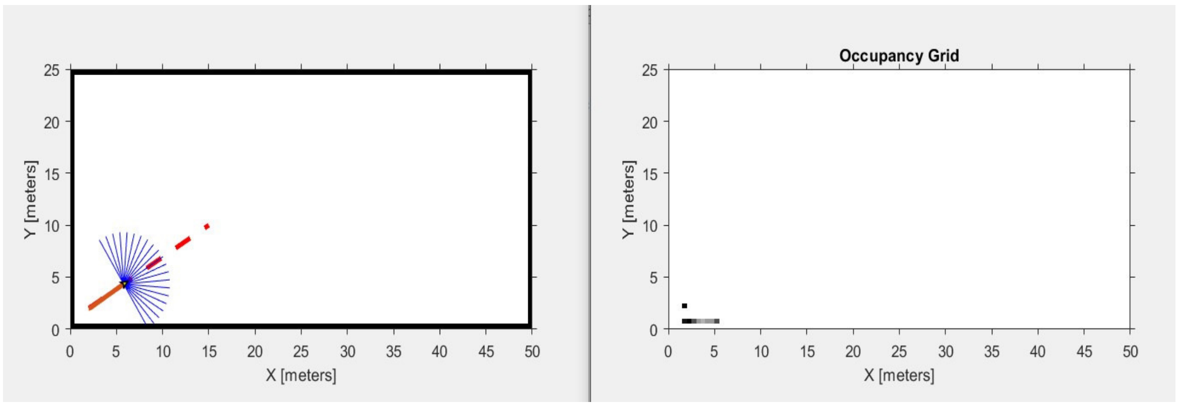

Figure 33. *Border map environment results*

In a way to understand in a visual form how the algorithm is designed, *Figure 34* shows how PRM really works. The trajectory is determined and the robot is following it (see *Figure 34 a) and b)).* It has been represented the current path over the occupancy grid to capture the moment that an obstacle that intersects the path appears (see *Figure 34 d)).* From this point, a new set of waypoints is obtained to define the new trajectory to get the goal as in *Figure 34 e).*

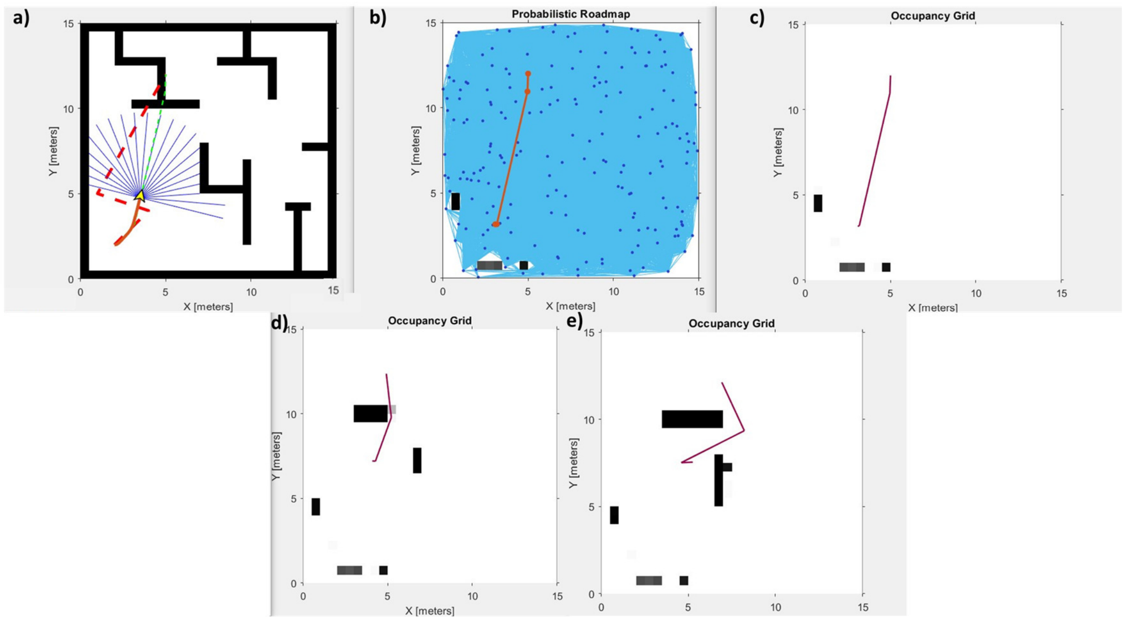

Figure 34. *Branches map results. a) Robot's behavior. b) Probabilistic Roadmap. c) Occupancy grid map. d) Trajectory intersected by an object. e) New trajectory computed.*

To increase difficulty of the environment, some square obstacles are included. *Figure 35 a)* is a good representation of the different trajectories computed. Red colour shows the initial trajectory stablished while orange colour references to the robot's real trajectory. The blue rays denote LIDAR measurements along the different angle sectors. As the robot found the first squared obstacle, the PRM planner is executed giving the orange waypoints represented in *Figure 35 b).* The trajectory established in *Figure 35 b)* is updated in colour green in *Figure 35 a)*. As we can observe the robot follows it properly till it gets the goal. In *Figure 35 c)* is represented the occupancy grid of the obstacles that the robot has seen. These obstacles are also updated in the PRM map to avoid giving waypoints that are objects.

The performance is progressive and the robot has been able to reach the goal all the times.

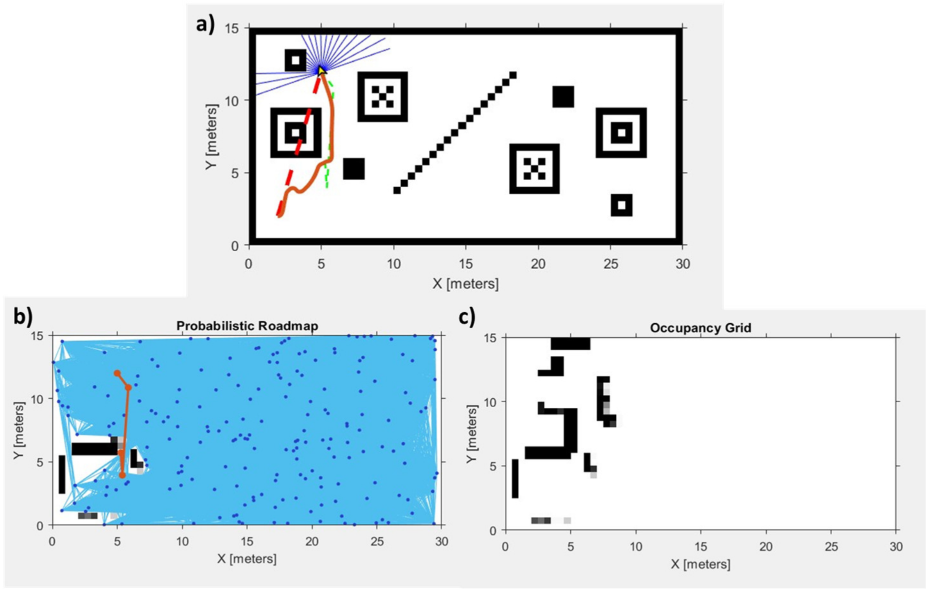

***Figure 35**. Squares map results. a) Robot's behaviour. b) Probabilistic Roadmap. c) Occupancy grid map*

However, let's evaluate what happens if we change the obstacle shapes and we ask the robot to cross the map. From the map represented in *Figure 36,* it is obtained a similar behaviour to the previous one.

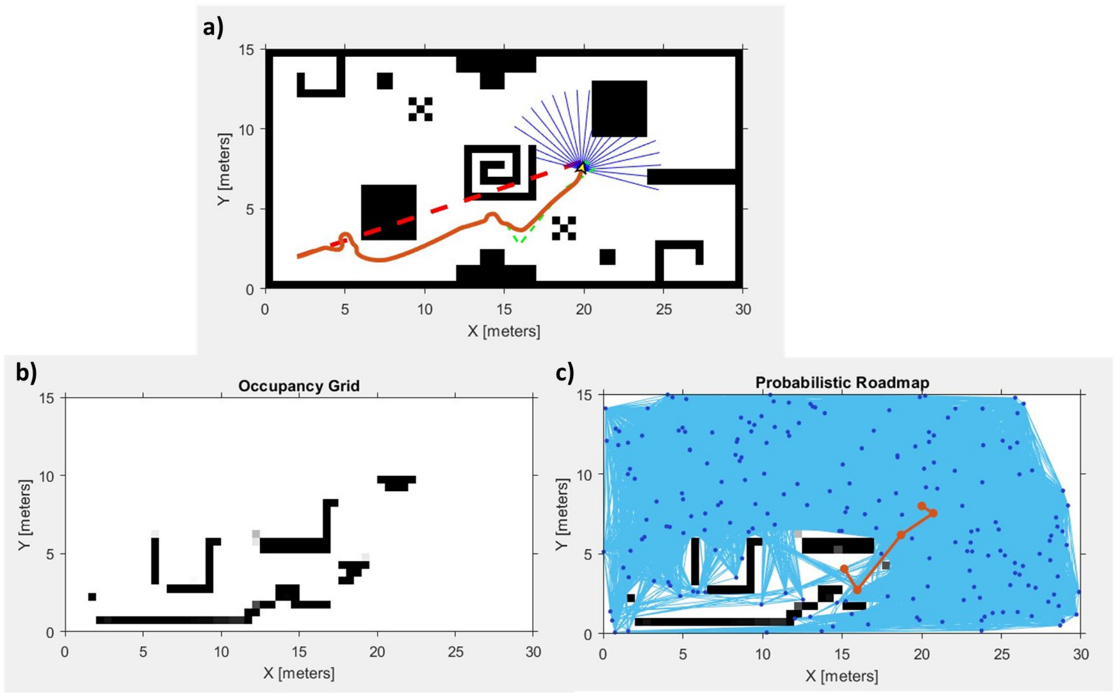

***Figure 36.** Obstacle map results. a) Robot's behavior. b) Occupancy grid map. c) Probabilistic Roadmap*

It has been demonstrated that the robot moves avoiding the objects and is able to compute a new trajectory. Now it will be evaluated in case of horizontal and vertical corridors. According the results presented in *Figure 37*, the robot satisfies the main aim. It presents no difficulties in detecting the different walls and establish a new trajectory to go around them.

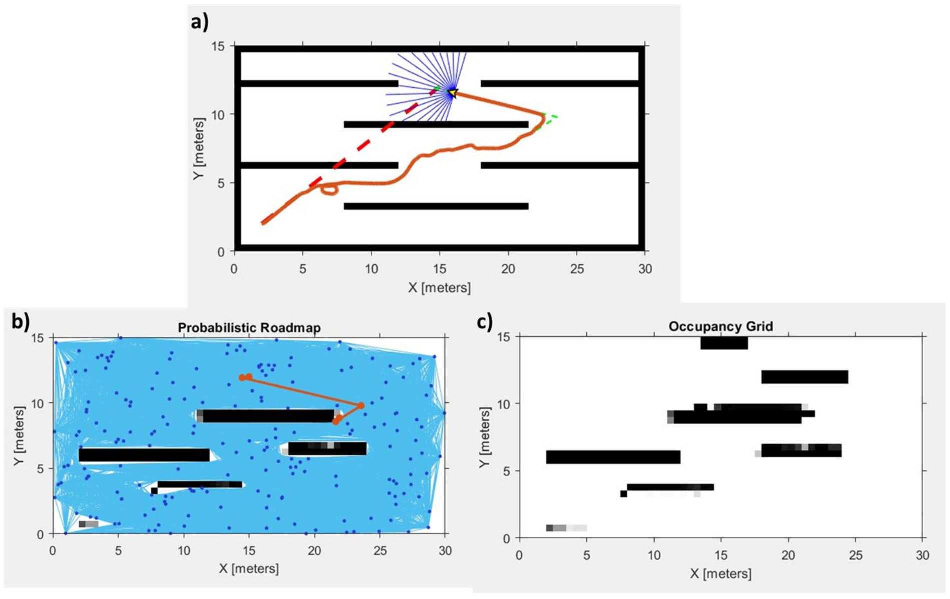

***Figure 37**. Horizontal map results. a) Robot's behavior. b) Probabilistic Roadmap. c) Occupancy grid map*

In the particular case of the vertical aisles, the robot presents a similar behaviour than in the horizontal case. The goal has been given in order to make the robot cross all the obstacles over the x-axis as it is observed in *Figure 38.*

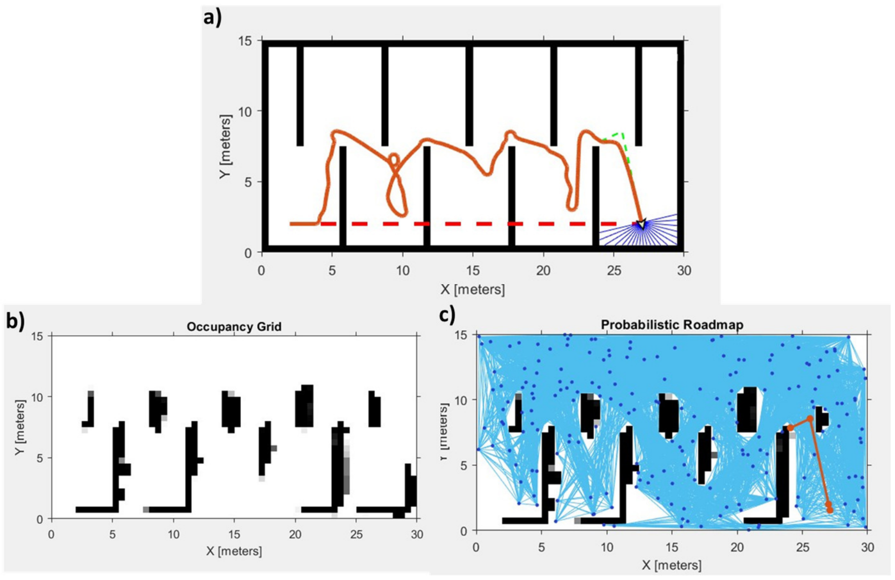

***Figure 38.** Vertical labyrinth map results. a) Robot's behavior. b) Occupancy grid map. c) Probabilistic Roadmap*

In this kind of situations, where the robot experiments continuous changes in the trajectory, the computation time due to the iterations increases considerably. Despite of getting the goal, we have observed that in these two last environments the robot is slower. It has to be taken into account that the walls in these examples were long. As the robot does not know its surroundings, it sometimes wastes time checking that the new unknown waypoints correspond to a free space.

On the other hand, in *Chapter 7: Future work* is considered to increase the occupancy grid obstacles in order to guarantee a safer robot's performance in case of continuous obstacles.

Reviewing the final results presented along this chapter, we can conclude that the goal of this thesis has been reached. The developed algorithm works quite well in different environments and it never has showed a crash. However, the speed of some performances could be studied as part of a future work to enhance the final performance time.

7. Future Work

As a future work, it would be interesting to have the chance of testing the algorithm in a real world frame. To develop this idea, it has been considered to list a specific set of components that could be part of the future AGV robot.

This thesis is focused on the computation of the navigation commands, so it is important to select the right path planner as we have seen along this project. At the same time, sensors have an important relevance: they are the responsible of collecting data samples. The information received is filtered and analyzed to estimate the obstacles around and their distance to. The principal set of sensors proposed are the following ones:

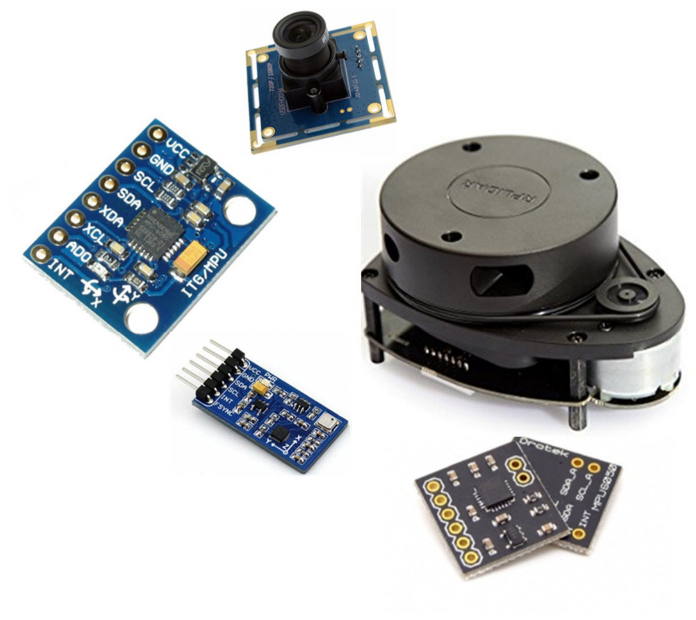

Figure 39. *Hardware elements*

- 180 degree fisheye lens 1080p.
- RPLIDAR A1M1, 360 degree 2D Laser Scanner (LIDAR) system.
- RPLIDAR A2
- IMU 10DOF, composed by:

- MPU-6050 gyro and accelerometer. Motion Processing Unit. 9-Axis sensor fusion integrated using its field-proven and proprietary MotionFusion™ engine for handset and tablet applications, game controllers, motion pointer remote controls, and other consumer devices.
- HMC5983 magnetometer. Multi-chip module designed for low-field magnetic sensing with a digital interface for applications such as low-cost compassing and magnetometry.

- S5611- 01BA altimeter. High resolution altimeter sensors from MEAS Switzerland with SPI and I^{2}C bus interface. It is optimized for altimeters and variometers with an altitude resolution of 10 cm.

The selection of the different sensors is based on the characteristics of the environment and the algorithms implemented.

In addition, as a safety measurement, it has been considered to inflate the obstacles represented in the occupancy grid. The main reason is to guarantee that the robot will not be damaged. If the obstacles are inflated, the robot will be considering an extra safety distance range that protects the robot to be crashed into.

Furthermore, as it has been detailed in *Chapter 6: Final Results,* it can be studied the behavior presented at vertical and horizontal aisles, to accelerate the performance of the algorithm in these kind of environments.

8. Budget

The aim of this chapter is to provide a brief economic analysis in case of introducing the thesis to the commercial reality.

COMPONENT DESCRIPTION	ESTIMATED COST (€)
RC Car	150,00 €
IMU 10DOF: • MPU6050 • HMC5983 • MS5611	24,90 €
360-degree Laser Scan	190,00 €
RPLIDAR A2: New Generation 360 Degree 2D Laser Scanner	878,00 €
USB Port	7,00 €
Arduino nano	16,29 €
64 Gb Emmc Linux	57,50 €
Wifi Module3	6,00 €
Odroid-XU4	59,00 €
Creative BlasterX Senz3D	154,31 €
180-degree Fisheye Lens 1080p Camera	45,00 €
15v to 5v Regulator	15,00 €
Additional features	80,00 €
Software licenses (Matlab, Graphical Environment,..)	2.500,00 €
Simulation tests	1.000,00 €
Algorithm development	50.000,00 €
TOTAL	**55.183,00 €**

Table 2. *Project cost budget*

This thesis is a study work, so we have to think about the idea of a company wants to make an offer to acquire the AGV prototype.

In *Table 2* are summarized both the cost of each of the components that constitute the autonomous ground vehicle, and the costs related to the software development, software licenses and simulation. However, the project presented along this document is supposed to be enhance as it was presented in *Chapter 7.* For this reason, it is considered to be more expensive as more hours and improvements are dedicated to.

9. Environmental and social impact

Nowadays, when technological projects are developed, it is important to take some time to think about how the project would affect the environment and the society. In this chapter, a brief analysis of these aspects is presented.

9.1. Environmental impact

More and more automotive companies are taking advance in the autonomous driving field. The current application of the autonomous driving is combined with electric or hybrid cars, which seems to be an idea focused on the emission reduction. Most of the driverless vehicles being driven (and test-driven) today are already fully electric.

According to research from the Department of Energy, automated vehicles could reduce energy consumption in transportation by as much as 90%. Autonomous cars use significantly less gas and energy when driving, compared to a vehicle driven by a human. Most gas is burned when driving at high speeds, braking, and re-accelerating excessively. This kind of vehicles cut these factors out of their driving style, meaning less gas is burned, or battery power consumed, resulting in less air pollution.

Autonomous cars also are expected to mean fewer vehicles per household. The idea simple, the same vehicle can drop the kids off at school, take the parents to work, and then park itself until it is time to pick them back up. In this way, one autonomous car could satisfy all the destinations you need, so families can cut back on having more extra cars to fit the trajectory needs of each person.

Another relevant point is their weight. It is also expected that the autonomous car weight would be lighter, so they would have a positive effect on combustible consumption.

9.2. Socioeconomic impact

Autonomous cars are not on the road yet. However, many companies are testing their prototypes in this technological area. For this reason, the most affected is the automotive industry. Despite of the disruption of car-sharing services may reduce private car ownership dramatically, the vehicle market will likely expand as the car distance travelled increases in a broad scope of users who are enabled by autonomous driving to enjoy the convenience of automobiles without physically driving them. The trucking and freight industry will be among the earliest adopters of automotive driving as companies seek to improve transport efficiency. According to that, in the first phase of change, this kind of cars will be implemented in controlled environments and maybe in some commercial transport tasks.

Another point is the reduction of car accidents. Around the 90% of car crashes are caused by human errors. The number of accidents is supposed to decrease once these cars are implemented in our daily lives. Demand for car repair services, medical services, and traffic police will decrease too as a consequence.

Furthermore, the most critical factor is how long would take the people to adopt this disruptive technology. Despite all the benefits presented in this section, it is hard to the society get used to the idea of saving jobs and delegating some human tasks (as driving, paying attention on the road, avoiding an obstacle that could suddenly appear…) to an automated vehicle.

So, who knows when we start to use them?

Conclusions

Along this project different algorithms have been evaluated taking into account the quality of their performance and the problems found. The basic concept idea was the performance of Follow-the-carrot algorithm. However, the simplicity of this algorithm makes it weak and less precise than expected. It has been experimented a predisposition to cut the corners as a consequence of turning towards the new goal each time it changes. Additionally, the robot tends to oscillate about the path when the speed is increased or when the look ahead distances have a short value.

As a consequence, it has been looked for a robust algorithm that follows Follow-the-carrot principle. At this point, Pure Pursuit Algorithm has been considered. It shows a good performance that satisfies the goal proposed: from one initial point, be able to stablish a set of waypoints to follow in order to reach the goal. The algorithm has a complex part in comparison with Follow-the-carrot algorithm due to it is based on pure pursuit curve to compute the trajectory to follow. As a result, using Pure Pursuit algorithm the heading angle errors are considerably reduced, the trajectory is smoother and the robot shows a higher accuracy when curves appear along the trajectory path.

To complement the path planner, an obstacle avoidance algorithm has been introduced to allow our robot to identify block spaces, and as a consequence, to detect the free space area which it can move along safely.

At this stage Potential Field Algorithm has been taken into account. The reason was simple; the algorithm interprets the idea of different direction forces acting on the robot. According if it is an obstacle or is the goal, they will have distinct charge value (positive or negative), responsible of the repulsive or attracted movement of the robot. From the resulting force, the robot will be guided by induction to the goal, showing a repulsive performance when obstacles or walls appear around its position over the path.

However, the real problem occurs when the algorithm is implemented. It can be easily observed in its performance that at some closed walls or narrow shapes the robot can be stuck in some points that belong to local minimums because it has no way to elude them.

For this reason, this project needs a robust method and Virtual Force Field Algorithm has been contemplated.

Virtual Force Field Algorithm merges the application of the histogram grid and the potential field algorithm in real-time. Each sample data is used to update not only the histogram grid, also the resulting force vector. In this way, the vehicle can perform a faster response at the moment that obstacles suddenly appear.

Nevertheless, there still exist chance of being enclosed in a local minimum. To solve the local minimum issue, it is presented the wall following method, which consists on guiding the robot around the obstacles when a local minimum is detected by the estimation of a new virtual attractive force.

Once implemented, the performance was not as expected: at narrow and large corridor environments, the robot lost the control according to the repulsive forces vary continuously from the nearest wall, which causes a side-to-side oscillation on the robot that makes impossible to get good results.

The next improvement was the implementation of Vector Field Histogram Algorithm. At this point the robot develops a real-time model of its environment where cells with a higher chance of an obstacle are represented with greater certainty values. Through the information obtained from the certainty cells located in the active window, the polar histogram is developed. Polar histogram represents in a 360-degree circle around the robot the polar obstacle density. Then, the reference values for the robot's drive and steering controllers are obtained. So, the new direction of motion is determined according to the candidate direction that better guides the robot towards the goal.

Despite of obtaining an accurate performance with VFH algorithm, Vector Field Histogram + algorithm was implemented to check if there was some difference or enhancement over the results obtained.
One of the main differences is that VFH + takes into account the robot's radius to compute the first primary polar histogram. From this histogram, active cells are enlarged to allow that it counts for the robot's width. Not only considers robot's radius, also takes into account a safety distance.

Now, the histogram sectors are computed to be either open or blocked space according the polar obstacle densities. To continue, a mask is applied to the binary polar histogram taking into account the different trajectories that the robot can follow. Under this constraint, the available steering directions the algorithm can select are reduced. Finally, the algorithm determines which is the best direction to develop the robot's steering based on the possible directions available determined along all these stages.

Once path planning and obstacle avoidance algorithm were observed working together, we can confirm that results are satisfactory: the AGV is able to reach the goal from an initial point following the trajectory defined by the waypoints, at the same time LIDAR sensors allow it to detect the obstacles and move away from them. If the goal is behind of an obstacle and the trajectory defined by the waypoints crosses it, the robot will follow the path till the moment it detects that the way is blocked. At this point, the robot analyzes which is the best direction to continue in order to reach the goal without crashing.

It has been experimented, even though the AGV works nice well, the robot sometimes spends more time than usual in determine the right direction. As one of the aims of this thesis is to try to obtain optimal results, it has been considered to add a PRM planner to improve the performance when obstacles are in a relative close distance. When the object is detected for the established minimum distance, PRM is displayed evaluating a set of n samples. According to that, the new waypoints are calculated in order to stablish the new trajectory till reaching the goal. In this way, the exploration time is reduced and the robot's performance is faster. However, an additional enhancement was considered. It was no necessary to compute again the trajectory if the obstacle is really close. It just only needs to take into account that the trajectory does not cross the obstacle.

As a final conclusion, taking into account all these previous steps, the iteration computing time is considerably reduced at the same time the algorithm has a good performance working in a faster way.

Bibliography

[1] B. I. K. A., “Lidar Sensor for Autonomous Vehicles,” Technische Universität Chemnitz, Chemnitz, 2017.

[2] S. M. La Valle, Planning Algorithms, Illinois: Cambridge University Press, 2006.

[3] M. Lundgreen, *Path Tracking for a Miniature Robot,* Sweden: Umea University, 2003.

[4] R. Craig Coulter, “Implementation of the Pure Pursuit Path Tracking Algorithm,” Camegie Mellon University, Pittsburgh, Pennsylvania, 1992.

[5] Y. Koren and J. Borenstein, “Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation,” University of Michigan, Sacramento.

[6] J. Ni, W. Wu, X. Fan and J. Shen, “An improved VFF Aproach for Robot Path Planning in Unknown and Dynamic Environments,” *Mathematical Problems in Engineering,* vol. 2014, p. 10, 2014.

[7] J. Borenstein and Y. Koren, “The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots,” *IEEE Transactions on Robotics and Automation,* vol. 7, no. 3, 1991.

[8] J. Borenstein and Y. Koren, “Optimal Path Algorithms for Autonomous Vehicles,” 1987.

[9] . J. Borenstein and I. Ulrich, “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots,” in *1998 IEEE International Conference on Robotics and Automation*, Belgium, 1998.

[10] A. Benitez and D. Vallejo, “New Technique to Improve Probabilistic Roadmap Methods,” Universidad de las Américas Puebla, México.

[11] R. Garaerts and M. H. Overmars, “A Comparative Study of Probabilistic Roadmap Planners,” Institute of Information and Computing Sciences, Utrecht University, Utretch, 2006.

[12] L. E. Kavraki, M. N. Kolountzakis and J.-C. Latombe, “Analysis of Probabilistic Roadmaps for Path Planning,” *IEEE Transactions on Robotics and Automation,,* vol. 14, no. 1, pp. 166-171, 1998.

[13] J. S. Wit, Vector Pursuit Path Tracking For Autonomous Ground Vehicles, Florida: Defense Technical Information Center, 2000.

[14] F. A. Salem, “Dynamic and Kinematic Models and Control for Differential Drive Mobile Robots,” Taif University, Taif, 2013.

[15] L. Feng, Y. Koren and J. Borenstein, “A Model-Reference Adaptive Motion COntroller for a Differential-Drive Mobile Robot,” University of Michigan, Michigan.