eman ta zabal zazu

Universidad del País Vasco — Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

**MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL**

**TRABAJO FIN DE MÁSTER**

# ***PATH PLANNING AND TRAJECTORY OPTIMIZATION FOR AN AUTONOMOUS CAR CONTROLLED BY MODEL PREDICTIVE CONTROL***

| | |
|---|---|
| **Alumno/Alumna** | *Iñigo Campino Díez* |
| **Director/Directora** | *prof. Angelo Bonfitto* |
| **Departamento** | **LIM, Politecnico di Torino** |
| **Curso académico** | *2018/2019* |

*Bilbao, 10, octubre, 2019*

# PATH PLANNING AND TRAJECTORY GENERATION IN AN AUTONOMOUS CAR CONTROLED BY MODEL PREDICTIVE CONTROL

Author: Iñigo Campino Díez
Supervisor: Prof. Angelo Bonfitto

2019

POLITECNICO DI TORINO

Politecnico Di Torino
Master's in industrial engineering. Final Thesis.

“It's the questions we can't answer that teach us the most.

They teach us how to think. If you give a man an answer, all he gains is a little fact.

But give him a question and he'll look for his own answers.”

Patrick Rothfuss

## Acknowledgements

Podría escribir multitud de hojas de gratitud y reconocimientos de apoyo en mi camino hasta este punto, pero eso es algo que llevo conmigo, y demuestro día a día. Mis padres, mi familia, ese apoyo es incondicional, y no solo ha de estar reflejado su reconocimiento en estas líneas de gratitud, ya que mi carácter ha sido forjado por dicho reconocimiento. Esa gratitud camina conmigo todos los días.

Si ha de quedar constancia de mis apoyos académicos durante mi estancia en Torino,  gracias al profesor Bonfitto por permitirme trabajar en este proyecto y a Irfan por ayudarme técnicamente con el trabajo siempre de buena gana y con una sonrisa.

De todas maneras, realizar una tesis de 30 ECTS durante tu estancia en el extranjero, hace que tenga unos enfoques diferentes y haya llegado a este punto, de maneras que no hubiera imaginado. Es por ello que tengo que agradecer a toda esa gente que hacía que mi día a día fuese algo no solo bueno, sino que fuese diferente. Y no solamente en el ámbito académico-profesional, más bien en el personal e interpersonal.

Y eh, Laura Blanco, casi consigues que me levante pronto algún día, por eso también estoy agradecido.

## Abstract

The autonomous driving vehicles are getting more and more importance in automotive and control engineering, and definitely in society. Lots of works and thesis have been performed in recent times about their control, the image processing, lane keeping, environment simulation and vehicle modelling, but actually path planning is not of very common study in this increasingly large field of engineering. This thesis proposes a solution to optimizing trajectories with a path planning algorithm which makes this easier to implement in different vehicle environment without mixing with other parts of the control or the lane detection. In summary, a modular part for the autonomy of the vehicle which has as objective to optimize the trajectory the vehicle is going to follow and being able to provide it to the lateral and longitudinal control of the vehicle.

Los coches de conducción autónoma esta cobrando más y más importancia en la industria de la automoción y la ingeniería de control, y por consiguiente en la sociedad.  Muchos trabajos y tesis se han realizado en los últimos tiempos sobre su control, procesamiento de imagen y modelización vehicular, pero el tema de planificación de trayectorias no ha sido demasiado objeto de estudio en este creciente campo de la ingeniería. Este proyecto propone una solución a la optimización de trayectorias con un algoritmo de planificación que además permite que dichas trayectorias se desarrollen sin interferir con otras partes del control o de la detección del entorno. En resumen, un módulo independiente para optimizar la autonomía de los vehículos y las trayectorias a seguir, además de ser capaz de ser leída e utilizada como entrada para el control lateral y longitudinal del vehículo.

I veicoli a guida autonoma stanno diventando sempre più importanti nell'ingegneria automobilistica e di controllo, e anche nella società. Diversi lavori e tesi sono stati eseguiti in tempi recenti sul loro controllo, elaborazione delle immagini, mantenimento della corsia, simulazione ambientale e modellistica dei veicoli, ma in realtà la pianificazione di percorsi e traiettorie non è uno studio molto comune. Questa tesi propone una soluzione per ottimizzare le traiettorie con un algoritmo di pianificazione del percorso che rende questo più facile da implementare in diversi ambienti del veicolo senza mescolarsi con altre parti del controllo. In sintesi, una parte modulare per l'autonomia del veicolo che ha come obiettivo di ottimizzare la traiettoria che il veicolo sta per seguire. Inoltre deve essere in grado di fornirgli il controllo laterale e longitudinale del veicolo.

Auto autonomoak gero eta garrantzi handiagoa hartzen ari dira automobilgintza eta kontrol ingeniaritzan, eta, beraz, gizartean. Azkenaldian lan eta tesi ugari egin dituzte kotxe hauek kontrolatzeko, kameren irudien tratamendurako eta ibilgailuen modelizazioaz, baina traiectora eta bideen plangintza ez da asko aztertu ingeniaritza arlo honetan. Proiektu honek traietorien eta bideen optimizaziorako soluzio bat proposatzen du, plangintza algoritmo batekin. Gainera kalkulatutako bide hauek ingurumena kontrolatzeko edo detektatzeko beste elementu batzuekin interferitu gabe garatu behar izan dira. Laburbilduz, ibilbideen autonomia autoetan optimizatzeko modulu independente bat garatu da informazio hau irakurri eta kontrolak ondo interpretatu ahal duena.

# Table of contents

## List of figures

# 1. Introduction

This thesis is explaining how the work with autonomous cars controlled by MPC has been developed during my stay and collaboration with the interdepartmental mechatronics laboratory in the polyethnic university of Turin (Politecnico di Torino). All the assumptions, theories, resources and tools are going to be explained and deconstructed in order to understand how the improvements and implementations have been done, with which aim and how they are going to be useful in a future.

First in this introduction chapter, the main framework and objectives of this engineering field, in general, and the tools for the improvements made in this thesis, in particular, are going to be described. In recent years a lot of researches have been focused on autonomous driving and, for this reason, an overview of the this has been done at the beginning of this work. The objectives, the reach of the project, the software used and the main concepts to understand the work afterwards developed are written here and in the next section where the problem is stated as well as the thesis objectives.

In the third section, all the tools used in the thesis, as well as the models and analytical statements that cover the whole work made in the laboratory by the whole team are explained, in order to understand in what point of the work this thesis is framed, and to expose how the overall control diagram works.

Also, this is useful to understand the theoretical problem, which has been stated, and analyzed in section 2. Also, the state of the art for this problem, how have other communities developed solutions is going to be clarified, in order to have contrast with the solution adopted. Here the whole work environment is going to be explained. The different principal parts are being separated and analyzed, in order to understand why the solutions adopted are the optimal for the circumstances.

The structure of the MPC, its optimization problem and its working principles are specified too. In addition, the simulation model is also explained to understand the verification systems and its work.

In the fourth and fifth chapter, the solution and its implementation for the path planning are written and explained. The analytical solution of the problem that is adopted and its implementation in the software. The algorithm development, the different versions carried out and the software model and block diagrams are also shown.

Fifth chapters continue after the path planning as the assimilation of it and explains the transformations, improvements and changes that are applied in order to be useful in the control closed loop. It can be assumed that this chapter explains the trajectory generation based on the path planning algorithm.

The bibliography, all the referenced books, papers and theses, are noted and referenced at the end of the document.

## 1.1. Autonomous cars

### 1.1.1. Concept

An autonomous car, also known as a robotic car, self-driving car, or driverless car is a vehicle that is capable of sensing its environment and moving with little or no human input, in contrast with classic vehicles that need a human driving in order to fulfill their mission.

Modern vehicles provide partly automated features such as keeping the car within its lane, speed controls or emergency braking. Nonetheless, differences remain between a fully autonomous self-driving car on one hand and driver assistance technologies on the other hand.

### 1.1.2. Advanced driver-assistance systems (ADAS)

ADAS are electronic systems that aid a vehicle driver while driving. When designed with a safe human-machine interface, they are intended to increase car safety and more generally road safety.

As it is said before, most road accidents occur due to human error. Advanced driver-assistance systems are systems developed to automate, adapt and enhance vehicle systems for safety and better driving. The automated system which is provided by ADAS to the vehicle is proven to reduce road fatalities, by minimizing the human error. Safety features are designed to avoid collisions and accidents by offering technologies that alert the driver to potential problems, or to avoid collisions by implementing safeguards and taking over control of the vehicle.

Additional inputs are possible from other sources separate from the primary vehicle platform, such as other vehicles, referred to as Vehicle-to-vehicle (V2V), or Vehicle-to-Infrastructure (V2X), such as mobile telephony or Wi-Fi data network systems.

Next-generation ADAS will increasingly leverage wireless network connectivity to offer improved value by using car-to-car (also known as Vehicle to Vehicle, or V2V) and car-to-infrastructure (also known as Vehicle to Infrastructure, or V2X) data.

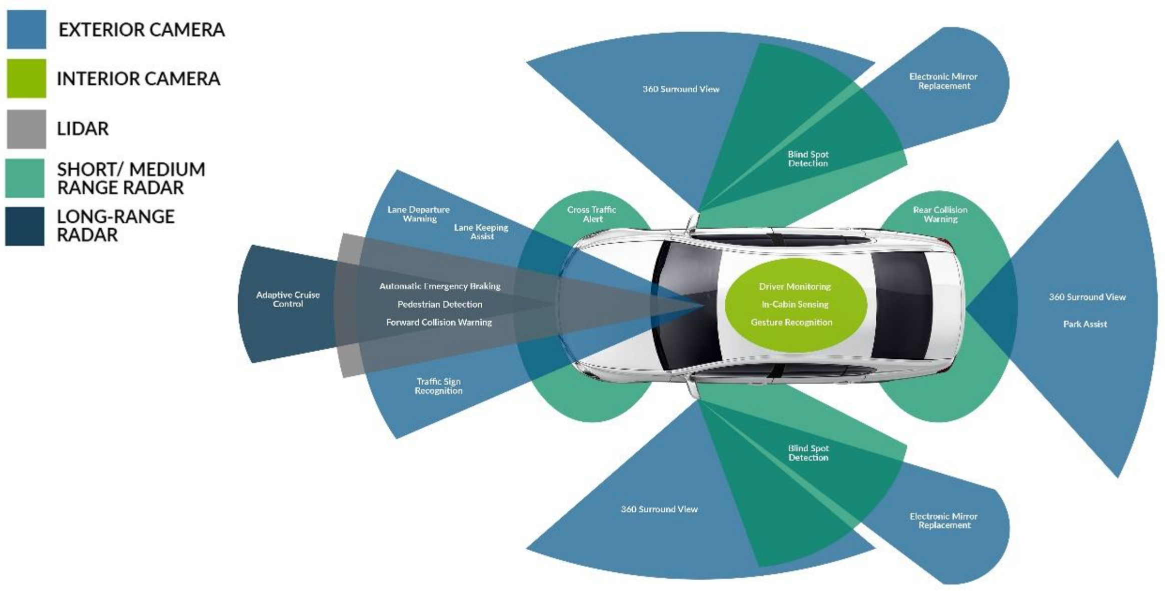


*Figure 1. Some examples of market ADAS*

### 1.1.3. Autonomous Car Classification.

Now that it is defined that autonomous driving cars can be considered vehicles that perform the transportation task without the human intervention (using algorithms executed by an on-board computer to simulate the behavior of the driver and make decision) and the ADAS are known, a classification is going to be made. Multiple classification could be done, according to the ADAS different types, but the one that is going to be used goes further, classifying by the autonomy degree. This one is the one made by the Society of Automotive Engineers in 2014.

The SAE's International classification system has the aim of describing the progression of the automation of vehicles, as shown in Figure 3. The United Nations and the US Department of Transformation have adopted SAE J3016 guidelines, that is today considered the industry standard.

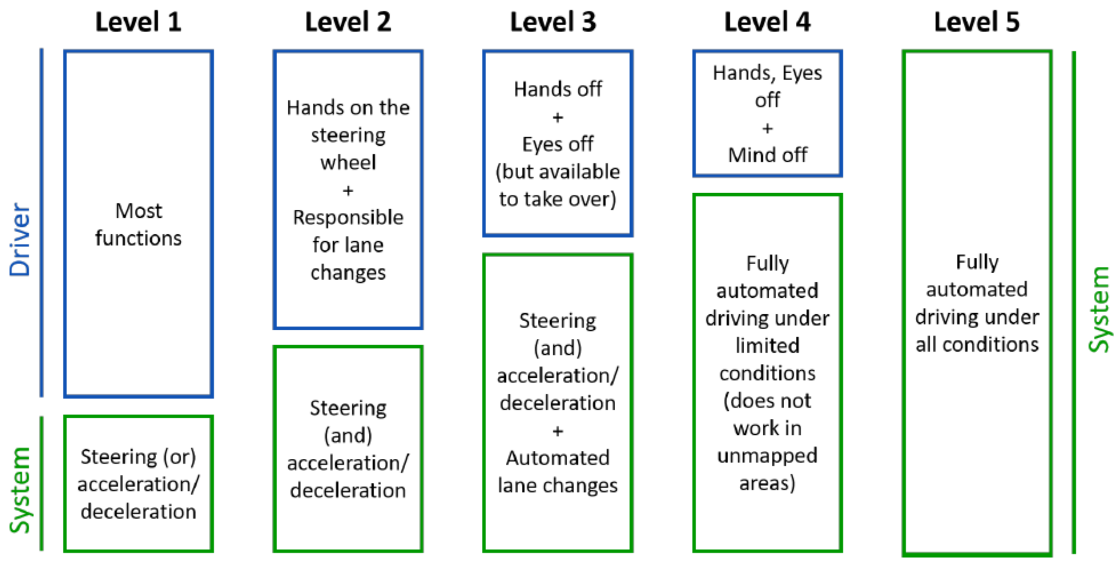

Level 1
Level 2
Level 3
Level 4
Level 5
Driver
System
Most functions
Steering (or) acceleration/ deceleration
Hands on the steering wheel + Responsible for lane changes
Steering (and) acceleration/ deceleration
Hands off + Eyes off (but available to take over)
Steering (and) acceleration/ deceleration + Automated lane changes
Hands, Eyes off + Mind off
Fully automated driving under limited conditions (does not work in unmapped areas)
Fully automated driving under all conditions
System

*Figure 2. SAE's classification of autonomous vehicles*

The classification is based on the amount of responsibility and attentiveness required by the driver. Six levels are defined accordingly, from a situation in which everything is controlled by the human (Level 0) to the full automation of the vehicle under all driving conditions (Level 5).

The definition of each level considers the specific role played by the driver, the driving automation system and by other vehicle systems and components that might be present. SAE's levels are descriptive and informative rather than normative, and technical rather than legal, they clarify the role of the ADAS which are progressively included in the vehicles.

The six levels can be defined as:

• Level 0 - No automation: steering or speed control may be momentarily assisted by the vehicle, but the human driver is in charge of all the aspects of driving;

• Level 1 - Driver assistance: longitudinal or lateral support under well-defined driving scenarios (e.g. highway) are guaranteed, because the vehicle takes over either the speed or the steering control on a sustained basis;

• Level 2 - Partial automation: both speed and steering control are taken over by the vehicle, therefore continuous longitudinal and lateral support under well-defined driving scenarios are guaranteed. A Level 2 vehicle is equipped with a wider set of ADAS;

• Level 3 - Conditional automation: the vehicle becomes capable of taking full control under well-defined driving scenarios, but the driver must be always in the condition of suddenly taking back control when required by the system;

• Level 4 - High automation: human interaction is not needed anymore; the vehicle takes full control and complete a journey in full autonomy under limited driving scenarios. Pedals and steering wheel are likely to be still present to guarantee the possibility to drive in scenarios that go beyond the defined uses cases (e.g. off-road);

• Level 5 - Full automation: the vehicle takes full control under all driving scenarios, no more provisions for human control are present. The concept of journey will be disruptively innovated, the entire vehicle design revolutionized.

## 1.2. Thesis Motivation

1.25 million people die and as many as 50 million are injured in road traffic accidents worldwide every single year according to United Nations statistics. 95% of all road traffic accidents in the EU are assumed to be due to human error, and in 2017 alone, 25,300 people died on the Union's roads.

A Morgan Stanley study concluded that autonomous cars could save the US $1.3 trillion a year. These earnings would be split between fuel savings from better traffic management, the lowered health care expenses due to the reduction of the number of road kills, and the productivity gains from spending less time driving. But these are maybe, the less important facts about the improvements that a complete autonomous driving system in our society would overcome.

On the social side, autonomous cars could allow elderly and disabled people to gain back some mobility (, the elderly or the physically challenged have little or no access to individual mobility) and reduce the need for parking space in cities as well. Every year half a million metric tons of CO2 emissions in Germany alone could be saved by eliminating the endless search for a parking space, which studies show account for up to 30% of inner-city traffic.

A 2016 study by the American Automobile Association (AAA) Foundation for Traffic Safety estimated that the average American motorist was driving around 50 km a day. In a week, this can represent several hours that cannot be used for leisure purposes. By delegating the driving task to an autonomous vehicle, these hours would be freed, resulting in a higher quality of life.

Another benefit of autonomous vehicle hitting the mainstream market is inter-vehicle communication. Nowadays, one human driver can only communicate with its peers using simple tools (directional signals, braking signals, etc.). Autonomous vehicles could take advantage of a wireless communication system using a frequency band (IEEE 1609 family) and use traffic optimization algorithms to reduce traffic jams. This is commonly referred as cooperative driving and theoretical algorithms have been proposed to implement this feature. The conclusions include a 33% reduction of total travel times and a significant reduction of traffic jams, which would have a positive environmental impact since it has been estimated that reducing them could lower the carbon dioxide emissions by 20%. Car manufacturers have realized the potential of cooperative driving for autonomous vehicles and imagine future applications for their new products.

## 1.3. Work Environment

Now that the main engineering field in which the thesis is framed, lets study more locally which one is the framework of this thesis. In order to understand how the main architecture and bases of the model used, the analytical previous resources and the knowledge acquired, the work environment must be explained.

The Mechatronics Laboratory LIM (Laboratorio Interdipartamentale di Meccatronica, 2019) at the *Politecnico di Torino* is an interdepartmental structure founded in 1993 as a "joint-venture" by a number of people of the Departments of Control and Computer Sciences (DAUIN), Electronics and Telecommunications (DET) and Department of Mechanical and Aerospace Engineering (DIMEAS) of the *Politecnico di Torino*.

*Figure 3. Laboratorio Interdisciplinare di Meccatronica*

Its main objective is to constitute a common ground where researchers and postgraduate students working in the mechatronic field could perform theoretical and experimental research and exchange experiences in a true interdisciplinary environment.

LIM activities are organized in terms of projects where different disciplinary expertise's are integrated to yield the best overall performance. During the last years, several students and researchers in the LIM have been working on the Autonomous Cars environment, including simulations and experimental challenges in their control.

This is the environment where this thesis is framed. The past models, control solutions, simulations, software, codes and other material that other thesis students and collaboratives are used in a teamwork way in order to keep implementing improvements and make the current material more efficient and productive.

More specifically, this thesis is framed is the Autonomous cars field and its control, in which some Thesis students and PhD's are also working as a team. The main objectives and work of this thesis is going to be explained next, as well as how has the research and improvements have been done.

## 1.4. Software - MATLAB

For all the thesis the software the software used is MATLAB®. MATLAB (matrix laboratory) is a numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, the plot of functions and different data, implementation and development of algorithms, creation of user interfaces, and interfacing with programs written in other languages. It is perfect for a multidisciplinary project as this.

*Figure 4. MATLAB and MathWorks Logos*

In order to make the different models it has been used Simulink, developed by MathWorks, which is a graphical programming environment for modeling, simulating and analyzing multidomain dynamical systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it, as it will be shown afterwards.

For this thesis some MATLAB toolboxes have been used. Toolboxes are collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. The ones that have been mainly used are the following:

### Model Predictive Control Toolbox

Model Predictive Control Toolbox™ provides functions, an app, and Simulink blocks for designing and simulating model predictive controllers (MPCs). The toolbox lets you specify plant and disturbance models, horizons, constraints, and weights. With this toolbox the MPC of the line tracking is implemented in the model.

### Robotic System Toolbox

Robotics System Toolbox™ provides algorithms and hardware connectivity for developing autonomous robotics applications for aerial and ground vehicles, manipulators, and humanoid robots. Toolbox algorithms include path planning algorithms, who's some of them have been used in certain parts of the Path Planning development.

### Automated Driving Toolbox

Automated Driving Toolbox™ provides algorithms and tools for designing, simulating, and testing ADAS and autonomous driving systems. With this toolbox the construction of the Simulink model and scenario data interpretation have been done.

### Driving scenario designer

With this toolbox the scenarios in which the model of the vehicle is moving have been designed and implemented into the overall model. It has been also configured vision and radar sensors mounted on the ego vehicle and use these sensors to simulate detections of actors and lane boundaries in the scenario.

### Computer Vision System Toolbox

The design and simulate computer vision and video processing systems is made using Computer Vision Toolbox. It may be used if the image processing is needed for the lane keeping algorithm. Computer Vision System Toolbox™ provides algorithms, functions and apps for the design and testing of video processing systems, machine vision and 3D vision. It is possible to carry out object detection and tracking, as well as detection, extraction and feature matching.

### Image processing toolbox

Image Processing Toolbox™ provides a complete set of standard reference algorithms which will also be used in the lane keeping blocks too, as well as workflow apps for image processing, analysis and visualization (it is also including an algorithm development, but it won't be used in this thesis). You can carry out image segmentation, image enhancement, noise reduction, geometric transformations, image registration and 3D image processing.

### Geom2d module

The geom2d module of the *MatGeom* library allows to process geometric planar shapes such as point sets, edges, straight lines, bounding boxes, conics (circles and ellipses)... Most functions work for planar shapes, but some ones have been extended to 3D or to any dimension.

# 2. Problem Statement

Designing controllers that are able to steer the autonomous car at its limits of handling will decrease the amount of potential road kills, thus increasing the public confidence in autonomous vehicles. A convenient setup to evaluate this type of controllers is the context of this thesis, which is a scenario where an autonomous car needs to complete the track following the trajectory where maximum progress is made.

Past thesis and researches have been done constructing a control model regarding the different main parts needed to fulfill the conditions of autonomy. The autonomous car team of the LIM is working on the development of **Lane detection** and **Lane keeping** functions. These functions allow an autonomous driving vehicle to recognize the lanes of the road where the car is driving and follow a specific trajectory generated with information of the road environment. The control of the car in order to fulfil certain conditions and keep the obtained trajectory as good and accurate as possible would be the final objective always.

The environment catchment and road lectures are going to be represented as the lane detection parts. This part is the one in charge of obtaining the vehicle's environment data needed for its driving and processes it for, then, using it to plan the trajectory.

In order to plan the trajectory, the next block is the **Trajectory generation**, which receives the environmental road information and computes by the ways of path planning, a trajectory that the vehicle should follow. This trajectory generation also computes the trajectory in terms of the control needs (lane keeping). Also, the trajectories computed, and its curvatures are used by the Reference Velocity profile generator to compute the maximum reference velocities the vehicle should adopt, which are another control input.

Once the trajectory is computed and clear next block represents the control of the vehicle. For the lateral and longitudinal control of a vehicle, which are the main motions to control the general motion of the vehicle, the Model Predictive Control theories are being used in several projects and thesis of this engineering field. Regardless if the object of study is a real vehicle or a simulation environment, the MPC is the main control theory for autonomous vehicles for lane or trajectory keeping. The controller's function is to make the vehicle follow the line as smooth and accurate as possible. From the controller, the vehicle inputs are obtained, which are sent to the real vehicle or to a simulation environment, which will compute the vehicle state (position, velocities and accelerations between others) that is feedbacked to the velocity profile generator and also to the controller.

Several studies and works analyze the different Lane Detection and Lane Keeping problems, but the Trajectory Generation is always of diffuse study. This is because the trajectories are not optimized at all several times and are commonly simply computed as the center of the two-lane boundaries of the lane detection. When comes to trajectory optimization, the main solutions adopted are including control layers and complexity to the optimization problem computed by the model predictive control. In *Autonomous racing using model predictive control* (Curinga, 2018) and *Optimization-based autonomous racing of 1:43 scale RC cars* (Liniger, 2014) the proposed control includes more constraints in the MPC, which will adopt as inputs the road boundaries and have as outputs the vehicle control parameters. This would mean that the trajectory generation will be completely done by the MPC, in addition to the lateral and longitudinal control.

This solution is optimal for a racing context, but not flexible at all. If the control is bounding the following trajectory to the optimal, the control model proposed will just not be able to drive in other scenarios like urban, suburban, highway, or even in obstacle avoidance ones.

A solution to this problem, is to separate the control of the trajectory generation. In this way, even if not optimal for racing, the adaptability will be part of the path planning. Different algorithms will be able to just generate different kind of reference trajectories, which will be the MPC's input (and not just the lane boundaries, in case it computes the vehicle movement according to optimized trajectories). In this way, the control will remain unaltered, and the trajectory generation should be the one adapting to the different contexts and environments. This way would mean a modular separation between the trajectory optimization (or generation) and the lane keeping, with all the advantages that come with it.

## 2.1. Thesis Objectives.

The overall autonomous driving system has been implemented with MATLAB and Simulink, but the model that is currently been used will be explained afterwards in more detail. One of the bases and first objectives of the work done has been to understand the theoretical concepts and the implementation of the whole control model and lay-out of the autonomous vehicle environment. However, the main objective of this thesis will be focused on the Path planning and it subsequent Trajectory Generation.

As in past works the trajectories generated were simple, improvements are crucial. The objective is to be able to optimize the generated trajectories, and not being only able to compute the road's center line.

For this, the aim is to develop an algorithm that is able to compute and create the path that the car should follow for each step of the MPC, and afterwards compute a trajectory on terms, not only, of space but with time and velocities. This path should be the optimal, or at least make sensitive improvement in comparison to the one computed in previous works (center line trajectory) whenever this trajectory (optimized and not center line) is needed. The developed algorithm should be able to generate the center line or the optimized trajectory without affecting the lateral and longitudinal control regardless the regime it is on, so making it adjustable depending on the driving regime.

It can be said so, that the principal objective of the thesis is to implement a path planning algorithm in the current model substituting the simple algorithm that is working now. A subsequent objective is to be able to parametrize the trajectories needed by the controller under the conditions of the computed paths, velocities and environment information.

Improvements are going to be made in the other two parts (Control and verification) due to necessities that will appear to make the path planning work as it should. For that other algorithms and enrichments for the model are going to be analyzed and implemented. It can be stated as a secondary objective to improve as much as possible the current model to make it easier to use, smoother and the most efficient as possible. This secondary improvements are going to be explained when the environment is analyzed in section 3.

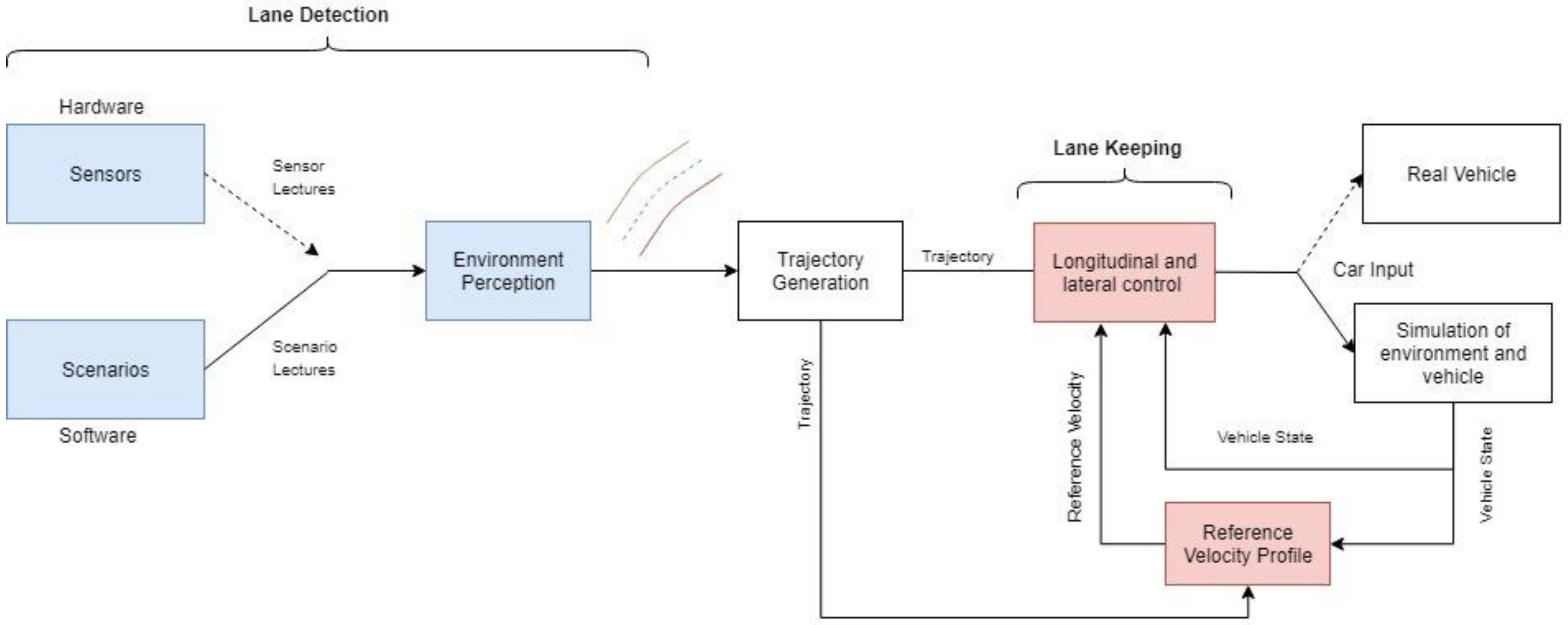


*Figure 5. Basic Control Diagram*

# 3. Autonomous vehicle environment

Now that the thesis has been introduced and the problem stated, a deeper analysis on how the full control lay-out of the autonomous vehicle have been implanted and how it is working is going to be explained. This is of high importance because the deep understanding of the overall model and lay-out (and the full control diagram) is needed to understand the stated the problem and posterior solution adopted in this thesis.

As mention in section 1.4, the overall system has been implemented in MATLAB and Simulink. In the representation of the autonomous vehicle environment it could be seen how the diverse parts of the main diagram are differentiated. However, the Figure 6's diagram is just a simple and indicative one, in order to differentiate the main problems. Henceforth, the control diagram that will be used is the one below.

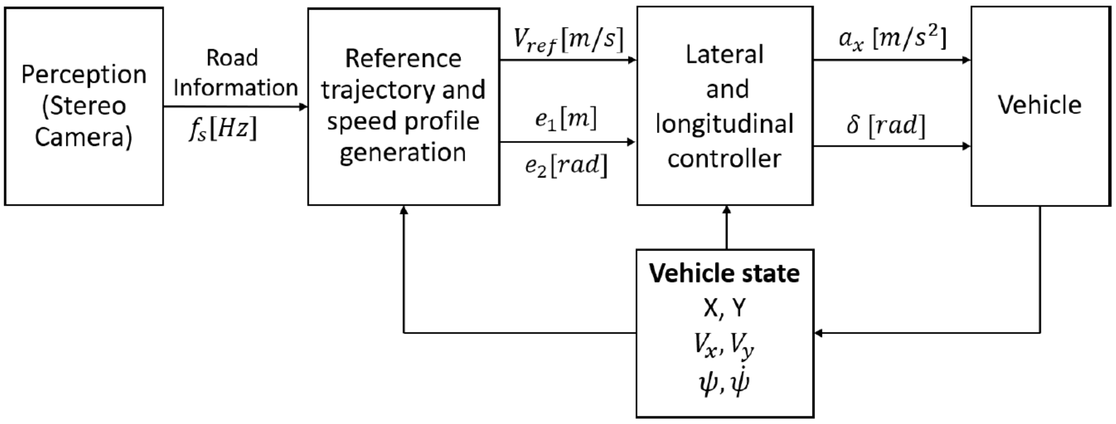


*Figure 6. Overall architecture of control strategy*

This overall control diagram starts with the environment perception. Its implementation in this thesis is explained in the first part of this section as the lane detection solutions. Once the road information is computed for each time step, the second block is focusing on the reference trajectory generation and in the velocity profile generation. As the Reference trajectory generation for the controller is going to be developed in the next section as the adopted solution for the problem, it will not appear in this one. Nevertheless, the speed velocity generation will also be briefly introduced in this section.

The lateral and longitudinal control and the vehicle which is feedbacking the vehicle state are that are being used in the study of this thesis are also being explained and exposed in this section as the lane keeping and the vehicle modelling for validation.

## 3.1. Lane detection

Lane detection is a well-research area of computer vision that allows to realize functions for ADAS and autonomous vehicles. In particular the detection of road lanes could be done in two particular ways. The first one, as it will be in the real vehicle is with image processing of the sensor. In this particular case, and as second option, the road information comes from the simulation scenarios.

Both data are in need to be interpreted in order to be of a certain use in order to control the autonomous vehicle in a coherent way with its environment. This data is computed and interpreted, so it can be converted into, for this particular case, coordinate point arrays for each boundary of the road.

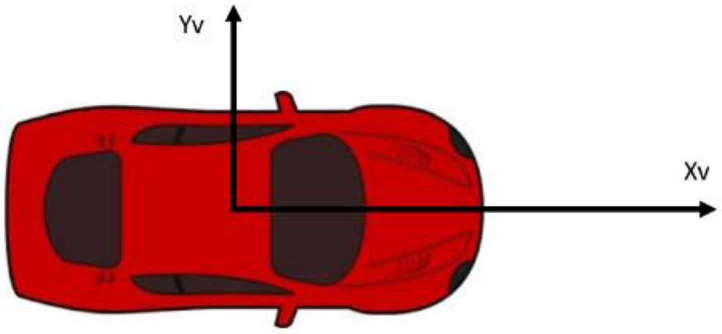


*Figure 7. Vehicle Local Reference system*

For now on, the coordinates in which the lane boundaries and scenarios will be represented, will be referred to the local vehicle reference system, or EGO reference system which will be always located in the car center of gravity of the vehicle, which will be its origin of coordinates.

As shown in the figure 7, the longitudinal axis of the car will correspond to the x-axis, and the lateral one with the y-axis. In other words, the x-axis is set in the direction in which the vehicle moves; the y-axis is set perpendicular to the x-axis and points to the left side of the vehicle.

### 3.1.1. Lane Detection by camera

In particular the detection of road lanes can be performed following a visual perception example included in the MATLAB documentation that uses Automated Driving System, Computer Vision System and Image Processing toolbox. All this method is explained more deeply in the second chapter of Study and implementation of lane detection and lane keeping for autonomous driving vehicles (Mancuso, 2018). The main steps for this road boundary identification are the camera calibration, the region of interest (ROI) extraction and the inverse perspective Mapping (IPM) in order to arrive to the lane detection as it is understood in this thesis.

The first step is the calibration of a camera. It is needed in order to compute the extraction of ROI and IPM. Camera calibration performs the computation of intrinsic and extrinsic parameters. They allow to convert the coordinates information of images from world to pixel coordinates as show in Figure 8. This pixel coordinates are the ones needed to obtain the ROI.

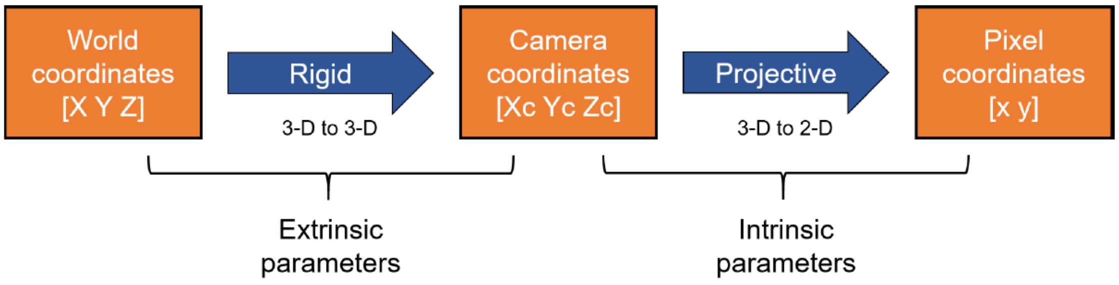


*Figure 8. Coordinate conversion for camera-based lane detection*

The extraction of Region of Interest (ROI) consists in select only the relevant part of the image which includes the lane of the road. In fact, the definition of a Region of Interest gives the possibility to exclude light poles, pedestrians, trees and vehicles that are foreign voices in the lane markers detection. Moreover, set a ROI allows to reduce the detection range in order to decrease the surrounding noise and the computational cost due to the processing time. In the system developed for Mancuso's thesis, the Region of Interest has been computed in a geometric way. It consists in select the relevant area in front of the vehicle to send to the function that transform the image in the bird's-eye-view as show in the Figure 10.

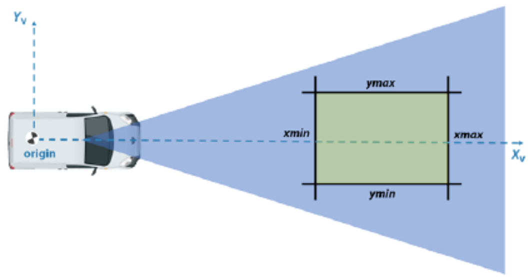


*Figure 10. Region of Interest selected for bird's-eye-view image transformation*

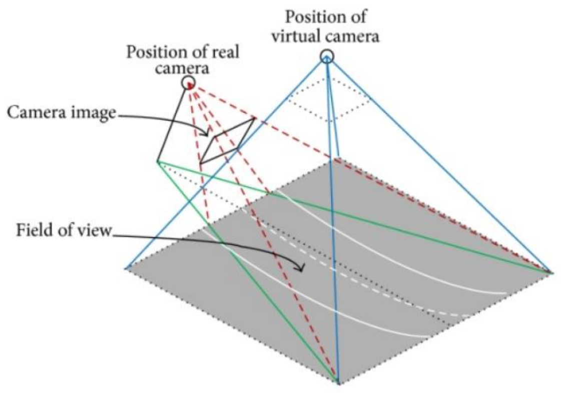


*Figure 9. Schematic illustration of the conversion from the real position of the camera to the virtual position*

The last step is Inverse Perspective Mapping (IPM). This method has been used to transform a real image coming from a camera into a bird's eye-view image that allows to have a top view of the road. In order to have the new view, the real position of the camera is converted into a new virtual position, which is the desired one according to the local vehicle reference system, and the coordinates obtained are coherent with the EGO Car position. (Shown in Figure 9)

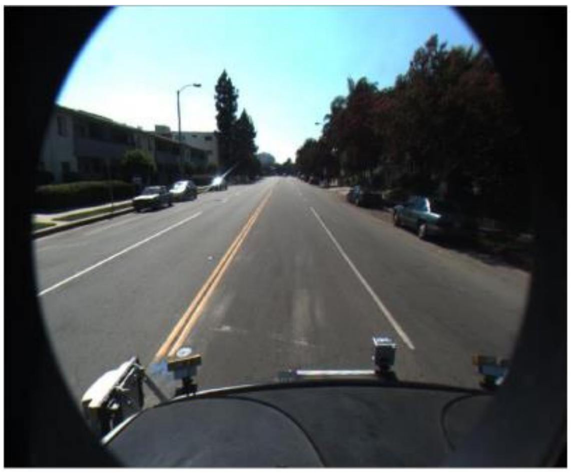


Figure 11. a)Original Image from a car's camera.

In the Figure 11. And 12. are shown how from the original image (a) the Inverse perspective mapping (IPM) transformation gives as a result (b). After the extraction of feature is made, which is used to compute relevant information in a image and representing them in the least possible space (that will be no further explained because it is no theme for this thesis) giving as a result c. The final lane model is given in d.

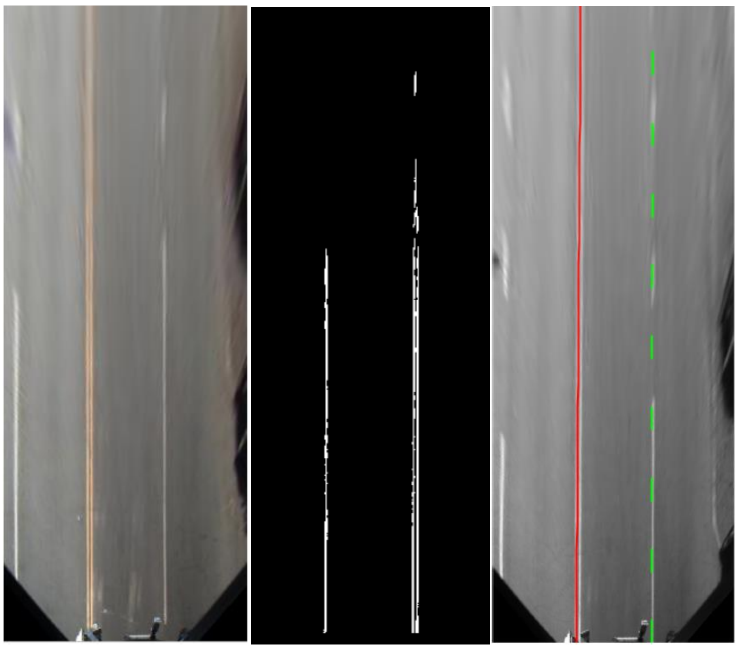


Figure 12. b) IPM c) Extraction of feature d) Final lane line model

The final result is the obtention of the left and right boundaries as coordinate points in the local reference system (modified to be in coherent with the CoG of the car, which is the coordinate system).

### 3.1.2. Scenario Lectures in simulation environment.

In this thesis the whole model and environment is implemented in Simulink, so the obtention of the lane boundaries will come from the driving scenarios designed for simulation with the *Driving scenario designer toolbox* instead of coming from the camera. Nevertheless, the results obtained from both sides are equal in format, so can be both used in the same way for what regards to the rest of the thesis. This whole part of scenario lectures is included into the Environment and vehicle simulation Block in Figure 6.

This toolbox allows the user to design simulation adaptable environments or scenarios. As an example, the ones designed and exposed in *Model Predictive Control based lateral and longitudinal controller for autonomous driving* (Khan, 2019) are attached in the Figure 12 to make a deeper understand of the kind of data the toolbox provides. With the toolbox different kind of road tracks can be created, in which the different road lanes and its properties are modified and adapted to the track needed. Also obstacles can be included, as well as other road driving vehicles, but they are not going to be included in the contents of this thesis.

Also, as it will be explained afterwards, it provides curvature of the scenarios, the obstacles coordinates, the yaw angle and lateral offset of the EGO coordinates with the left and right boundary, which will be useful in future sections. This information is the one that must be used by the trajectory generation in order to link itself with the lateral and longitudinal control.

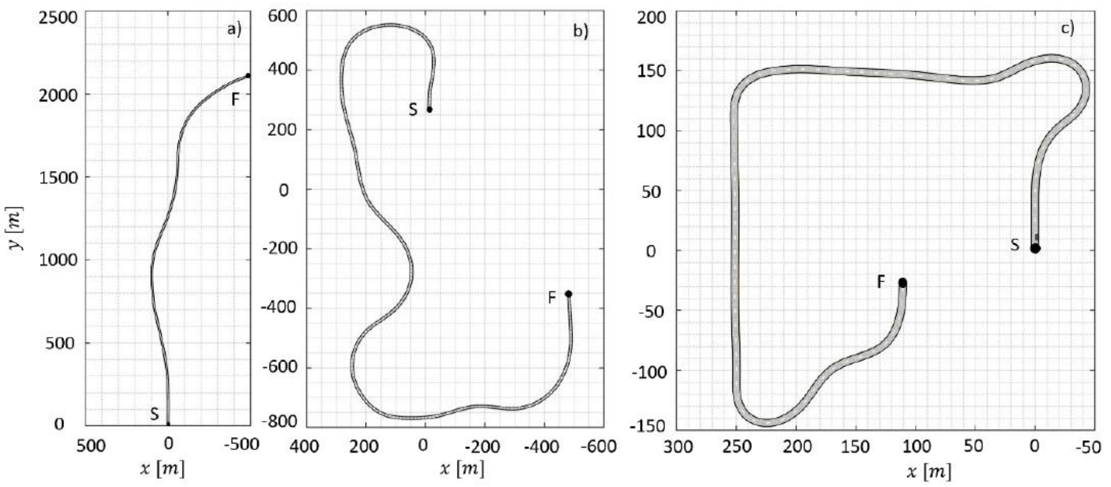
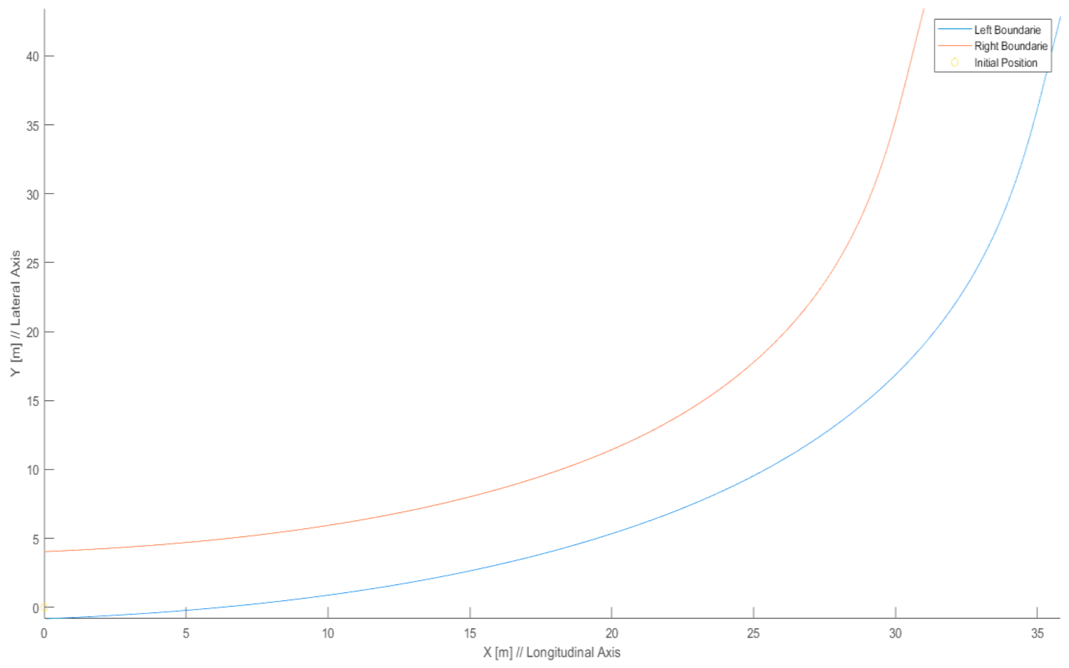


*Figure 13. The driving scenarios: a) highway; b) inter-urban; c) urban. S is the vehicle's starting point. F is the end of the road track.*

For these scenarios, as it is also shown in the Figure 12, a global reference system is adopted. So, for now on, the global reference system will be the one of the scenarios, and the local one will be the one described above. As it seems reasonable, the local reference system will be moving along with the CoG of the car and rotated with its longitudinal axis. The vehicle coordinates in the global reference system will be provided to the toolbox algorithm, in addition to the vehicles longitudinal velocity and its yaw angle and rate. That way, a recognition of the position and state of the vehicle will allow the toolbox to compute the coordinate points of the left and right boundaries according to the vehicle state.

Every time step of the whole model's computation will give new coordinates of boundaries with the cars current state (of the current time state). In the Figure 13 can be seen two examples of a time step's lectures shown in a bird's eye plot. Coherently, the CoG of the vehicle is each plot's origin of coordinates.



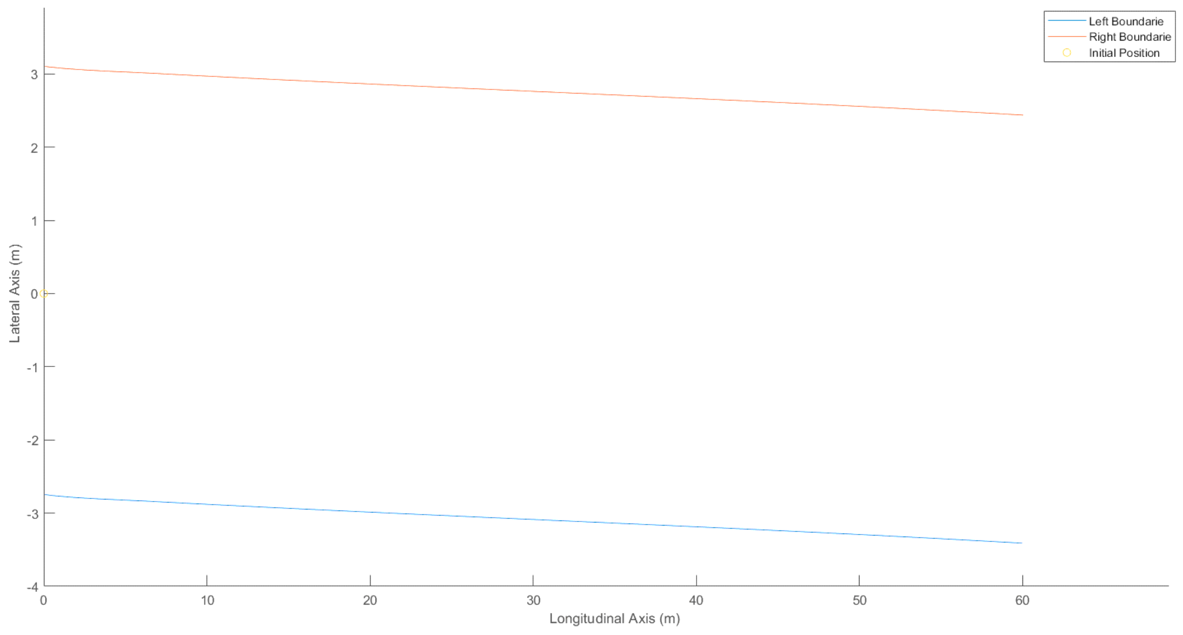


*Figure 14. Driving scenario's lecture example of a curve leftwards and straight line.*

## 3.2. Lane Keeping

In this section, the general knowledge of the control theory is shown to explain the reason why autonomous driving controllers use this theory to develop their functions such as lane keeping. Firstly, the main ideas behind the Model Predictive Control and the architecture used for autonomous driving vehicle are presented. After that, the controller implemented for this thesis work is explained. In the rest of the section, the velocity profile generation is explained. The global architecture of the proposed control strategy is given by Figure 16 as it the control is implemented in *Model Predictive Control based lateral and longitudinal controller for autonomous driving* (Khan, 2019). It serves as a more accurate diagram on how the control is working in input/output terms than the general one of the Figure 6.

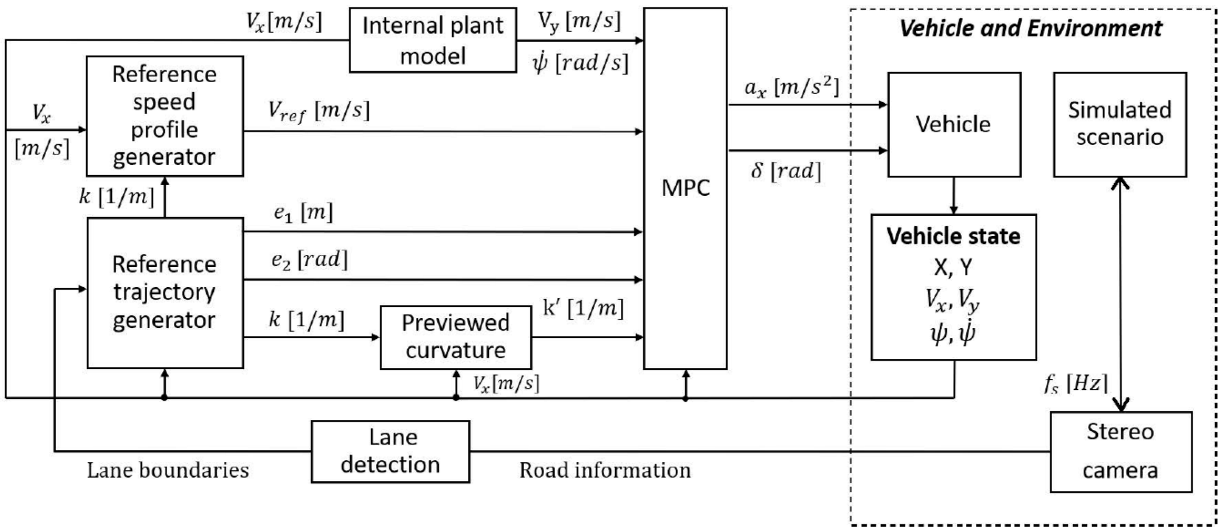


*Figure 15. Detailed architecture of control strategy*

It highlights the main inputs and outputs for the controller and the vehicle blocks. The control technique exploits a stereo camera that utilizes the synthetic data coming from the simulated driving scenario (A simulated driving scenario is used to simulate the environment and generate the synthetic data required for the control algorithm, as mentioned before in section 2.1,) for reference trajectory and speed profile generation. Using these reference profiles, the controller act on throttle/brake and steering angle to control the vehicle. Vehicle states are provided each time step to the controller and the reference trajectory and speed generator block to perform the closed loop simulation.

As the lane detection that is converting the road information into lane boundaries has been explained, and the reference trajectory generator (that is in charge of generating the trajectory the car should follow, as a reference in the model predictive control) will be exposed as the problem’s adopted solution, there are three control parts that are going to be elucidated. These three parts are the Internal Plant Model (vehicle model for the MPC), the Reference speed profile generator and the MPC structure. The general vehicle model will be explained in the last subsection (3.3.) of this section. Referring to the overall model (Figure 16) the MPC block provides information about the lane keeping control block, or longitudinal and lateral control of the vehicle to follow the stablished trajectory. This block has the aim to give the value of the front wheel steering angle and vehicle’s throttle by controlling the values of curvature, lateral deviation and relative yaw angle coming from the previous step.

### 3.2.1. Model Predictive Control basics.

Model Predictive Control theory for the implementation of the lane keeping will be mainly analyzed. Model Predictive Control (MPC) is an advanced technique developed to automate industrial control processes by meeting a series of operating constraints.

According to A *Survey of Industrial Model Predictive Control Technology* (Qin & Badgwell) , the overall conceptual objectives of an MPC controller are:

- Prevent that input and output constraints are violated;
- Optimize some output variables, while other outputs are kept in a specified range;
- Prevent that the input variables have excessive movement;
- Control the major number of process variables when a sensor or actuator is down or is not available.

The MPC controller provides the optimal output to send to a plant based on a finite horizon using an iterative approach. Its main goal is to calculate a sequence of control moves, that consist of manipulated input changes, so that the predicted output moves to the set point in an optimal manner. In the case of study of this thesis the MPC should receive as inputs the trajectory and the vehicle state in order to optimize the adaptation of the car to the given trajectory in the predicted horizon giving as output, as previously said the steering angle and the throttle.

Figure 17 shows simply how the MPC theory works. $y$ is the actual output, $\hat{y}$ the predicted output and *u* consists of the manipulated input. At the current sampling time *k*, the initial value of the plant state is known and the MPC computes a set of *M* values of the input *u(k+i-1)*, *i* = 1, 2, ..., *M*, where *M* is called *control horizon*. This set refers to the current input *u(k)* and to (*M* - 1) prediction inputs, and it is held constant after the *M* control moves. The inputs are computed so that a set of *P* predicted outputs $\hat{y}$*(k + i), i = 1, 2, ..., P* reaches the set point in optimal manner. *P* is called *prediction horizon* and consists of the number of future steps to look ahead.

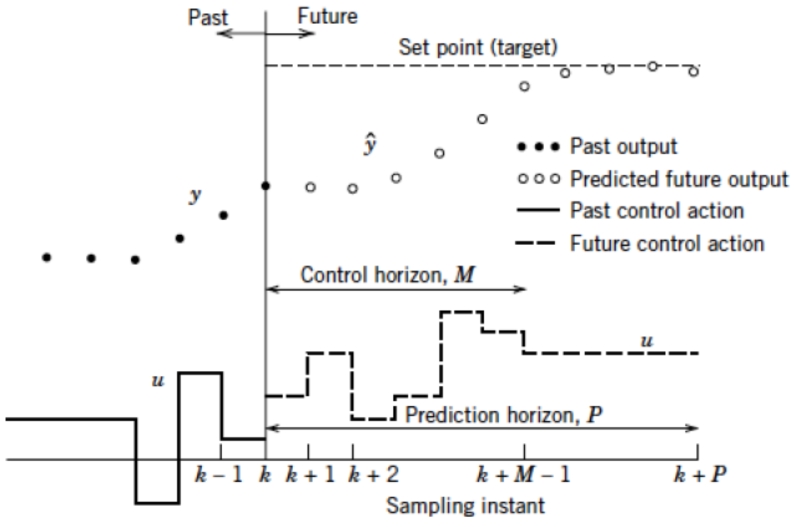


*Figure 16. Basic concepts of MPC*

The whole MPC controller that is going to be used in this thesis is obtained from *Model Predictive Control based lateral and longitudinal controller for autonomous driving* (Khan, 2019).

### 3.2.2. Linearized vehicle model for MPC

As the Internal Plant Model, a 2 degree of freedom vehicle model is used in order to define the lateral dynamics of the vehicle for controller internal plant model in terms of error with respect to the reference trajectory.

The two errors are lateral displacement error $e_1$, which is defined as the lateral distance between center of gravity of vehicle and the center line of the reference trajectory. Yaw angle error $e_2$ is defined as the difference between the yaw angle of the vehicle and desired yaw angle as dictated by the reference trajectory, as represented in Figure 17. The rate of change of lateral displacement error and yaw angle error are given by the equations in Equation 1, and also the desired yaw angle rate.

$$\dot{e_1} = V_x e_2 + V_y$$
$$e_2 = \psi - \psi_d \qquad \dot{\psi}_d = \frac{V_X}{R}$$

Equation 1.

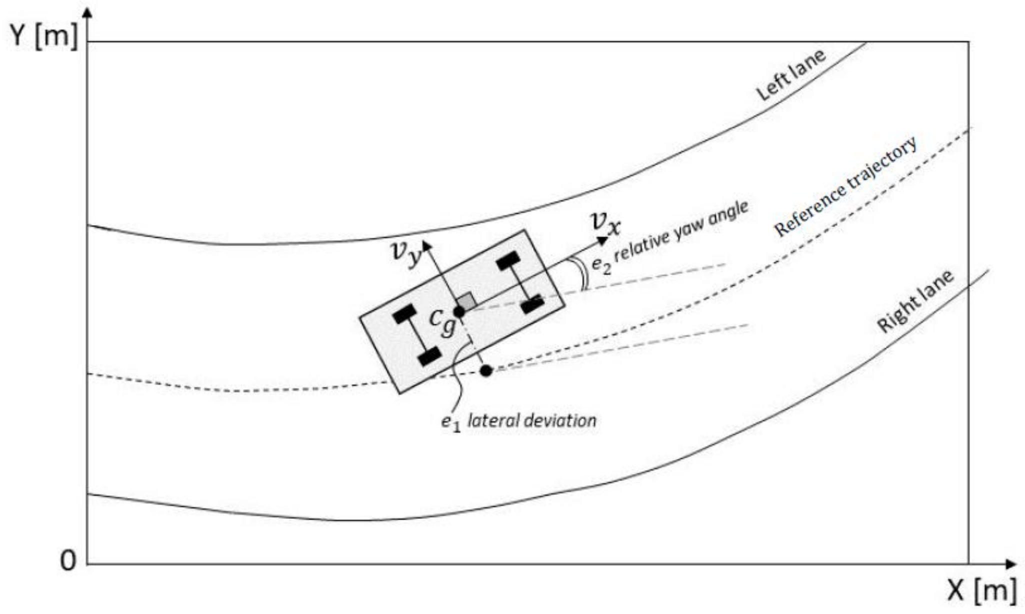


Figure 17. Lateral dynamics vehicle model of MPC

For the longitudinal dynamics, the plant model used for control design is the transfer function between desired acceleration and actual vehicle speed and is given by (Tau is the time constant):

$$P(s) = \frac{1}{s(\tau s+1)}$$

Equation 2.

The plant model used as the basis for adaptive MPC is an LTI discrete-time, state-space model with a sampling time $T_s = 100ms$. The combined state space model for lateral and longitudinal dynamics which is used as the internal plant model for MPC is represented in Figure 18.

$$x(k+1) = Ax(k) + B_u u(k) + B_d v(k)$$
$$y(k) = Cx(k) + D_d v(k)$$

k — Time index (current control interval).
x — Plant model states.
u — Manipulated inputs. These are the one or more inputs that are adjusted by the MPC controller.
v — Measured disturbance inputs.
A — It is the state matrix.
$B_u$ *and* $B_d$ are the input matrices corresponding to inputs u and v respectively.
C — It is the output matrix.

*Figure 18. LTI state-space model of the vehicle for the MPC*

The inputs are separated to indicate that u correspond the steering angle and acceleration command of the vehicle (it is the controlled input), while v indicates the longitudinal velocity multiplied by the curvature ▯(it is the disturbance). The output y corresponds to the lateral deviation $e1$, relative yaw angle $e2$and velocity of the vehicle $Vx$. In the state vector, $Vy$ denotes the lateral velocity, $Vx$ denotes the longitudinal velocity and $\psi$ denotes the yaw angle. The final and complete model of the vehicle for the MPC is the one below:

$$\frac{d}{dt}\begin{bmatrix} \dot{V}_x \\ V_x \\ V_y \\ \dot{\psi} \\ e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \frac{-1}{\tau} & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & -V_x - \frac{2l_f C_{\alpha f} - 2l_r C_{\alpha r}}{mV_x} & 0 & 0 \\ 0 & 0 & -\frac{2l_f C_{\alpha f} - 2l_r C_{\alpha r}}{I_z V_x} & -\frac{2{l_f}^2 C_{\alpha f} - 2{l_r}^2 C_{\alpha r}}{I_z V_x} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \mathrm{V_x} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{V}_x \\ V_x \\ V_y \\ \dot{\psi} \\ e_1 \\ \mathrm{e_2} \end{bmatrix} + \begin{bmatrix} \frac{1}{\tau} & 0 \\ 0 & 0 \\ 0 & \frac{2C_{\alpha f}}{m} \\ 0 & \frac{2l_f C_{\alpha f}}{I_z} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ \delta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\mathrm{V_x} \end{bmatrix} [\rho]$$

| Parameter | Value | Unit |
|---|---|---|
| $m$ | 1575 | $[kg]$ |
| $I_{zz}$ | 2875 | $[kg \cdot s^2]$ |
| $a$ | 1.2 | $[m]$ |
| $b$ | 1.6 | $[m]$ |
| $C_{\alpha f}$ | 19000 | $[N/rad]$ |
| $C_{\alpha r}$ | 33000 | $[N/rad]$ |

*Figure 19. Vehicle Model for MPC and simulation vehicle parameters*

### 3.2.3. Reference velocity Generator

In order to determine the reference speed profile, two different criteria are considered here, which are available in literature. First one is based on the geometry of the road and the second one is based on the lateral comfort of the vehicle. Based on the road information, the maximum speed is calculated using the curvature of the road, given by:

$$V_{max} = \sqrt{\frac{g\mu}{k}}$$

*Equation 3.*

Where g, μ and k are gravity acceleration, friction coefficient and road curvature respectively. While, based on human comfort experiments in *An experimental study on lateral considering human factors* (Xu, Yang, & Shao, 2007) the absolute value of lateral acceleration is limited by velocity dependent constraints as:

$$|a_{y,comfort}| = a_{y,o}\left(1 - \frac{V_x}{V_{max}}\right)$$

$$V_{comfort} = \sqrt{\frac{|a_{y,comfort}|}{k}}$$

*Equation 4.*

Where, $a_{y,o}$=3.6 $m/s2$ is the acceptable lateral acceleration. Therefore, the speed profile is set as:

$$V_{ref} = \min(V_{max}, V_{comfort})$$

*Equation 5.*

For lateral stability of the vehicle an additional condition is applied to improve the lateral motion. So, a desired longitudinal acceleration is calculated from physical limitation in braking with cornering.

$$\overline{a_x} = -\frac{\sqrt{(\mu m g)^2 - (\sum F_y)^2}}{m}$$

*Equation 6.*

In this way, a constrain on the longitudinal acceleration is imposed using the Kamm inequality, which keeps the forces developed in the tires within the physical limitations of the tire-road friction.

$$\sqrt{{F_x}^2 + {F_y}^2} \leq \mu F_z$$

*Equation 7.*

Where $F_y$ can be either estimated of it can be measured using recently developed technology like smart tires or load sensing bearings to compute $a_x$ in real time.

### 3.2.4. Adaptative MPC

In Model Predictive Control (MPC) a model of the plant is used to predict the future evolution of the system on a fixed finite time horizon $Tp$. Based on this prediction, at each time step $t$ a cost function is minimized under the operating constraints with respect to a sequence of future input moves in order to best follow a reference output. The first of such optimal moves is the control action applied to the plant at time t. At time t + 1, a new optimization is solved over a shifted prediction horizon. Generally, the cost function to be minimized is given by:

$$J = \sum_{j=1}^{T_p} \left\| y_p(k+i) - y_{ref}(k+i) \right\| w_y + \sum_{j=0}^{T_c-1} \| u(k+i) \| w_u$$

*Equation 8.*

Where $yp$ and $yr$ are predicted and reference outputs, $Tp$ is the prediction horizon and $Tc$ is control horizon, u is the manipulated variable. $wy$ and $wu$ are weights for outputs and manipulated variables respectively.

The control inputs for the MPC are reference velocity $Vref$, lateral deviation $e_1$ and relative yaw angle $e_2$. Based on these three inputs the MPC act on steering angle and throttle/brake to minimize the control objective given by:

$$\begin{aligned} &\min \quad J = w_y \left| y_p - y_{ref} \right| + w_u |u| \\ &\text{s.t} \quad x(k+1) = Ax(k) + B\_u\, u(k) + B\_d\, v(k) \qquad a_{min} < a_x < a_{max} \\ &\qquad\; y(k) = Cx(k) + D_d v(k) \qquad\qquad\qquad \delta_{min} < \delta < \delta_{max} \end{aligned}$$

*Figure 20. Optimization Problem*

The MPC Cost function implemented and used in this thesis, as said before, is the one developed in *Model Predictive Control based lateral and longitudinal controller for autonomous driving* (Khan, 2019) and the optimization problem is solved by a QP solver, in Model Predictive Control Toolbox.

## 3.3. Vehicle Modelling for validation

Mathematical modelling of vehicle represents a major part in the control and analysis of autonomous driving vehicles. Several mathematical models are present in literature with different levels of complexity and accuracy based on the research context.

As it has been explained before, this model is the one receiving the car inputs outputted by the controller, and is the one in charge of feedbacking an accurate updated vehicle state to both, the controller and the position in the scenario (just in the case the work is being done in a simulation environment and not by camera's lane detection). In Figure 6 this is included in the Vehicle and environment simulation Block too.

### 3.3.1. Three degrees of freedom Model and Tire Model.

The vehicle and tire model is going to be used for the validation and environment simulation is the one used and more deeply explained in *Model Predictive Control based lateral and longitudinal controller for autonomous driving* (Khan, 2019).

#### 3.3.1.1. Three degrees of freedom Model

In Khan's model dynamics of the vehicle is modeled using the three degree of freedom rigid vehicle model (Single Track), which is imported from Vehicle dynamics Block set in Simulink®. This model accounts for the two displacements on the plane (longitudinal, depicted by subscript x and lateral depicted by subscript y) and the rotation around an axis normal to the plane (yaw motion).

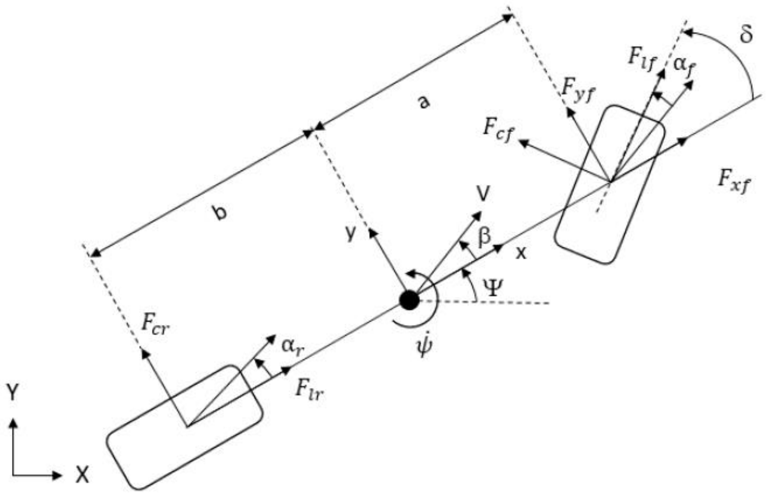


*Figure 21. Three degrees of Freedom Model*

It implements a rigid two axle vehicle body model. So, the two front wheels and the two rear wheels of the vehicle are represented as a single center wheel. As the vehicle is only steerable from the front wheels, the test vehicle is modeled to be only steerable from the front wheel. The nomenclature refers to the model depicted in Figure 14. It is denoted by $F_l$, $F_c$ the longitudinal (or "tractive") and lateral (or "cornering") tire forces, respectively, $F_x$, $F_y$ the longitudinal and lateral forces acting on the vehicle center of gravity, $F_z$ the normal tire load, X, Y the absolute car position in inertial coordinates, a, b (distance of front and rear wheels from center of gravity), g the gravitational constant, m the car mass, $I_{zz}$ the car inertia, α the slip angle, δ the wheel steering angle and ψ the heading angle. The lower scripts (·)f and (·)r particularize a variable at the front wheels and the rear wheels, respectively, e.g. $Flf$ is the front wheel longitudinal force.

In the next figure (Equation 8) the equations that govern the model divided in three are shown.

$$
\begin{aligned}
m\ddot{x} &= m\dot{y}r + F_{xf} + F_{xr} \\
m\ddot{y} &= -m\dot{x}r + F_{yf} + F_{yr} \\
I_{zz}\ddot{\psi} &= aF_{yf} - bF_{yr}
\end{aligned}
\qquad
\begin{aligned}
F_{xf} &= F_{lf}\cos\delta - F_{cf}\sin\delta \\
F_{yf} &= F_{lf}\sin\delta + F_{cf}\cos\delta \\
F_{xr} &= F_{lr} \\
F_{yr} &= F_{cr}
\end{aligned}
\qquad
\begin{aligned}
\dot{X} &= \dot{x}\cos\psi - \dot{y}\sin\psi \\
\dot{Y} &= \dot{x}\sin\psi + \dot{y}\psi \\
\dot{\psi} &= \dot{\psi}
\end{aligned}
$$

*Equation 9.*

It can be observed (Equation 1) longitudinal and lateral momentum with respect to CoG, the yaw dynamics, the forces acting on the vehicle center of gravity are related to tires forces and front steering angle by the given equations. The fixed frame coordinates are related to the vehicle coordinates by the given equations.

### 3.3.1.2. Tire Model

The figure 15 details the notations used to describe the tires interaction with the road. This is needed to fully characterize the dynamics of the vehicle in complementation with the three degrees of freedom model.

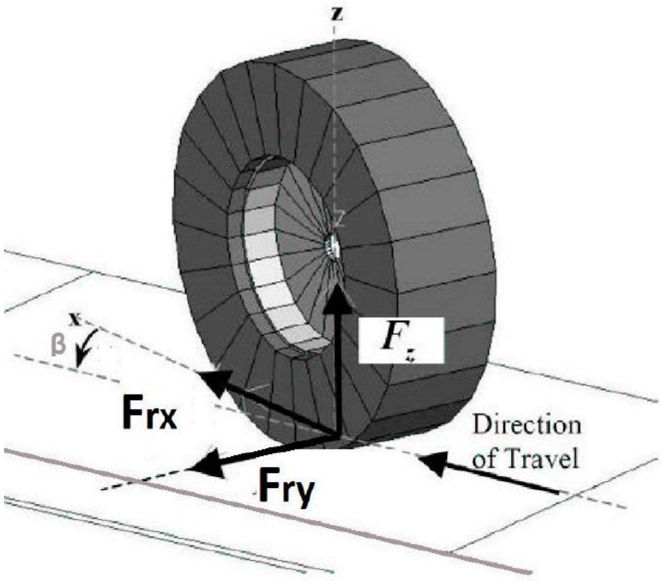


*Figure 22. Tire lateral and longitudinal forces*

As outlined in Equation 1 (first 3 equations) the longitudinal forces $Flf$ and $Flr$ are generated in the tires due to the acceleration and braking command $ax$. So, the longitudinal momentum due to this can be written as

$$m.a_x = F_{lf}\cos\delta + F_{lr}$$

*Equation 10.*

Front and rear tire force distribution can be defined as

$$d_x = \frac{F_{lf}}{F_{lf}+F_{lr}}$$

Equation 11.

So, the $F_{lf}$ and $F_{lr}$ can be determined as a function of $a_x$

$$F_{lf} = \frac{m}{\cos\delta + \frac{1-d_x}{d_x}} \cdot a_x$$
$$F_{lr} = \frac{m}{1+\frac{d_x}{1-d_x} \cdot \cos\delta} \cdot a_x$$

Equation 12.

For simplicity it is assumed that

$$d_x = \frac{F_{zf}}{F_{zf}+F_{zr}}$$

Equation 13.

Where $F_{zf}$ and $F_{zr}$ defines the normal forces on the front and rear axle considering the load transfer during acceleration and steering.

$$F_{zf} = \frac{bmg-(\ddot{x}-\dot{y}r)mh_g}{a+b}$$
$$F_{zr} = \frac{amg+(\ddot{x}-\dot{y}r)mh_g}{a+b}$$

Equation 14.

On the other hand, the lateral forces $F_{cf}$ and $F_{cr}$ in Equation 8 (first 3 equations) are calculated using the linear model of the tire. Which assumes that lateral tire forces are linear function of tire slip angles $\alpha_i$ and cornering stiffness:

$$F_{cf} = 2.C_{yf}\alpha_f$$
$$F_{cr} = 2.C_{yr}\alpha_r$$

Equation 15.

Where:

$$\alpha_f = \operatorname{atan}\left(\frac{\dot{y}+a\dot{\psi}}{\dot{x}}\right) - \delta$$
$$\alpha_r = \operatorname{atan}\left(\frac{\dot{y}-b\dot{\psi}}{\dot{x}}\right)$$

Equation 16.

### 2.3.1.3. Slip Angle Analysis

To fully characterize the dynamics of the vehicle and thus drive it at the limits of handling it must be known that under some conditions, the vehicle can lose adherence and then slip. In automotive vehicle dynamics, slip is the relative motion between a tire and the road surface it is moving on. This slip can be generated either by the tire's rotational speed being greater or less than the free-rolling speed, usually described as percent slip, or by the tire's plane of rotation being at an angle to its direction of motion (referred to as slip angle).

Actually, the used tire model is not taking into consideration the slip angle and its effect in a linearized way, but actually the forces and slip regime are nonlinear. That could be a problem if the vehicle dynamics overcome certain circumstances, because the wheel blocking won't be considered even if it would occur in the real vehicle. For that it has been implemented an algorithm that is given warning when the blocking of the wheel may occur.

For the implementation of this observation algorithm the ABS threshold controls of the Vehicle Control Systems (Chapter 10) of Automotive Control Systems (Kniecke & Nielsen, 2005). For this statement is known that the lateral acceleration $a_y$ is limited by the friction coefficient $\mu_S$ (static friction coefficient). Theoretically, a vehicle can turn with a lateral acceleration of 9.81 times the maximum lateral friction coefficient (for a $\mu_S$ of 1, the lateral vehicle acceleration could be 9.81 $m/s^2$ if the vehicle body side slip angle were zero). For vehicle body side slip angles greater than zero, a maximum acceleration of 8 $m/s^2$ is adopted. For friction coefficients below 1, the maximum lateral acceleration is given by:

$$a_{Y\,max} = \mu_S \cdot 8\, m/s^2$$

*Equation 17.*

Even if in previous modelling and equations the body side slip angle is noted down as α, for now the Automotive Control System (Kniecke & Nielsen, 2005) notation is going to be used. In this book the side slip angle is related as β. The body slip angle β is also limited. The predominant variable affecting the is the vehicle speed Vcog. (see Figure 17). So, the regime of the maximum slip regime is noted ad the equation 10.

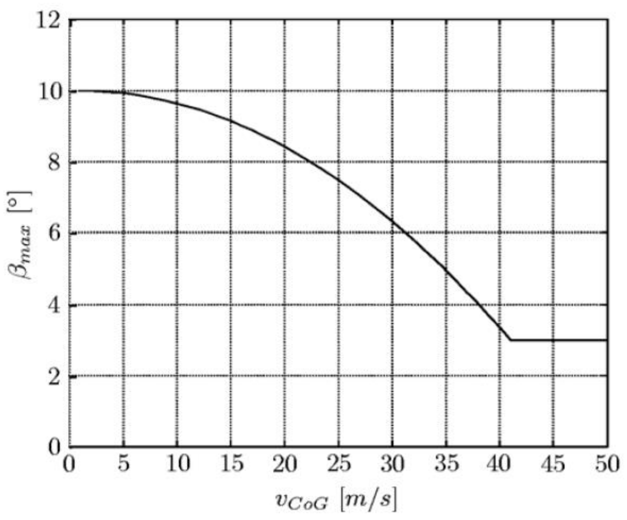


*Figure 23. Dependence of maximum vehicle body side slip angle.*

$$\beta_{max} = 10^\circ - 7^\circ \cdot \frac{v_{CoG}^2}{(40\,m/s)^2}$$

*Equation 18.*

$$\beta_{ref} = \begin{cases} \beta & , \quad |\beta| \leq |\beta_{max}| \\ \pm\beta_{max} & , \quad \text{otherwise} \end{cases}$$

*Equation 19.*

The reference vehicle body slip angle of the equation 11 is the body slip angle the ABS should provide by avoiding the wheel blocking. The maximum Betha should be, so, the Betha that the body side slip angle should never pass by, because it is no ABS implemented in the simulation model. As it is only dependent on the longitudinal velocity of the vehicle, the algorithms are easily implementable. In Figure 18 we could se how the algorithm plots the superior and inferior limits of the side slip angle (in which theoretically the ABS would activate), as well as the instantaneous Betha for each time step.

Whenever the Betha gets out of the set superior and inferiorly bounded by the limits, a wheel blocking may occur, so that would mean that the simulations and control parameters should be changed and adapted to the current track due to high velocities, difficult in adherences or very sharp curves.

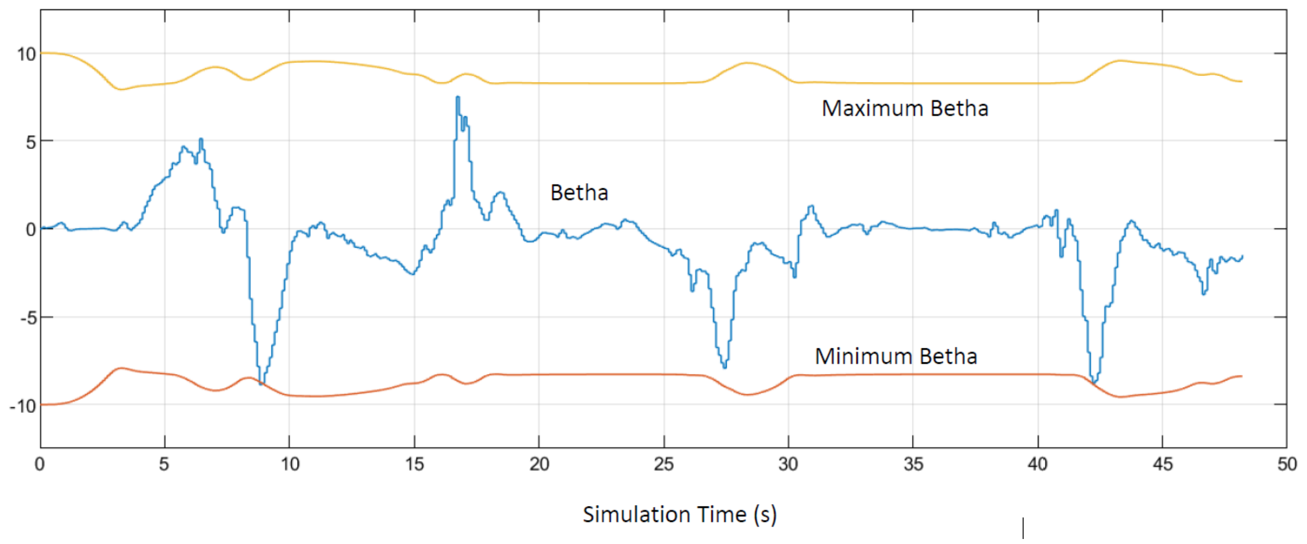


*Figure 24. Simulation in a suburban environment for Betha analysis*

In this figure, it can be observed example on how the side slip changes and its behavior in a suburban environment. , and how in two points (seconds of simulation 8 and 43) the Betha is reaching risky points, but the wheels won't block due to the fact of the parameter not surpassing the limits.

# 4. Path Planning

The trajectory generation phase consists to find the trajectory and compute its curvature based on the information of the lane line model coming from the previous step. This phase refers to the problem of trajectory planning, also called motion planning, in automotive context, that has the purpose to find a trajectory feasible for the vehicle, and safe and comfortable for the passenger. The motion planning for an autonomous vehicle is based on the same theory handled in robotics area. In fact, as in the field of robotics, it is necessary to provide and distinguish some definitions such as path and trajectory .Firstly, it is significant to give the definitions of path and trajectory and underline that they have two different meanings:

- *Path* is the pure geometric description of motion;
- *Trajectory* is the merge of the path and the time laws (velocities and accelerations)required to follow the path.

In this thesis, for the sake of simplicity, no strict distinction has been adopted to distinguish path and trajectory when needed, but the one that refers to the format. When Path is being named, it refers to a array of coordinate points, which in the correct order and connected forma a trajectory. The trajectory will be used as the function (whatever is its form) that defines the path in a continuous way. In this section the definition of the plan will be exposed, even if sometimes trajectories are named.

The trajectory computed in past works consisted of the center line of the lane. It was computed like the average between the left line of the lane and the right ones. Nevertheless, as mentioned in the problem statement and thesis objectives the main aim of this thesis is to generate a trajectory which is the shortest way to arrive to the farthest point of the recognizable environment. This means, that the trajectory generated should be the one which leads to the same point as the center line (computed in the past works) but with the shortest possible route.

For that it has been created an algorithm based on the Probabilistic RoadMap theory commonly used in robotics. The configuration of the parameters of the PRM (Probabilistic Road Map), the map creation and the execution of the algorithm will be done with the Robotics System Toolbox.

## 4.1. Probabilistic Roadmap Fundamentals

As exposed in *Probabilistic roadmaps for path planning in high-dimensional configuration spaces* (Kavraki, Svestka, & Latombe, 1996) the probabilistic roadmap planner is a motion planning algorithm (a term used in robotics is to find a sequence of valid configurations that moves the robot from the source to destination) in robotics, so the robot in this thesis will be the autonomous vehicle. This planning algorithm solves the problem of determining a path between a starting configuration of the robot and a goal configuration while avoiding collisions.

The simulation of the vehicle in the scenario is computing for each time step coordinates of where the car is, so for each time step the algorithm should be executed once, generating a path. The reason of this is because the trajectory computed is entering as input to the RVP (Reference velocity profile) and the MPC, which are computing their objectives once per time step. So, the starting configuration of the robot will be the current vehicle position and the goal configuration is being computed as the last point of the readable road's center line.

In the example of figure 25, as an example, the scenario is a two-way road, and the camera is able to read 60 meters forward from the Cog of the car. In Figure 25 it can be seen how the scenario information provides the coordinates taking as a reference (origin of coordinates) the vehicles center of gravity. It also can bee observed the configured 60-meter range of the camera, and as an example the goal configuration of that instant of time. This camera configuration will be used for now on, even if in a more usual situation, its range will be far shorter due to difference in elevation of the road or other vehicles or obstacles. The images in Figure 35 are taken from the Bird's Eye plot Block of the Automated Driving Toolbox.

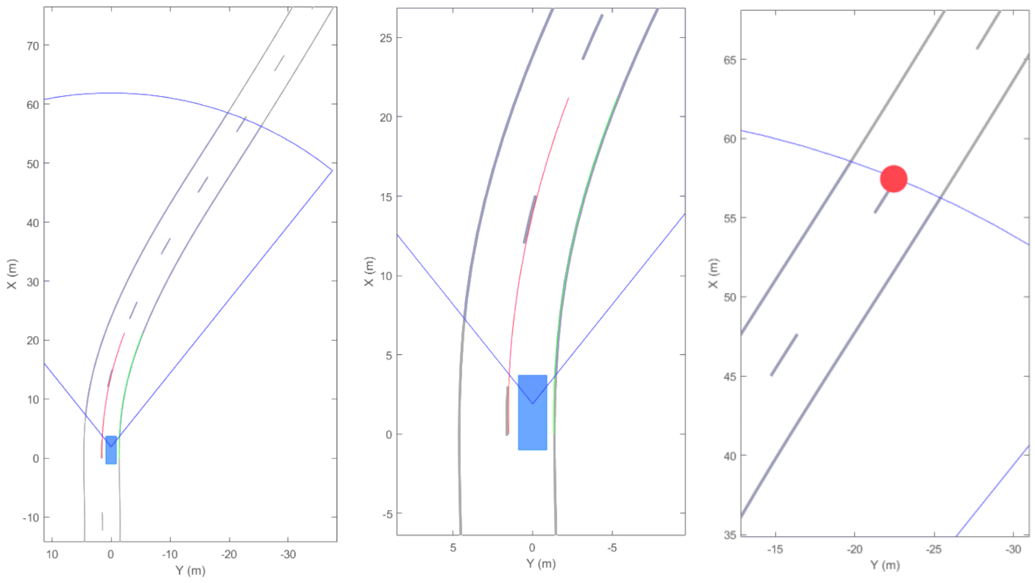


*Figure 25. a) Vehicle and Camera Range b) Vehicle's position c) Goal Configuration*

The basic idea behind PRM is to take random samples from the configuration space of the robot, testing them for whether they are in the free space, and use a local planner to attempt to connect these configurations to other nearby configurations. The starting and goal configurations are added in, and a graph search algorithm is applied to the resulting graph to determine a path between the starting and goal configurations.

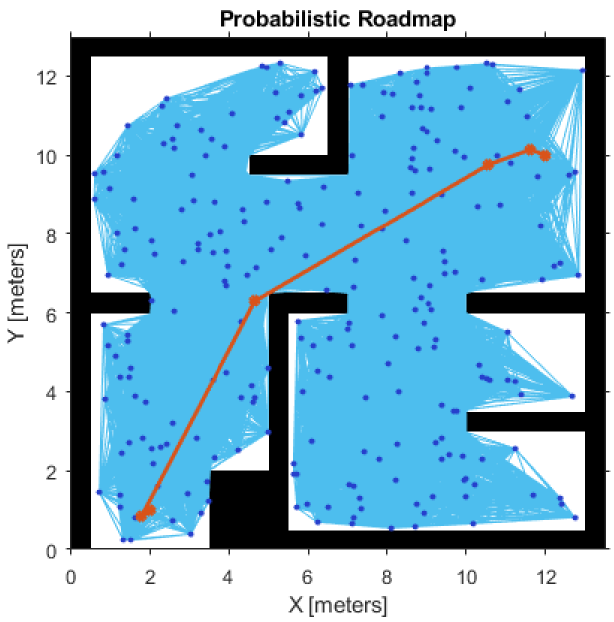


*Figure 26. Example of a simple PRM map and solution.*

The probabilistic roadmap planner consists of two phases: a construction and a query phase. In the construction phase, a roadmap (graph) is built, approximating the motions that can be made in the environment.

As it will be explained next, first, a random configuration is created. Then, it is connected to some neighbors, typically either the k nearest neighbors or all neighbors less than some predetermined distance. Configurations and connections are added to the graph until the roadmap is dense enough. In the query phase, the start and goal configurations are connected to the graph, and the path is obtained by a Dijkstra's shortest path query.

## 4.2. Roadmap Construction

The first step in order to compute the trajectory is to generate a Roadmap. The Robotic System Toolbox (*BinaryOccupancyGrid* function) let the user create a 2-D occupancy grid object, which can used to represent and visualize a robot workspace, including obstacles. The integration of sensor data and position estimates create a spatial representation of the approximate locations of the obstacles. In this case, as the robots would be the vehicle, the obstacles should be occupying all the are which the car must not go over. The main idea is to create a map in which the road boundaries are represented as impassable obstacles.

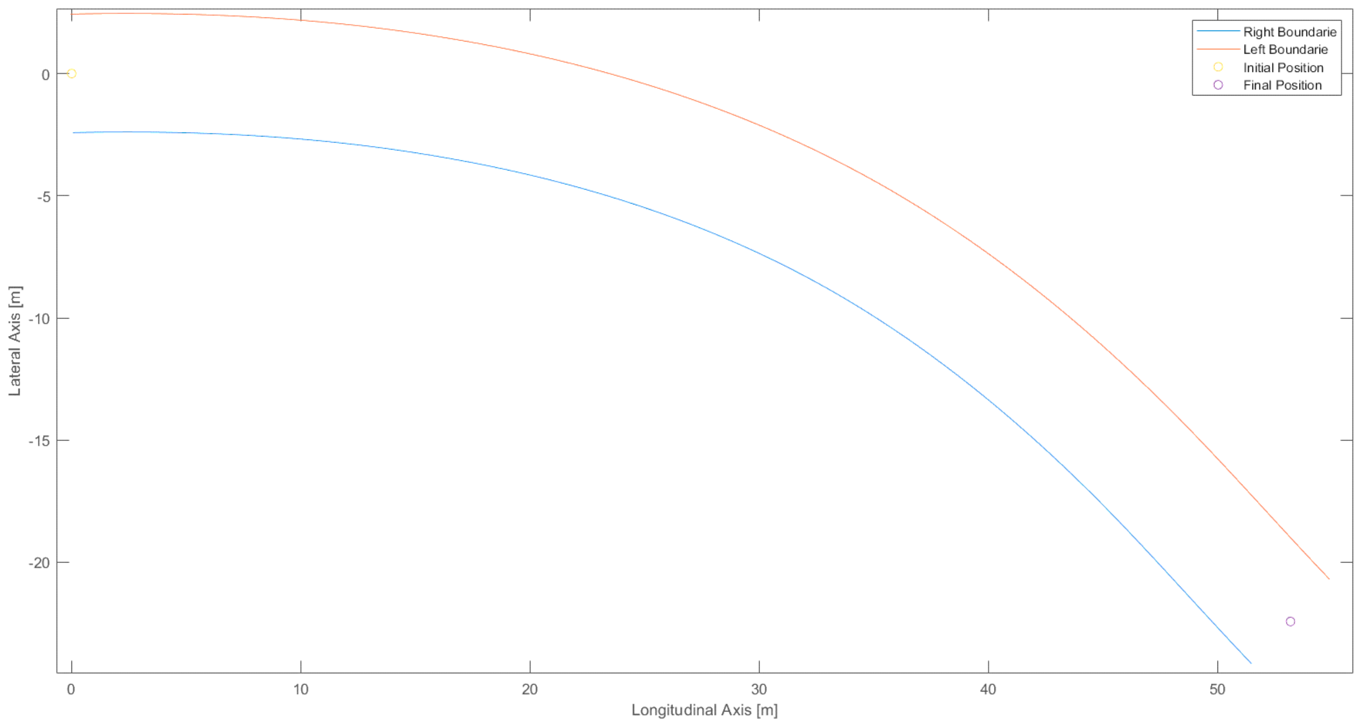
Another important fact is that the map is variable sized depending on the time step, due to the fact that the road lectures are dependent on the part of the scenario where the vehicle is actually located. If the vehicle is located in the center line and has a perfect straight road in his front, the longitudinal size of the map will be 60 meters, but it will decrease if there are curves. The roadmap’s size, as it is going to be explained in a deeper way, is important due to many facts as operational cost between others.

An important comment to make, is that the *BinaryOccupancyGrid* in its map generation, doesn’t allow negative parts, every point to generate the map should be positive. This is a problem because as if the coordinates system is located in the vehicle’s CoG, necessarily the right boundaries are going to be, at least the first meters, negative. To avoid this to be an obstacle, the solution adopted has been to change the reference system. So, this way, the CoG is elevated (as well as the boundaries) in order to make every coordinate point positive. Obviously, this reference change is dependent to each time step and road lecture, so once the path is being calculated, it is referred again to the original reference system.

In the Figure 27 it can be seen how the lectures and the map have the coordinate system changed.

The parameter used to increase the Y-axis value is the absolute value of the most negative Y-axis point in the original coordinate system. That way, once the path is generated, just by subtracting that scalar parameter to the Y coordinate of each point of the path will move it to the desired reference system.

In Figure 27 (a) it can be also who the initial position and the computed final/objective positions are plotted. Also, this final and initial position, once the map is created, are being changed of reference/coordinate system. That way the whole system is being moved up, but only the generated path, that ultimately is the aim of this part of the solution, will be again referred to the desired reference.



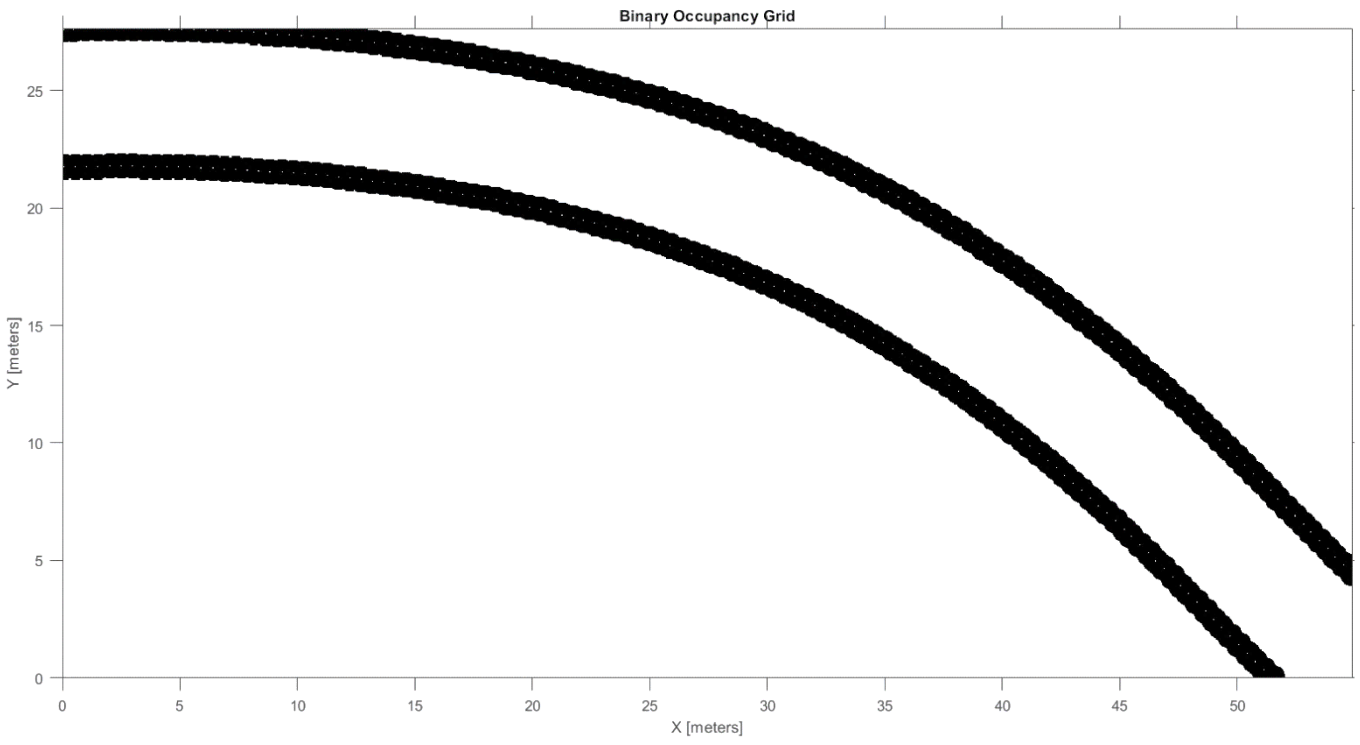


*Figure 27. a) Road Lectures b) Creation of the map with the changed reference*

## 4.3. Dijkstra's shortest path

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later in *A note on two problems in connexion with graphs* (Dijkstra, 1959)

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. In the case of this thesis, if the nodes of the graph represent positions and edge path costs represent driving distances between pairs of position, Dijkstra's algorithm can be used to find the shortest route between one source position and a goal position. This is linked to the previously defined initial and goal positions.

Let the starting position of the CoG (for that time step) be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B through A will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
4. When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

When planning a route, it is actually not necessary to wait until the destination node is "visited" as above: the algorithm can stop once the destination node has the smallest tentative distance among all "unvisited" nodes (and thus could be selected as the next "current").

Back to the algorithm implementation, the map previously created will serve as a layout for the deployed nodes. That map will be set with the nodes the Dijkstra's algorithm will evaluate for the path construction. As it can be observed in Figure 27 the initial and goal position are inside the road boundaries, and all the road is limited by these boundaries. This road boundaries are not transitable for the vehicle, so the node distances are not being computed if they need to cross the physical obstacles that represent the boundaries in the map.

In the case of this thesis, two parameters for the algorithm are adjustable. These parameters bound the accuracy and optimization of the path.

The first one is the number of nodes. When more nodes are deployed, more accurate the path will be. Nevertheless, the Dijkstra's algorithm will have to compute more distances, which will increase operational cost. Increasing the number of nodes can increase the efficiency of the path by giving more feasible paths. However, the increased complexity increases computation time. To get good coverage of the map, it might be needed a large number of nodes.

The second parameter is the connection distance. Using the connection distance property as a threshold for distance, the algorithm connects all points that do not have obstacles blocking the direct path between them.

The Figure 28 shows a computed path for the Figure 27 ‘s lectures with a connection distance parameter of 8 meters and 2000 nodes deployed.

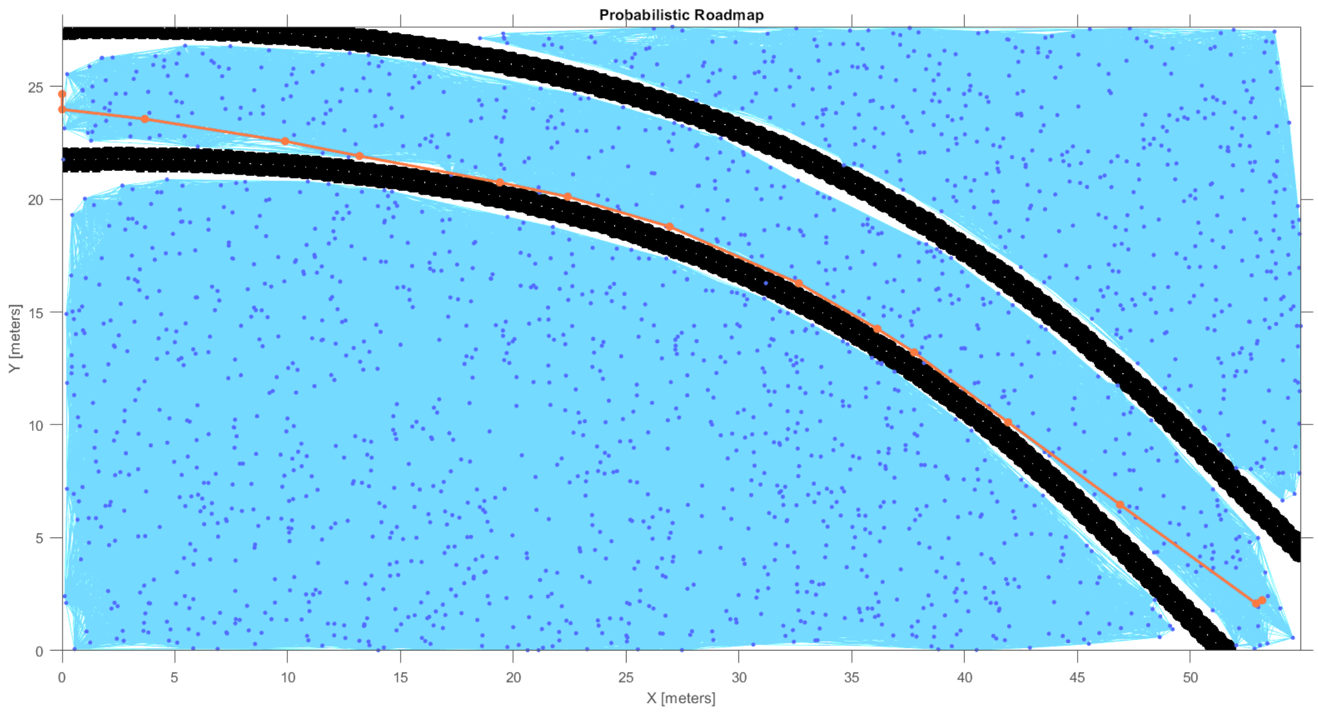


*Figure 28. Probabilistic Roadmap with computed path*

As it can be seen, the path is in the PRM coordinate system, which is going to be changed back to the CoG reference system, as it has been said before (Figure 29). Nevertheless, the path is far from being perfect. The next section will explain the different optimization improvements that have being implemented.

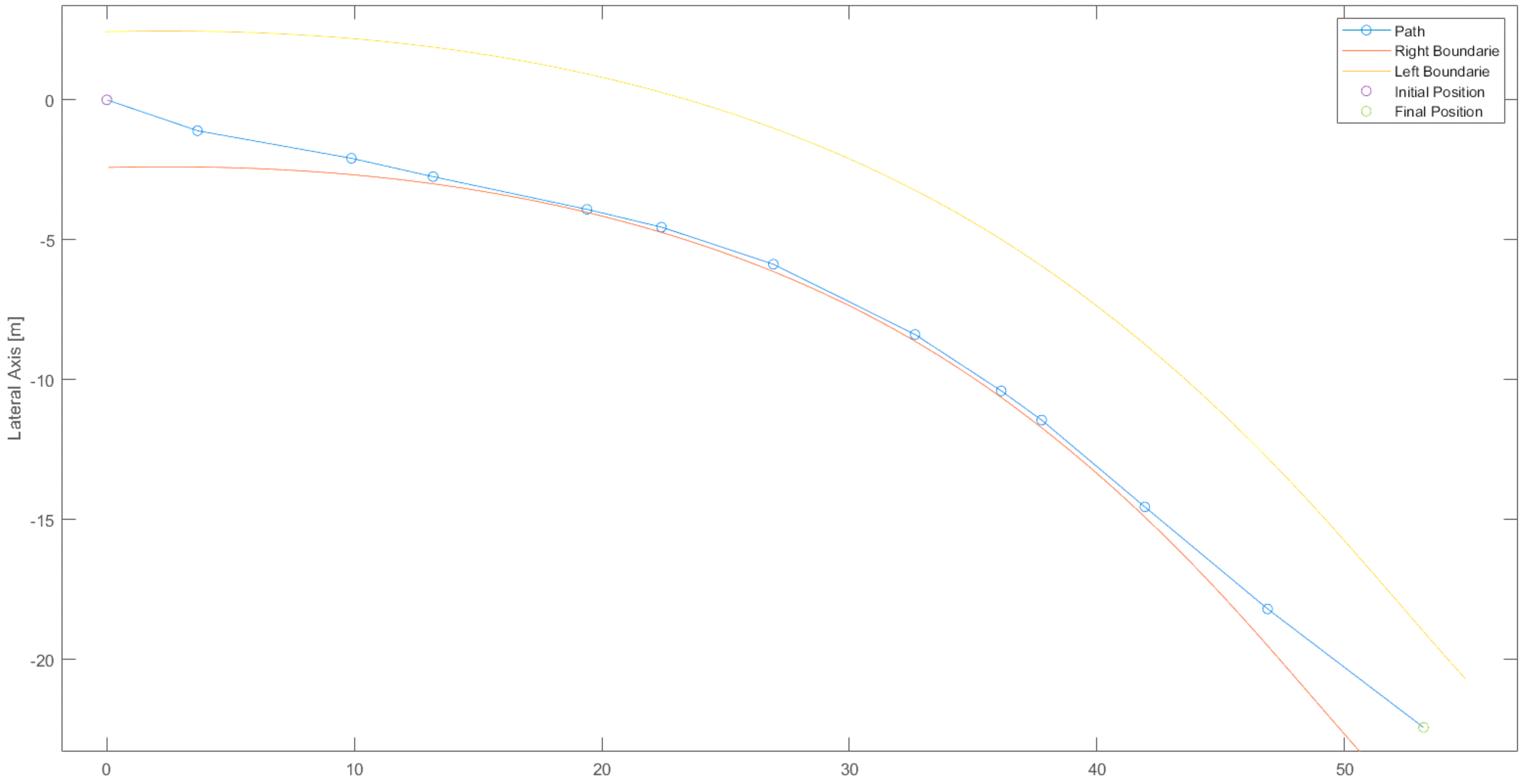


*Figure 29. Example Computed Path*

# 5. Path Planning Optimization

As seen in the previous section, the path is being already generated but it is not suitable for the control of the car. In order to make it useful some improvements have been made to optimize the operational cost, calculation time, suitability of the path and others.

## 5.1. Fulfillment of the PRM.

As it can be seen in Figure 28, from the total number of nodes just a small fraction is located in the inside of the road, where the path computation is feasible. Nevertheless, the deployment of this high number of nodes that get located outside of the road, and consequently are useless for the whole algorithm, still consume memory, and make the algorithm slower and harder.

The solution adopted in order to avoid this useless time and computation is to make a way for the algorithm just to deploy nodes inside the road. In order to achieve that another algorithm has been created. This second algorithm fills the map with random coordinate points. A selection is made to separate the points inside the road from the outside ones. That way the inside points are being deleted, but the outside ones are written in the map as obstacles. So, if the outside part of the road in the map gets filled, the nodes won't be deployed there. Continuing with Figure 27's example, Figure 30 shows how the map is filled with the random points.

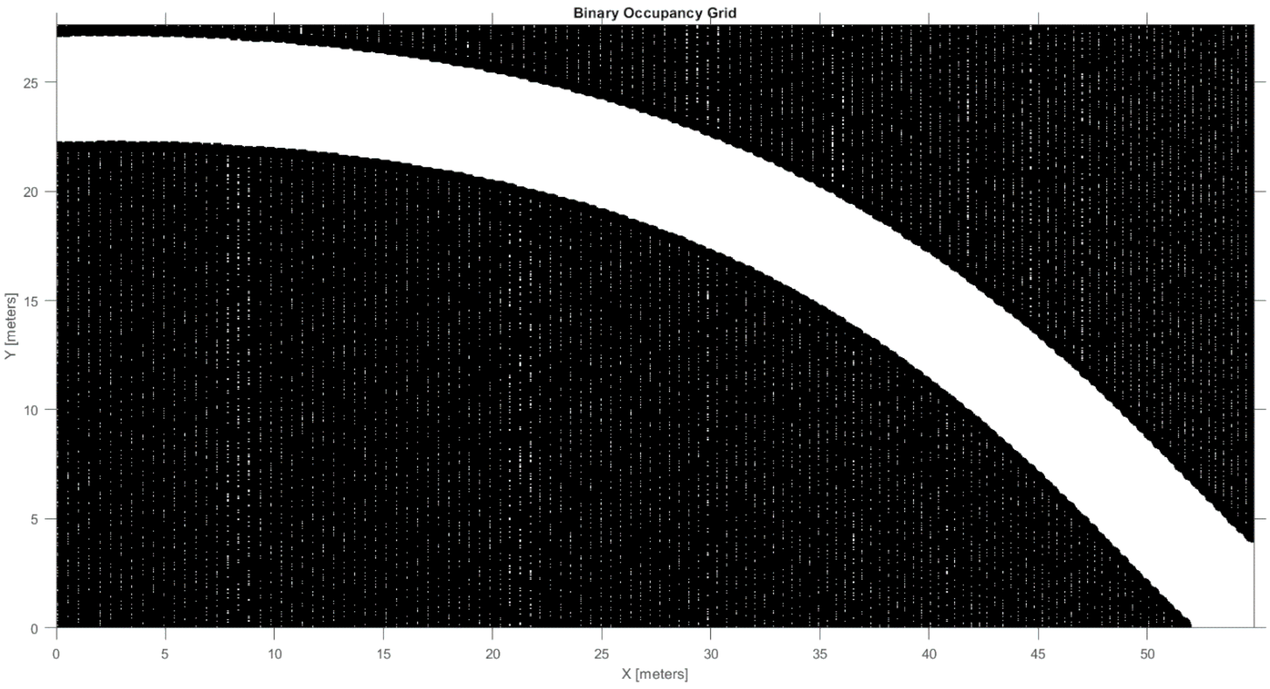


*Figure 30. Filled RoadMap*

This way the unique place to deploy nodes is the map is in the road, so all the computed distances will be feasible and useful to create the path (Some nodes are deployed between the random points occupancy grids, but are neglectable to operational cost). A PRM created path is shown in Figure 31 with this improvement implemented.

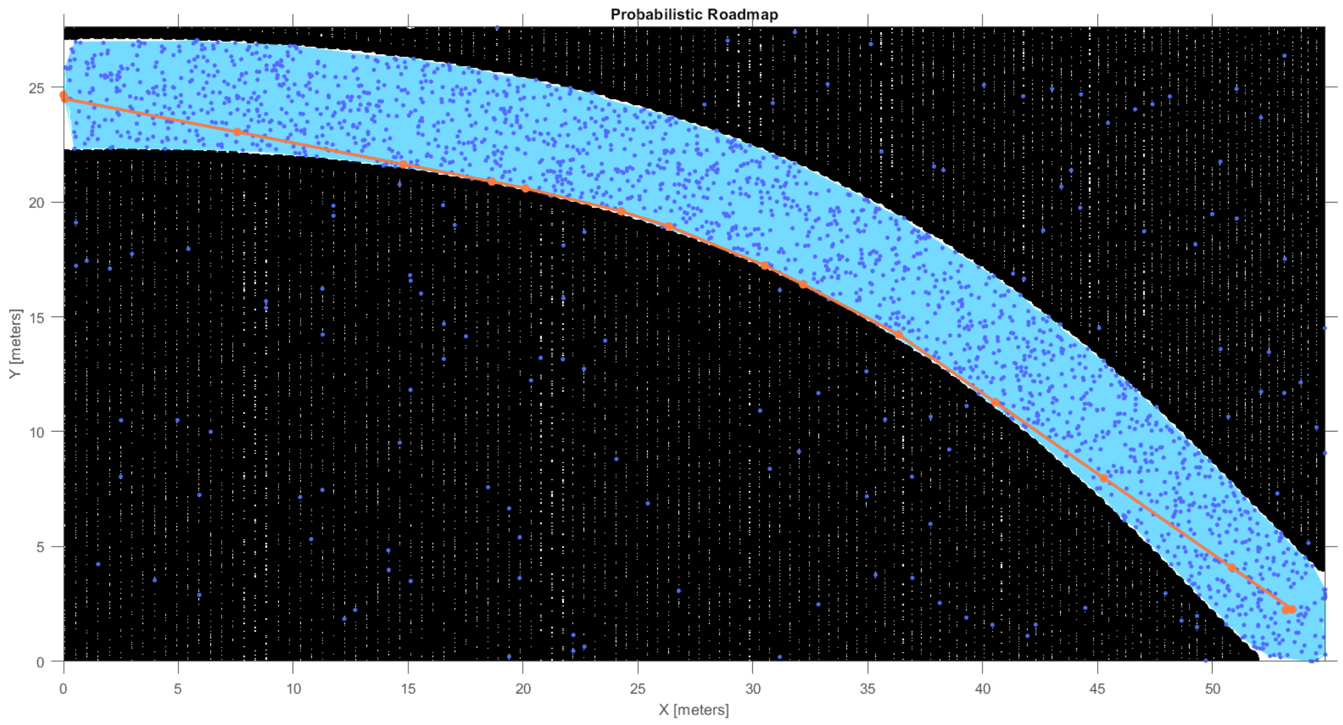


*Figure 31. Path Planning with filled PRM.*

## 5.2. Smoothing of the path

If the path gets sharp changes between the points, the trajectory generation will get more difficult. After some experiment, it has been denoted that the toolbox sometimes gets path points very near to the initial position or the goal position.

In order to avoid this to happen (which will be a problem in order to fit a function to generate the trajectory) a cleaning of the path is being made. To do that, an algorithm that computes the distance between the initial position and the next points of the path is being done. If the distance is too short or the changes to sharp, the algorithm will remove the points of the path that are necessary.

An example of this can be seen how in the PRM of Figure 29 a very near node is going to be used for the path. This computation will result in a failure, but with this algorithm it can be seen in Figure 30 how this point is being neglected.

## 5.3. Regime classification and algorithm adaptation

After some experimentation with the algorithm in some scenarios it has been concluded that the PRM algorithm is very useful for curvy tracks, but when it comes to computing straight lines or very straight road sections, it is a waste of resources, time and operating costs.

In some cases, the road gets straight enough for just to compute the path and the trajectory with straight lines in a linear way. In others, the curves come after a first straight section. The las option is as the one exposed in Figure 27, when the curve is being taken or is approaching.

In order to differentiate the regime and apply the algorithm in an optimal way, the read road is going to be divided into ten different part, being analyzed from the nearest part to the furthest one. Now, it is important to understand the concept of the rate of change in the curves. This should be focus of a deeper analysis, but for this thesis the supposition is that if the analyzing part’s boundaries have no bigger change (in lateral Y-axis) than 0.5 meters, that part of the track is being considered as straight. So, if the whole read track for the time step has no bigger changes in each of the divisions, the track will be considered straight.  Besides that, If the road starts getting straight but in a certain point of the time step’s lecture the rate of change surpasses 0.5 meters, a partial PRM will be developed. This means that the first part of the track will get a linear computation of the path, and the final point of the straight linear path will work as the initial position of the PRM that will be computing the second part’s path. However, as the trajectory computation could make a sharp change when going from the straight path to the curve’s one, the change from one to other will be made one part before the first curve part. That way the PRM will compute a less abrupt change from one kind to the other.

Finally, the last option is the one where the first part of the read road is already with a change rate of more than 0.5 meters. In this case the PRM will compute the whole path due to the curvy character of the read road in that time step.

Figure 32 is showing two examples of the second and third cases, where the PRM is partially computed and fully computed. (It has been fitted the paths into polynomial to understand better the figures, as it will be explained in the subsection 6.1.)

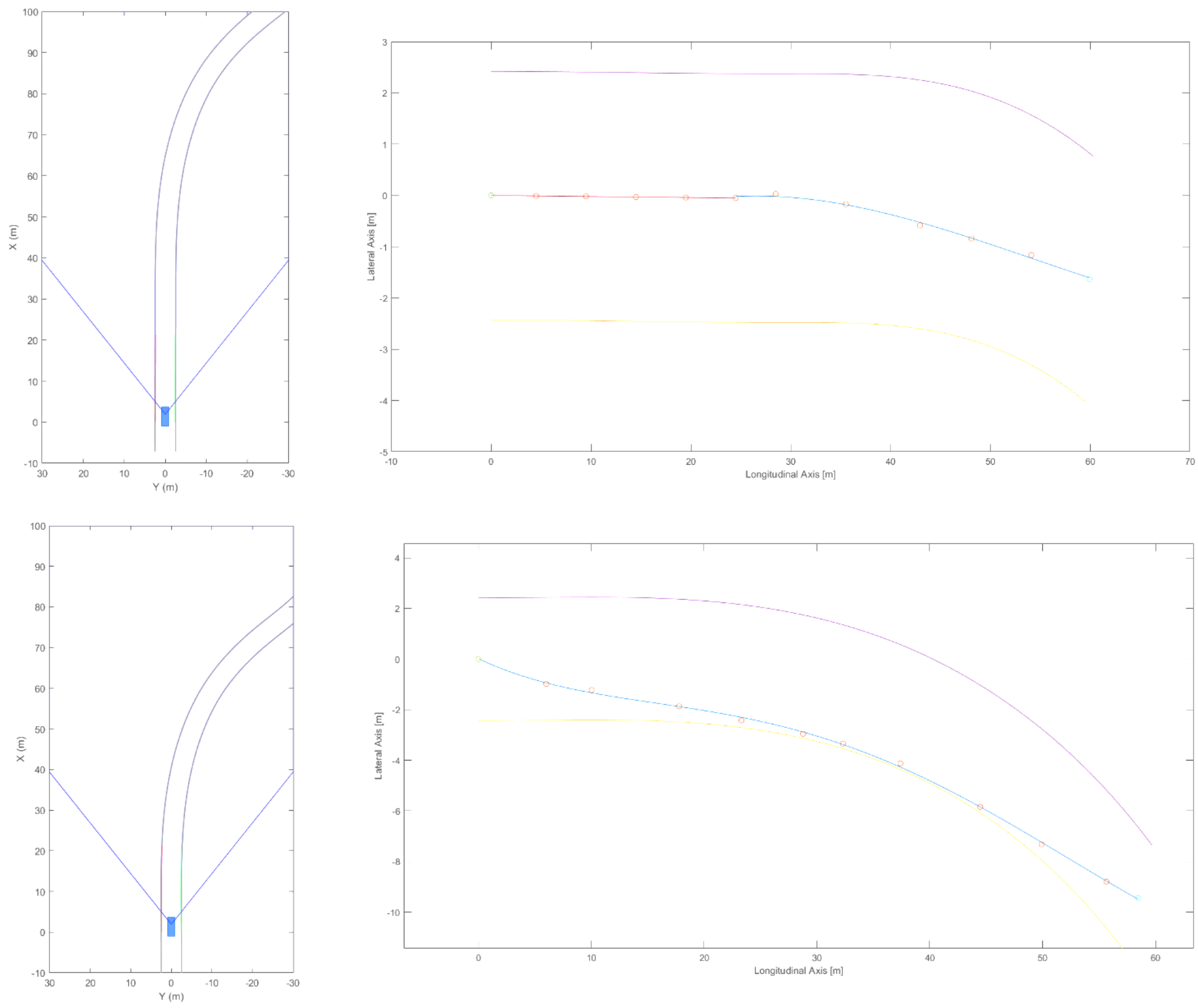


*Figure 32. Partial and full PRM path computation*

## 5.4. Velocity Bounding Lines

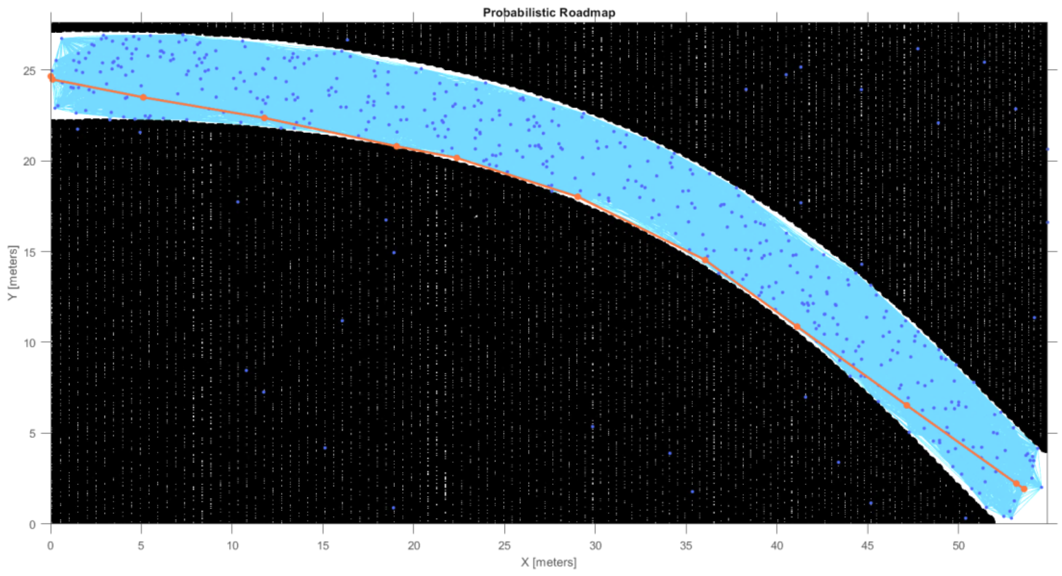
Once understood how the path is being created many problems can be denoted. Nevertheless, once the overall control layout is observed, it is noticed that one of the most problematic issues the vehicle has to achieve is being able to adapt to the reference path (center line or optimized trajectory) when it comes with sharp and drastic changes. One of the critical issues is to, so, be able to generate path that won't become very changing and unadaptable trajectories.

This trajectory sudden changes in the three inputs of the MPC (relative yaw angle, lateral deviation and the curvature sequence of the trajectory generation) or very inconstant regimes, make the control to be inaccurate and it works bad. A way to have into consideration the velocity and steerability of the vehicle, which are the main affected by these changes, is to introduce two straight lines in the map, starting from the car's lateral positions, which bound the sharp changes in the path.

The slope of this symmetric lines (over the longitudinal axis of the car) will depend of the velocity of the car, so it has been developed a constat k (º m / s) which divided by the current longitudinal velocity will be the angle in degrees. That angle is the one making the slope (positive and negative) of the both straight lines. This way the path won't be computed with sharp changes, which will smooth the generated trajectories.

The k constant should be changed if the scenario is urban, suburban or highway/free drive. A further study should be made for the slope of the lines, and also its variability and adaptation to different velocity and curve regimes.

The following figures show, at a certain point, what is the difference between a path without the Velocity Bounding Lines and with them. The lines can be seen easily, because the are considered for the PRM as untransactable, the same as it considers the road boundaries. In the figure a k of 0.8333 (ºm/s) with a velocity of 130 m/s, which makes the slope of the line a tangent of a 3º angle.



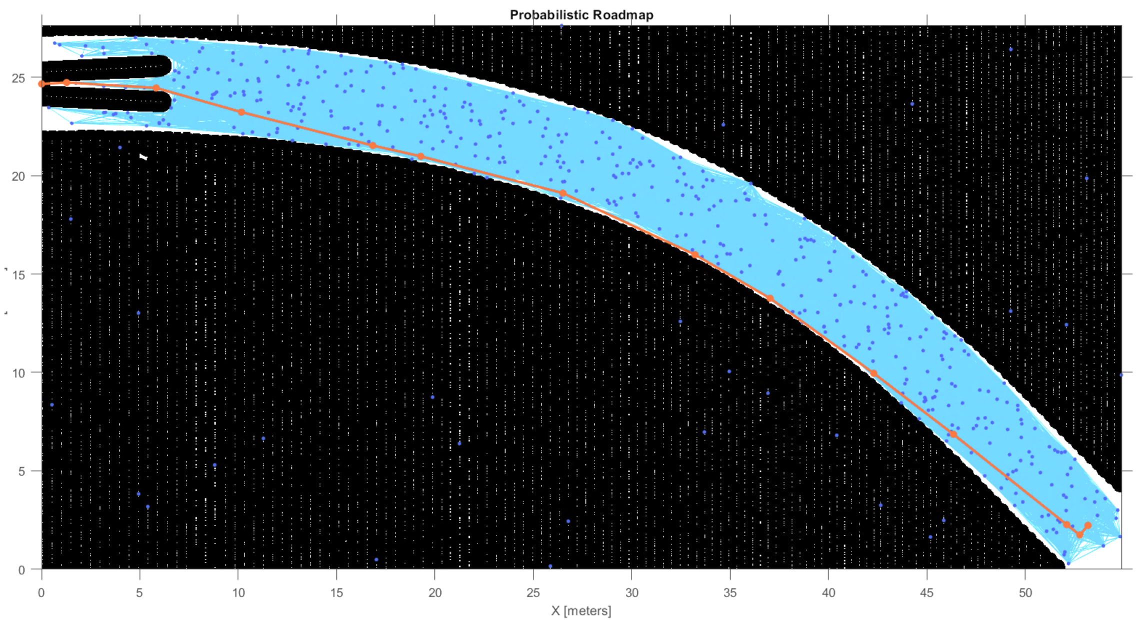
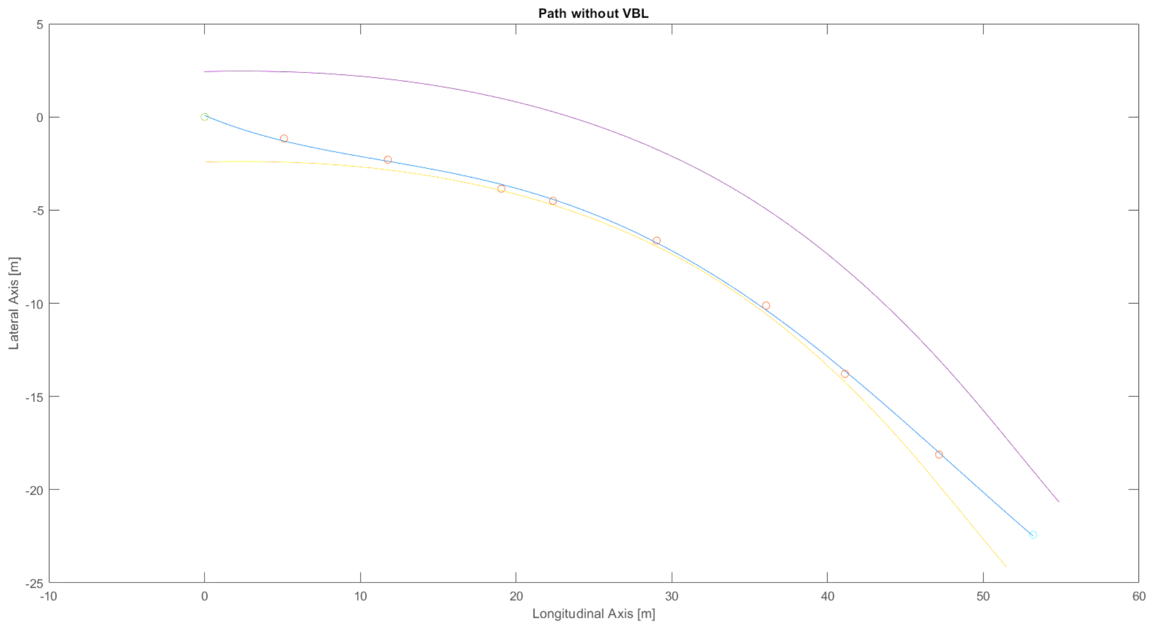


*Figure 33. a) PRM Without Velocity Bounding Lines b) PRM with VBL*

Even if it can be thought that the second path has more changes , it is observable that the reference trajectory that will be generated has less curvature, lateral offset and relative yaw angles. This will be better understood when the trajectory generation based on the path planning is explained in the next section. Nevertheless, once the polynomial function has been fitted (section 6.1) and the trajectory is clear, the changes are very noticeable. The example of figure 33 has been attached next in figure 34.



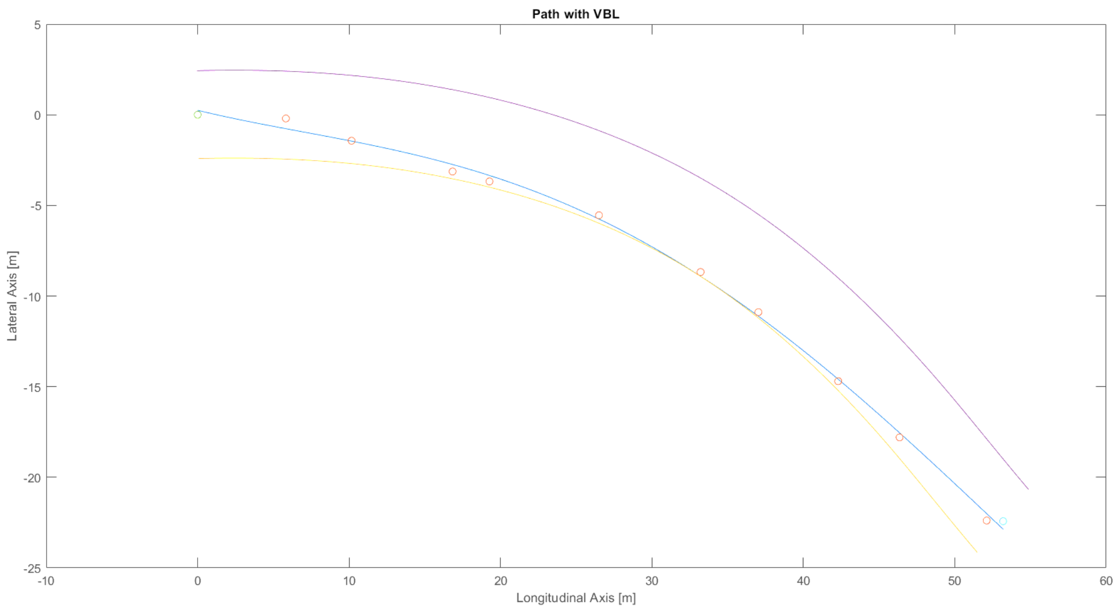


*Figure 34. a) Trajectory without VBL b) Trajectory with VBL*

# 6. Trajectory Generation

Now that the path calculation has been explained, the trajectories are being computed. It is useful to remember the conceptual differentiation made in the introduction of section 4. As the path is the amount of coordinate points that will be following the vehicle, the overall model and the control needs it in another format. The following steps are explaining how the path is converted into trajectories for the vehicle to follow.

## 6.1. Polynomial Function

The first step is to convert the obtained path into a polynomial function. This is being done with the called Polynomial Regression. Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial in x. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y. As it has been said before several times, the x corresponds to the longitunal axis of the vehicle, and the y, to the lateral one. In this case the *PolyFit* MATLAB function is being used. *p = polyfit(x, y, n)* finds the coefficients of a polynomial p(x) of degree n that fits the data y best in a least-squares sense.

As there are three different regimes, three options are contemplated.

### 6.1.1. Straight Line

If the vehicle encounters itself in a straight road, the polynomial, as it was explained before, will consist in a straight line or a polynomial of first degree. This case is the simplest, and the degree of polynomial regression will be 1.

### 6.1.2. Straight line and third-degree polynomial

If the track is divided as mentioned before, the first part of the path will be fitted also as a straight line. However, the second part, which means the one with curves and computed by the PRM, will be fitted into a third-degree polynomial function (n=3). A further study of the polynomial degree of this case could be done in future works, in order to optimize the union of the two secondary trajectories that conform the global one for the time steps which are computed by the partial PRM.

### 6.1.3. Second degree polynomial

When the vehicle is in a curve regime, the PRM computes the total path, and it is fitted into a fourth-degree polynomial (n=2). The reason why the second-degree polynomial is chosen is because it has been demonstrated experimentally by simulations that is not the one which is the most accurate, but it has the less abrupt and violent changes. More degree polynomials may be better in very changing curve regimes, but if the curve regime has one single curve, it could be forced into sudden and rough steering changes. Also, if the Prediction Horizon or the camera length is changed, these degrees could not be the optimal ones.

## 6.2. Trajectory curvature computation

The controller of the lane keeping needs to receive the curvature of the trajectory like input to perform the control action on the steering angle and the throttle as it was explained in the 3.2 subsection. The curvature of a curve parametrized by its arc length is defined in *Geometry of Curves* (Rutter, 2000) as the rate of change of direction of the tangent vector.

Considering a curve $\alpha$(s), where s is the arc length and the tangential angle φ, computed counterclockwise from the x-axis to the tangent T = $\alpha'$ (s), as shown in Figure 33, the curvature $\kappa$ of $\alpha$ is defined, following the definition, as:

$$\kappa = \frac{d\phi}{ds}$$

*Equation 20*

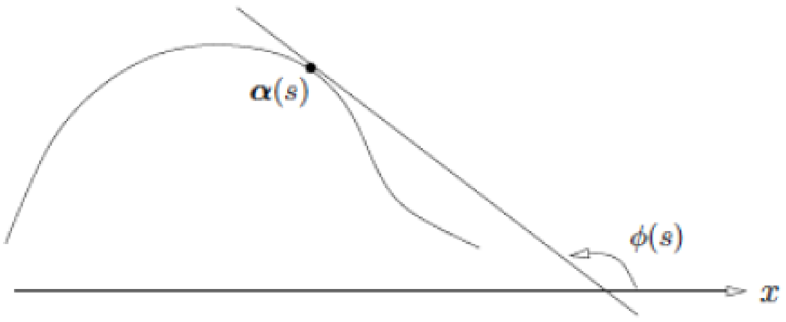


*Figure 35. Curve α and tangential angle ϕ*

These figures are being taken, as well as many information, from *Study and implementation of lane detection and lane keeping for autonomous driving vehicles* (Mancuso, 2018).

The curvature can be also defined as the value of the turning of the tangent T(s) along the direction of the normal N(s), that is:

$$\kappa = T' \cdot N$$

*Equation 21*

It is easily to derive the first definition (equation 20) from the second (equation 21), as follows:

$$\kappa = T' \cdot N = \frac{dT}{ds} \cdot N = \lim_{\Delta s \to 0} \frac{T(s+\Delta s) - T(s)}{\Delta s} \cdot N = \lim_{\Delta s \to 0} \frac{\Delta\phi \cdot \|T\|}{\Delta s} = \frac{d\phi}{ds}$$

*Equation 22*

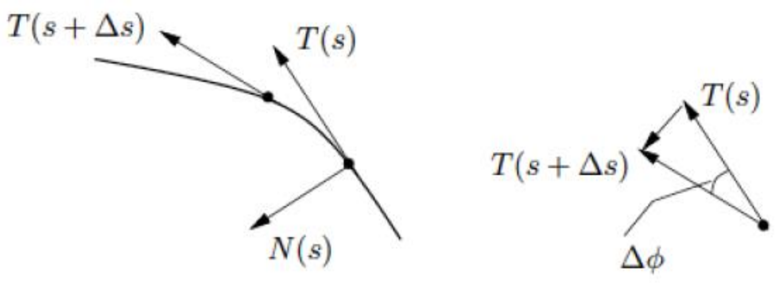


*Figure 36. Demonstration that the equation 21 can be derived from the equation 20*

To perform the measure of how sharply the curve bends, the absolute curvature of the curve at a point has been computed and it consists of the absolute value of the curvature | $\kappa$ |.

A small absolute curvature corresponds to curves with a slight bend or almost straight lines. Curves with left bend have positive curvature, while a negative curvature refers to curves with right bend. With the second definition (equation 21) it is possible defined that the curvature of a circle is the inverse of its radius everywhere. For this reason, the radius of curvature R has been identified as the inverse of the absolute value of the curvature $\kappa$ of the curve at a point.

$$R = \frac{1}{|\kappa|}$$

Equation 23

The circle with radius equal to the curvature radius R, when $\kappa$ /= 0, and positioning at the center of curvature is called *osculating circle*, as shown in Figure 35. It allows to approximate the curve locally up to the second order.

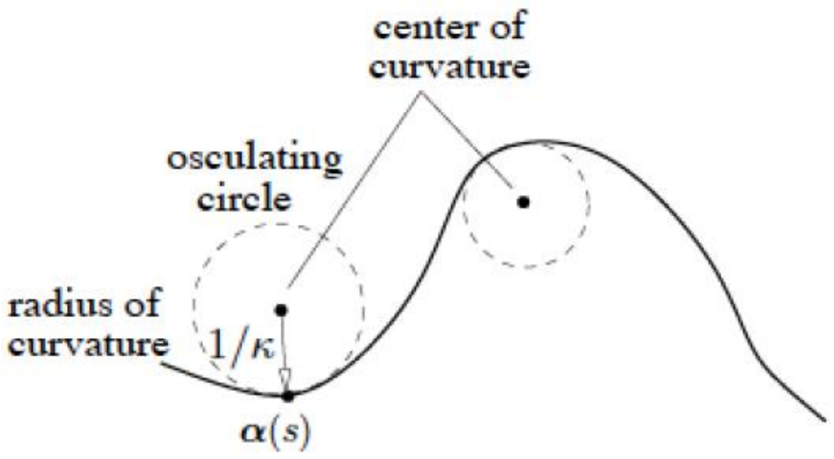


Figure 37. Osculating circle and radius of curvature

The curvature can be expressed in terms of the first and second derivatives of the curve $\alpha$ for simplicity in the computation, by the following formula:

$$\kappa = \frac{|\alpha''|}{[1 + (\alpha')^2]^{\frac{3}{2}}}$$

Equation 24

In order to compute the curvature in this thesis work, the Geom2d toolbox in MATLAB has been used. This toolbox provides the *polynomialCurveCurvature* function that allows to compute the local curvature at specific point of a polynomial curve. It receives in input the curve in parametric form x = x(t) and y = y(t) and the point in which the curvature has to be evaluate. The function polynomialCurveCurvature computes the curvature following the formula of the equation 24 that becomes:

$$\kappa = \frac{|x'y'' - x''y'|}{[(x')^2 + (y')^2]^{\frac{3}{2}}}$$

*Equation 25*

Using the parametric formula and the derivatives of the previously explained (subsection 6.1.) polynomial functions, the curvature of the trajectory can easily be obtained.

## 6.3. Lateral deviation and relative yaw angle

The last phase of the lane detection algorithm refers to the computation of vehicle model dynamic parameters. These values are necessary in order to achieve the goal of the control stage for the lane keeping. The controller has to minimize the values of lateral deviation and relative yaw angle in order to compute the optimal steering angle. Lateral deviation and relative yaw angle are defined as follow:

• *Lateral deviation* is the distance of the center of mass of the vehicle from the center line of the lane;
• *Relative yaw angle* is the orientation error of the vehicle with respect to the road.

These parameters are computed geometrically as defined in Figure 38. The relative Yaw angle is calculated as the angular difference between the longitudinal axis of the car and the tangent line to the vehicle position point of the reference trajectory. It is calculated as the slope of the reference trajectory in the coordinate origin because the longitudinal axis of the car is the X axis of the reference system. The lateral deviation will always be the distance between vehicle's CoG and the first point (x=0) of the trajectory followed.

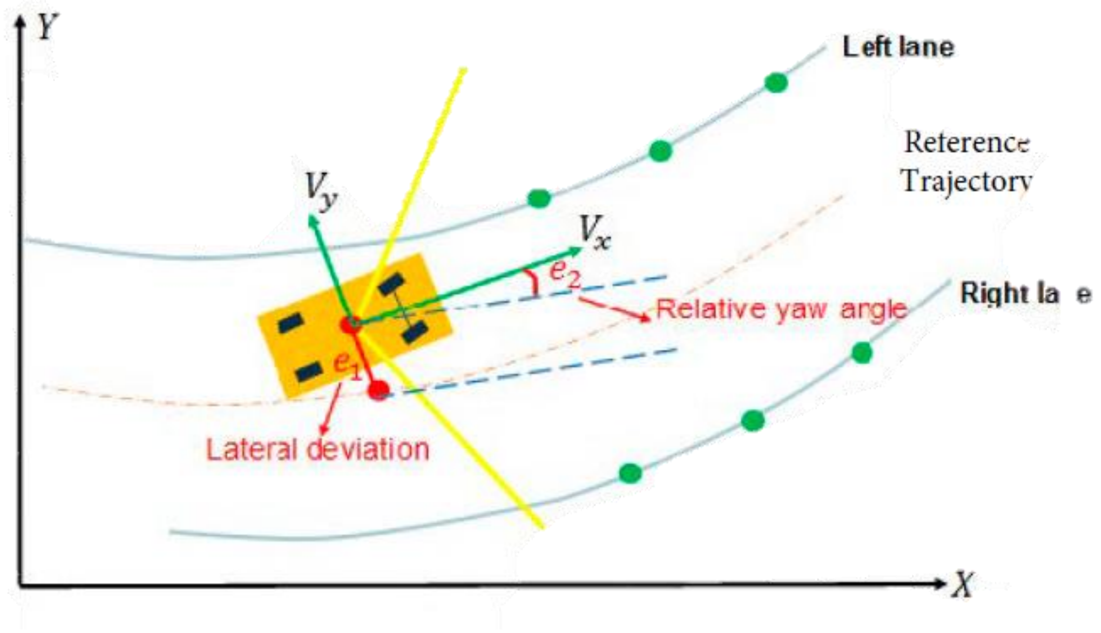


*Figure 38. Definition of lateral deviation and relative yaw angle according to the reference trajectory*

# 7. Simulation and experimental results

In this section, the simulation and the experimental results related to the trajectory generation has been presented. As mention before, lane keeping model has the aim to perform the optimal front wheel steering angle (and acceleration) in order to keep the vehicle in the computed reference trajectory and follow the curved road. The control purpose is achieved minimizing the values of lateral deviation and relative yaw angle provided by the lane detection model.

## 7.1. Simulink Implementation

In order to implement the trajectory generation and path planning algorithm in Simulink several attempts have been made. Finally, the chosen one is to first implement a MATLAB function which generated the path. From that path and other parameters the trajectory is computed in a second subsystem.

### 7.1.1. Path Planning Block

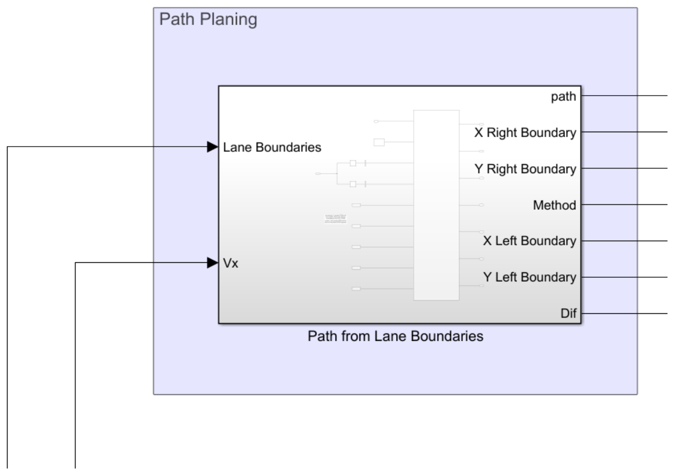


*Figure 39. Path Planning subsystem*

In the path planning subsystem, the main vehicle inputs are the lane boundaries and the longitudinal velocity. The lane boundaries are used in order to make the algorithm work and create the roadmap. The longitudinal velocity in this case is just used for the Velocity Bounding lines.

Nevertheless, there are other inputs that don't come from the vehicle model. These ones are the ones that are defining the algorithm. For the next simulations these ones have been the ones used.

- Curve Change Ratio: This ratio (in meters) defines when a part of the map is changing and stablishes when the curvy regime starts. For the case of this simulations it as been set as 0.5 meters for a tenth of the read boundaries.
- Inflation Parameter: 0.5 meter as the inflation parameter for the points in the roadmap when they come to become solids (impassable).
- Limit of the straight part ( C ): Internal parameter for the code which determines the point where the straight part should stop to make a way to the PRM algorithm. Increasing C, greater PRMs will be needed, which will make an increasing in the FULL PRM regime, and greater PRM will be done in the partial PRM regimes.
- Number of nodes: For the algorithm in these simulations 400 nodes have been using. As explained before more nodes will make more accurate paths but of a higher operational cost.

- Connection distance: It has always been used a maximum connection distance of 8 meters. If the prediction horizon or the perceptible length of the camera is decreased this parameter must change.

As outputs, there are the computed path with the PRM algorithm, the lane boundaries as 4 arrays converted into plottable data types, and two other parameters. This two are:

- Method: A parameter which indicates if the algorithm is making a straight line (center line), a partial PRM with a first straight part, or if it will compute the PRM algorithm for the full read track.
- Dif: In case the method indicates a partial reduced PRM, dif tells exactly where the straight part of the track finishes and where the curvy PRM one starts.

### 7.1.2. Trajectory Planning Block

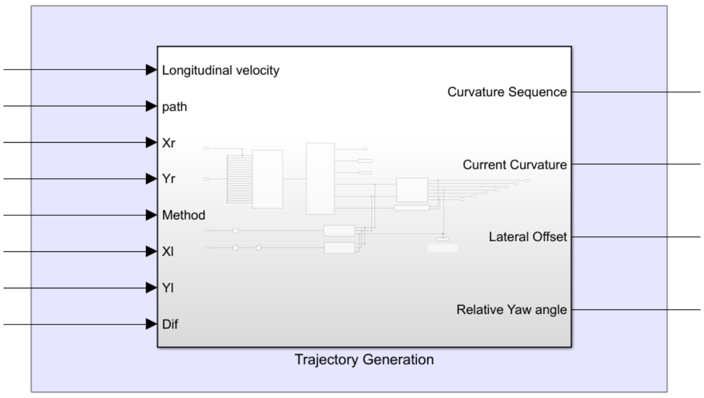


*Figure 40. Trajectory generation subsytem*

The path planning outputs and the longitudinal velocity work as model inputs here, in addition to the prediction horizon and the Ts (time step period) which are going to be used to understand how many distance of the computed trajectory will need the MPC. First the computed path is fitted into a polynomial as showed before, and after that polynomial is, within different geometric or mathematical techniques, converted into the four outputs the MPC needs.

### 7.1.3. Stability Parameter analysis Block

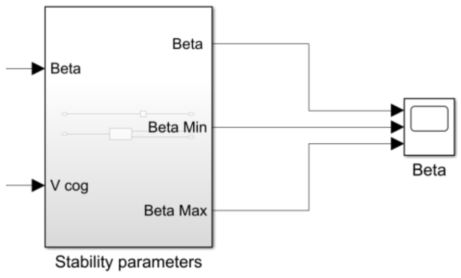


*Figure 41. Stability parameter (Beta slip angle) analysis block*

A secondary objective of the thesis was the analysis of the adherence and stability. The implementation of the *Automotive Control Systems* (Kniecke & Nielsen, 2005) algorithm for ABS is shown in Figure 42 with the longitudinal velocity.

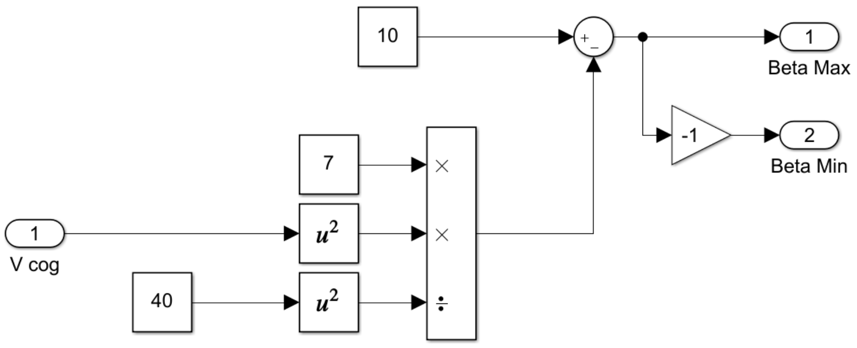


*Figure 42. ABS ACS algorithm.*

This block is going to serve as a secondary support for the analysis of the following simulations with the Betha analysis. This block just uses as inputs the longitudinal velocity for the computation of the maximum and minimum Betha, and the current slip angle from the vehicle and environment.

## 7.2. Experimental Results

Now that the algorithm parameters, and the whole input output system has been explained in the overall control layout, some experimental results are going to be shown, in comparison with the performance of following the center line instead of the computed optimized one. The following table sets the vehicle and algorithm parameters that are being used or the following simulations (whenever the method is optimized trajectory).

As it has been seen, the computed trajectories are much more changing that the center line ones. This takes a great importance in the control development, due to the fact that the MPC controller that is being used is tuned for the following of the center line. As it will be explained afterwards, a better control tuning will be necessary for the vehicle to follow correctly the reference trajectory. In order to make a noticeable comparison, a set velocity of 20 meters per second (72 kilometers per hour) is being used in the reference velocity generator, because for leading the vehicle to its limits of handling a more deep tuning in the MPC must be made, and it is not the aim of this thesis.

A scenario of a highway curve has been set in order to make this main comparison. In Figure 43 the followed trajectories of the two cases (Center line and computed trajectory). The both of them can be seen in a single plot to see the difference.

Before starting to analyze in much more detail the dynamics and performance of each trajectory generation's results it must be said that:

- Both simulations have been made in the same scenario environment.
- Both simulations have been made under the same vehicle conditions.
- Both simulations have been made with the same set velocity as a reference.
- Both simulations have been made with the same vehicle model and parameters.
- Both simulations have been made under the same adherence conditions.

In figure 43, global and not local/vehicular coordinate systems have been used in order to plot the scenario and both followed trajectories.

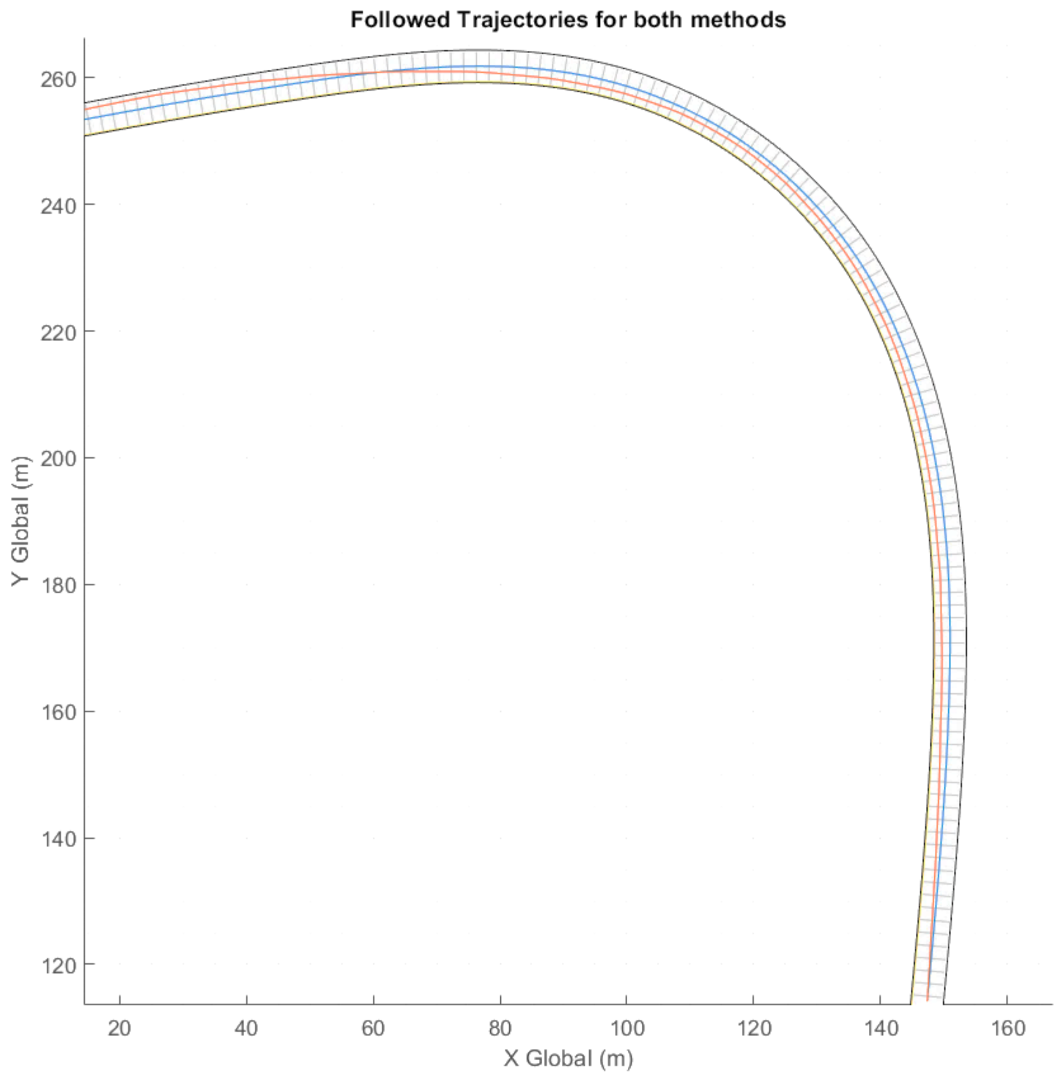


*Figure 43. Center line and optimized trajectory performances*

According to the two performances the time distance lecture is:

- The distance that is going to be considered as the reference distance the vehicle is going to go through in this scenario is of 259 meters, which is the longitude of the center line of the road.
- The optimized trajectory performance makes the distance in 15.4 seconds. This makes an average velocity of 16.81 m/s.
- The center line performance is made in 15.9 seconds. This makes an average velocity of 16.28 m/s
- The optimized performance is not doing actually 259 meters. The distance lecture indicates that the optimized performance, the vehicle travels, in fact 252.4 meters.

As a first comparison in distance and time data, it can be stated that in terms of average velocity the reference trajectory performance is 3.2% faster. Also, it goes through 6.6 meters less (a 2.61% less traveled distance). In terms of stability, the two performances are very similar, as it is shown in figures 44 a) and b). The difference, as it will be explained in more detail afterwards, is that the slip angle, even if it has a similar performance in average terms, is quite more changing in the optimized case.

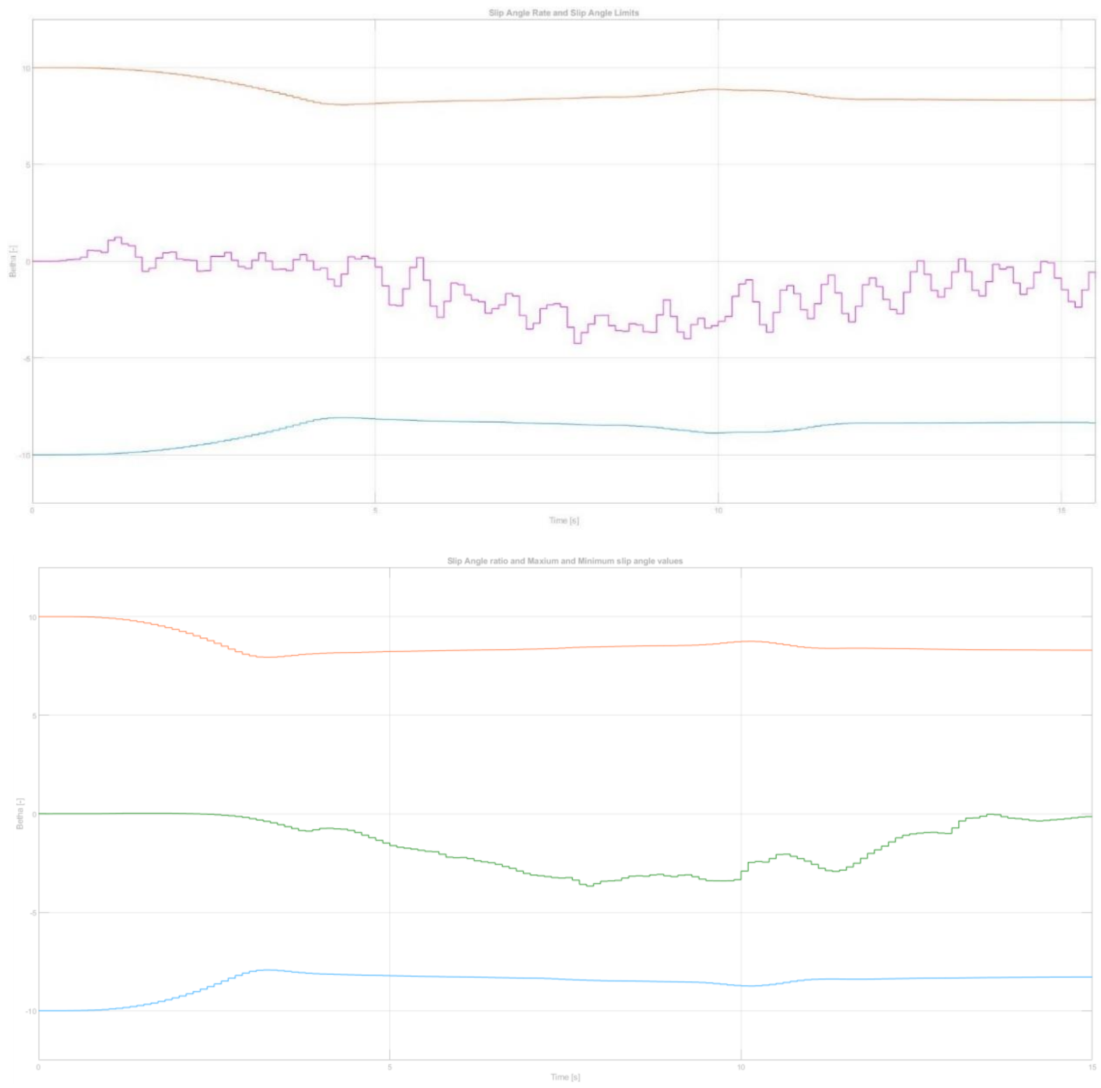


*Figure 44. a) Optimized trajectory Betha b) Center Line Betha*

The main reason why the changes are so noticeable is because, when the trajectories are being computed, the change that go through are sharp. This, as explained in the 3.2. subsection means that the changes in the control inputs will make the vehicles performance more instable, or at least, changing, inconstant and veering. This affects the vehicle's whole state and is represented either in figure 44 as in figure 45, where the optimized performance's data is shown. Nevertheless, even if veering, the Betha (slip angle) in trajectory optimization performance, is under normal values and won't make a problem in adherence.

In the two next pages the figure 45 and 46 are shown. These figures show the main data of the both performances, that are plotted in order to see how the control parameters are working as well as the understanding the performance in terms of parameters.

The first parameter is plotted in order to expose how the reference velocity generator and how the control adapts the actual vehicle longitudinal velocity to it is very similar regardless the trajectory that the vehicle is following.

More oscillations and changes are shown in the Optimized Trajectory performance because the MPC is tuned to be robust following the center line. As the center line has very low rates of changes relative to the changing computed trajectories, the curvature changing as well as the longitudinal velocity's rate of change (acceleration, which is, in fact, one of the two output of the MPC) are less alterative. This makes, also for the rest of parameters, to overcome more unstably the adaptation of the vehicle to the reference computer trajectory.

This curvature changes can be easily observable in both third graphs of figures 45 and 46. In the optimized trajectory simulation, the average curvature is the same, but the oscillations within it are very noticeable. The makes the MPC send also changing outputs to the vehicle.

Nevertheless, as the PRM algorithm and its trajectory generation creates paths/trajectories from vehicle's CoG or at least as near to it as possible, the lateral deviation in this case is quite minor, comparing it with the one in the center line simulations. It is not plotted but, taken this into consideration, e1 and e2 (lateral offset and relative yaw angles) are minor by all the ways. However, the curvature, as it has been told, is much veering, and this parameter gains importance.

As a last parameter that is quite important, the steering angle (MPC's other output to manage the vehicle) is observed in both fourth graphs. In this case, it is very observable how the MPC overreacts in order to follow the trajectory, because the changes in the steering angle aren't smooth at all. This is quite problematic due to dynamics, but also regarding the comfort of the people inside the vehicle.

Once that all the important parameters have been analyzed, the conclusions will expose which are the next steps to follow according to improve the current algorithm and, more in general ways, the performance that the vehicle will make.

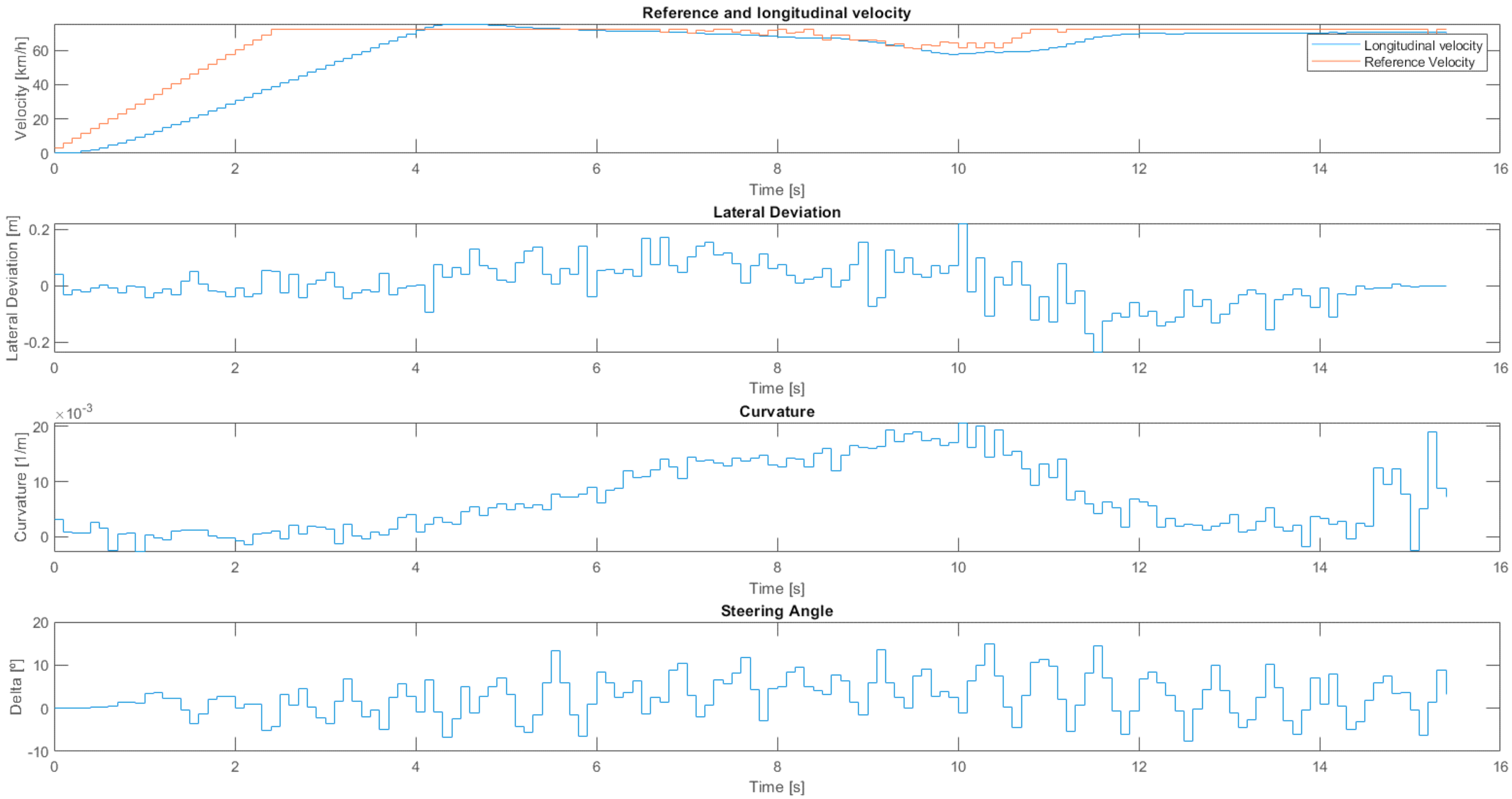


*Figure 45. Parameters for the Optimized trajectory perrformance*

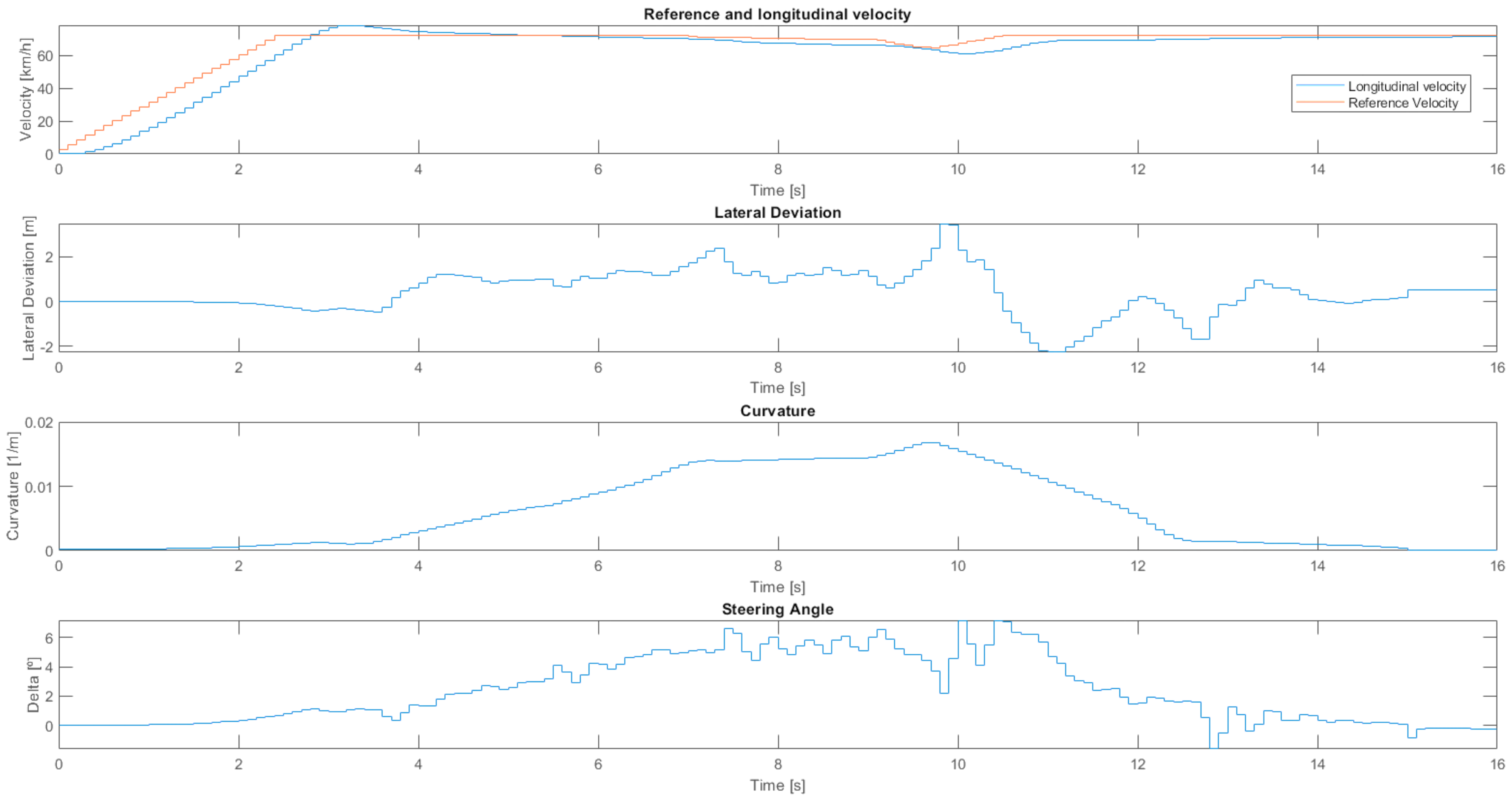


*Figure 46. Center Line performance parameters.*

# 8. Conclusions and future works.

In this work the whole design and implementation of a path planning and trajectory generation algorithm has been exposed into a control layout of an autonomous vehicle's control. The tuning of the algorithm, its construction and its parts have been fitted in an MPC controlled autonomous car, and some experimental results have been explained.

The algorithm is working well and in the closed loop the results have been demonstrated once the MPC have been put to work with the new trajectory generator. Nevertheless, the trajectories made aren't the optimal ones in two ways as it has been seen.

The first reason why the aren't optimal is because just a few geometrical considerations have been made according to vehicle dynamics (Velocity Bounding Lines), but the optimal trajectory is not always the shortest one. Some improvements should be made on the simple algorithm in order to take into more consideration vehicle dynamics, steerability and others.

The second reason is no due to the generation of trajectories, but about the consecution of different ones in time. The change rate of initial curvatures for each consecutive curvature is very high. This becomes a problem when the outputs of the MPC sent to the vehicle are very veering. This makes the steering angle to vary more than it should. This gets a result not only in maneuverability, but in the reference velocities and in other errors. These two problems are surmountable if the path planning gets smother initial paths, and more similar between the consequent ones.

Nevertheless, even if the algorithm is not perfect, regarding the fact that it is working with a MPC that has been tuned for a much more smooth trajectory generator (the center line has les changes) the trajectory generator is making a performance that is faster than the past ones, going through less distance and giving a better performance over the scenario. If the control is tuned in a more robust way according to curvature changes, even if the algorithm stays without improves (as it has been set in this thesis) the results that are going to be obtained are being much more efficient, fast and secure.

The developed algorithm is adaptable to different regimes, and also is compatible with the center line trajectory, adapting between both generation method whenever it is needed. It can be set a rule to change from one method to another whenever, for example, a urban scenario becomes interurban or highway regime. Also, it has no problem of being subject to control, scenario, vehicle model or simulation environment changes, because it has been designed in a modular way. This means that if the control is updated the algorithm and the trajectory generation will also be useful. Always that the lane boundaries are recognized the algorithm will make its work with a few schematic changes.

This way, similar path planning and trajectory generation environments can be used in different control diagrams due to is easy adaptation. This comes, principally, because the path planning algorithm is a geometrical algorithm, and the trajectory generation based on paths is done with numerical methods. This leads to a easy implementation in changing environments, other vehicles, and it is also subject of several improvements.

In summary, a method has been created that, although preliminary and not optimal, serves as support for the work of the LIM, and opens up a new range of possibilities in order to continue improving and making contributions to this prominent field of engineering.

## References

Curinga, T. (2018). Autonomous racing using Model Predictive Control.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs.

Kavraki, Svestka, & Latombe. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces.

Khan, I. (2019). Model Predictive Control based lateral and longitudinal controller for autonomous driving. In I. Khan.

Kniecke, L., & Nielsen, U. (2005). *Automotive Control Sytems (Second Edition).*

Liniger, A. (2014). Optimization-based autonomous racing of 1:43 scale RC cars.

Mancuso, T. (2018). Study and implementation of lane. In T. Mancuso.

Qin, S., & Badgwell, T. (n.d.). *A Survey of Industrial Model Predictive Control.*

Rutter, J. W. (2000). *Geometry of curves.*

Xu, Yang, & Shao. (2007). Yang.