Local Path Planning with Moving Obstacle Avoidance based on Adaptive MPC in ATLASCAR2

a Master Thesis presented
by
Alberto Franco
to
The Department of Information Engineering

in partial fulfillment of the requirements
for the Second Cycle degree in
Automation Engineering

Università degli Studi di Padova
Padova, Italia
15th April 2019

Povera mente.
Io ti uccido ogni giorno con le mie idee.
Povero cuore.
Io ti metto alla prova ma povero me.

Acknowledgments

dedica a parenti e amici

The research work for this thesis was carried out at the Laboratory for Automation and Robotics (LAR) in the Department of Mechanical Engineering of Aveiro (Portugal), during a period of 5 months as an exchange student.

I would like to express my gratitude to Professor Vitor Santos, for welcoming me at the Laboratory for Automation and Robotics and making possible my experience of study at the University of Aveiro. His help has been invaluable both personally and academically. Moreover I would like to thank Professor Angelo Cenedese for his advices and his help during this work.

Abstract

Inserted in the ATLASCAR2 project this work aims to develop a short-term path planning framework for driver assistance in dynamic environments. In order to achieve this objective, it was made a preliminary study of the existing local path planning methods and the projects that have already been developed in this field, their advantages and disadvantages were weighed and the most successful approaches applied to local navigation in real autonomous driving projects were taken into account. This thesis presents two different strategies for a self-driving car short-term path planning among multiple moving obstacles. The main task is to study and implement a motion planning and execution framework in order to make ATLASCAR2 coexist with other moving obstacle vehicles by avoiding collision and overtake them when necessary and possible. The first method developted, is an obstacle avoidance system that moves the vehicle around different moving obstacles while the second algorithm is a lane following system that keeps the ATLASCAR2 traveling along the centerline of the lanes on the road. The proposed techniques, based on the adaptive Model Predictive Control paradigm, solve optimization problems formulated in terms of cost minimization under constraints. Simulation results, developted in a MATLAB/Simulink enviroment, demonstrate and verify the feasibility and the usefulness of methods considering different scenarios, opening space for real scenario implementation.

Contents

"Self-driving cars are the natural extension of active safety

and obviously something we thing we should do."

Elon Musk

1

Introduction

In robotic research, the problem of navigation is among the most important. Basically, all autonomous mobile robots need some kind of navigation abilities to perform, localization, motion planning and guidance [1]. In the present context, we focus on navigation as a process of planning a path of a mobile robot from its current position to a desired goal location, following the planned path, and avoiding any discovered obstacles along the way. The desired paths have to fulfill several conditions

to ensure safety and feasibility of the navigation. Moreover, the paths can be also defined in terms of specifications; for example, short or smooth paths are usually more desirable than long and curved ones in every dynamic environments. Beyond the path planning, the navigation problem also involves reacting to changes of the environment model. Robots are required to move towards the target in a short time and avoid either static or dynamic obstacles observed by their sensors, which involves efficient path planning and valid obstacle avoidance. Research and development of Unmanned Ground Vehicles has been dominated by DARPA (Defense Advanced Research Projects Agency) and NASA (National Aeronautics and Space Administration). The DARPA initiative started with the development of the first mobile robot, Shakey, and also includes the Autonomous Land Vehicle and the DARPA Demo II Program. NASA sponsors the development of unmanned vehicles for planetary surface exploration, from the Jet Propulsion Laboratory Mars Rover to the most recent Mars Pathfinder. Recent UGV design and development has been enhanced to build UGVs capable of operating in Intelligent Vehicle Highway Systems [2]. Over the last decades, the development of Advanced Driver Assistance Systems has become a critical endeavor to attain different objectives: safety enhancement, mobility improvement, energy optimization and comfort [3]. Much of the argument used in this discussion is based on the road mortality we see today. According to data from the World Health Organization, in 2013 there were about 1.25 million road deaths worldwide, and this number is expected to increase in the next decade [4]. Algorithms for autonomous navigation are increasingly robust and reliable and are starting to handle complex situations and decision problems. According to Katrakazas [5], local navigation is responsible for guiding the vehicle, that is, for the planning of the direction and speed to be taken, in a space close to the current

position based on information exclusively obtained by the sensors on board. The guiding must be planned in such a way as to guarantee the displacement, from the current state to the objective, without collisions. The algorithms we have developed at the LAR, follow a new and different approach for an advanced control strategy for autonomous navigation. The idea is that these methods do not replace the algorithms developed previously but are a valid alternative, so that depending on the situation, the vehicle can choose the best strategy to overcome obstacles or solve problems that can occur on the road that need a decision in real time. Simulation results demonstrate and verify the feasibility and the usefulness of methods considering different scenarios.

In this introductory chapter, the ATLAS project is presented in more detail in section 1.1, while examples of autonomous navigation projects are discussed later (section 1.2). The context of the problem and the proposed solution of this thesis are carried out in section 1.3 while the organization of the document is discussed in section 1.4.

1.1 ATLAS Project

The ATLAS project was created in 2003 by the Group for Automation and Robotics from the Department of Mechanical Engineering at the University of Aveiro [6]. The objective of this project is to study and develop advanced sensors and active systems to promote the autonomous control of cars and other platforms. The first projects in the autonomous driving area focused on small scale models in controlled environments for participation in the National Robotics Festival (FNR) and in many other competitions winning some awards for the best performance (subsection 1.1.1). The

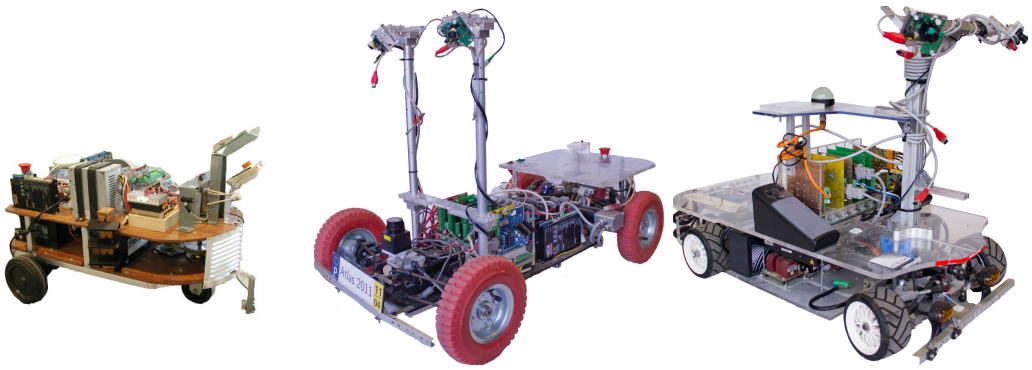
success and experience gained with these models allowed the evolution of the project for full-scale vehicles, where ATLASCAR (subsection 1.1.2) in 2010 and ATLASCAR2 (subsection 1.1.3) in 2016 have been developed.

1.1.1 ATLAS platforms

The first developed robot (Figure 1.1a) was based on an aluminum and wood structure. In this prototype only one camera was installed that pointed to a mirror to allow the complete visualization of the road in which the robot circulated. The traction movement was assured by a mechanical differential coupled to the rear wheels and the steering movement was given by a single front wheel. In order to create a model more similar to an ordinary car, the ATLAS group developed the ATLAS 2000 (Figure 1.1b) in scale (1:4), with which it managed to win the first autonomous driving competition of the FNR in 2006. After several improvements made in ATLAS 2000, in 2008 a new platform, ATLAS MV (Figure 1.1c) was created. This robot was designed on a smaller scale (1:5), with the intention of being lighter and faster. On board were installed a new steering system, hydraulic braking and an active perception unit. This robot allowed the conquest of new victories in the autonomus driving competitions.

1.1.2 ATLASCAR

Driven by the positive results achieved with scale models and years of navigation experience in controlled environments, in 2010 the Group of Automation and Robotics decided to invest in a large-



(a) First ATLAS prototype. **(b)** ATLAS 2000. **(c)** ATLAS MV.

Figure 1.1: Some of the ATLAS project small scale platforms (adapted from [7]).

scale project, ATLASCAR (Figure 1.2). The vehicle used for this project was a Ford Escort Station Wagon of 1998 powered by a gasoline internal combustion engine, in which several sensors, processing units and actuators were installed. On-board sensors processed data collected from the vehicle and its surroundings, with different LIDARs for obstacle detection and environmental reconstruction, pedestrian detection cameras and a Global Navigation Satellite System (GNSS) for location and route planning. After passing through the processing units, these data were sent to the actuators that allowed the movement and execution of the maneuvers in a completely autonomous way on the part of the vehicle. To power all the equipment, a Uninterruptible Power Supply (UPS) was used, loaded from an auxiliary alternator. During this project, many works were developed in the Laboratory for Automation and Robotics, many of which produced master thesis. For example, in 2014 Cabral de Azevedo [8] developed a module to detect pedestrians using sensory fusion of LIDAR and vision data while in 2016 Vieira da Silva [9] created a multisensory calibration module that was exported to subsequent projects.

Figure 1.2: The car used in ATLASCAR, based on Ford Escort Station Wagon of 1998 (adapted from [7]).

1.1.3 ATLASCAR2

Given the different limits, to continue the project, in 2016, a new vehicle was acquired: ATLASCAR2 (Figure 1.3). This time it was chosen as a platform an electric car, a Mitsubishi iMiEV, with an autonomy range of 100 km. The fact that the vehicle is electric allows to use the energy stored in the batteries, making it easier to power the sensors installed. In fact, despite the short time of existence, 3 LIDARs, a camera, inclinometry sensors and a GNSS unit are already installed on the ATLASCAR2. Many of these sensors were transferred from ATLASCAR to this project during the work of Madureira Correia [10] in 2017, where a module for detecting free space around the car was also developed while in 2018 Ricardo Silva [11] created a local navigation module for driver assistance in the immediate decision making, identifing a solution based on a multiple hypothesis approach.

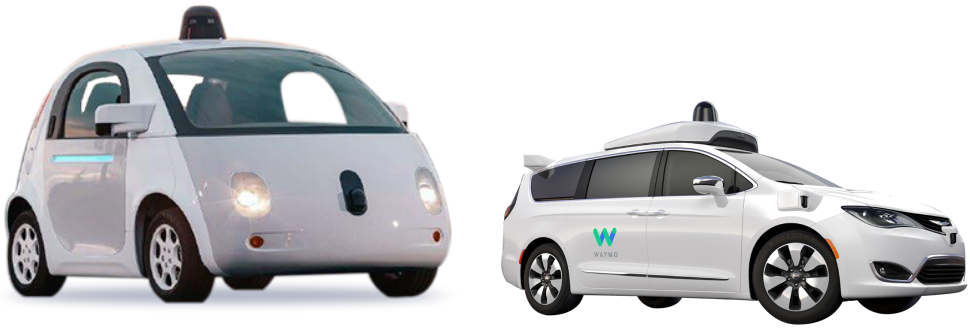
Figure 1.3: The vehicle used in ATLASCAR2, based on an electric car, a Mitsubishi iMiEV of 2015 (adapted from [11]).

1.2 Autonomous Cars

The legal definition of autonomous vehicle in the District of Columbia code [12] is:

> *"Autonomous vehicle" means a vehicle capable of navigating District roadways and interpreting traffic-control devices without a driver actively operating any of the vehicle's control systems. The term "autonomous vehicle" excludes a motor vehicle enabled with active safety systems or driver-assistance systems, including systems to provide electronic blind-spot assistance, crash avoidance, emergency braking, parking assistance, adaptive cruise control (ACC), lane-keep assistance (LKA), lane-departure warning, or traffic-jam and queuing assistance, unless the system alone or in combination with other systems enables the vehicle on which the technology is installed to drive without active control or monitoring by a human operator.*

The modern automobile companies keep coming up with newer autonomous features in their recent models. Technological advancements seen every day in areas like information technology, communication, data analysis and storage etc. is not exclusive to these areas alone. The realm of autonomous cars is also progressing at a rapid rate these days [13]. Google's development of self-driving technology began in January 2009. The initial objective of the project was to develop a car able to navigate on highways with minimal human intervention. In December 2016, the unit was renamed Waymo; this name derived from its mission, "a new way forward in mobility". Waymo moved to further test its cars on public roads after becoming its own subsidiary.



(a) Google's Firefly self-driving prototype in 2015 (adapted from [14]).

(b) Waymo Chrysler Pacifica Hybrid self-driving prototype in 2017 (adapted from [14]).

Figure 1.4: Some of the Waymo/Google prototypes tested across multiple locations in the Unided States in recent years.

Waymo uses LIDAR which sends out millions of laser beams per second to build up a detailed picture of the world all 360 degrees around it. It also uses radar to detect how far away objects are and their speed and high-resolution cameras detect visual information like whether a traffic signal is red or green. It then combines all that data to understand the world around it and predict what

those things might do next. It can do that for things up to three football fields away. Based on all this information, Waymo's software determines the exact trajectory, speed, lane and steering maneuvers needed to progress along this route safely [14] [15].

With the advances in autonomous technology, VIAC or VisLab Intercontinental Autonomous Challenge was one of the major competitions which led to improvements in the testing and analysis of autonomous vehicles and robotics. It was a 13,000 kilometers trip, nearly three months from Parma, Italy to Shanghai, China from July 20, 2010 to October 28, 2010. It involved four autonomous vehicles with negligible human intervention and high level of autonomy [16]. One of VisLab's advanced autonomous car, BRAiVE (Figure 1.5), drove in downtown Parma on July 12th, 2013. It successfully navigated narrow rural roads, crosswalks, traffic lights, pedestrian areas, roundabouts and artificial hazards. It was a pioneer in the field of vehicular robotics, since it was totally autonomous [17].

Figure 1.5: BRAiVE prototype developted by VisLab, based on a Hyundai Sonata (adapted from [17]).

Another example of autonomous system is Navya, a robotically driven electric shuttle which operates at a maximum speed of 25 kilometers per hour. Made by Induct Technology, France, it

can accommodate 15 passengers. It uses four LIDAR units and stereoscopic optical cameras, and it does not require any road modifications. Its LIDAR unit and optical cameras help in generating a real-time three dimensional map of the surroundings. It is being successfully tested at various universities across Switzerland, England and Singapore [18].

Figure 1.6: Navya Shuttle developted in 2016 by Navya Group in France.

1.3 Context of the Problem and Proposed Approach

Dynamic environments pose several added difficulties to the motion planning problem. The dynamics of the ATLASCAR2 must be taken into account, and there are limitations due to the sensors range and uncertainty in measurements, that must be reflected on the motion plan. Besides it, a motion plan must incorporate time restrictions, meaning the vehicle will require a certain amount

of time to accomplish a task. For example, when crossing a road, the ATLASCAR2 must do it fast enough to avoid incoming cars. The proposed algorithms were studied for the ATLASCAR2 project in which the group for Robotics and Automation at the University of Aveiro has setup and adapted a common commercial electric vehicle to provide a versatile framework to develop studies and research [6] [19]. The fact that the vehicle is electric allows to use the energy stored in the batteries, making it easier to power the sensors installed. In fact, the ATLACAR2 is equipped with sensors, such as lidar, that measure the distance to obstacles in front and around the vehicle. The obstacles can be static, such as a large pothole, or moving, such as a moving vehicle on the same or a nearby lane. The most common maneuver from the driver is to temporarily move to another lane, drive past the obstacle, and move back to the original lane afterward. In this case, we want to design an obstacle avoidance system that moves the ATLASCAR2 around moving obstacles in the lane using throttle and steering angle. This system uses an adaptive Model Predictive Controller that updates both the predictive model and the mixed input/output constraints at each control interval. Moreover we want to develop a lane-keeping assist system for the vehicle: it has a sensor, such as camera or laser, that measures the lateral deviation and relative yaw angle between the centerline of a lane and the ATLASCAR2; it also measures the current lane curvature and its derivative. Depending on the curve length that the sensor can view, the curvature in front of the vehicle can be calculated from the current curvature and its derivative. This system keeps the autonomous car travelling along the centerline of the lanes on the road by adjusting the front steering angle. The goal for lane keeping control is to drive both lateral deviation and relative yaw angle close to zero.

1.4 Thesis Outline

In this section, we outline the thesis organization:

Chapter 1 is used to introduce the thesis focus areas of autonomous vehicle technology. In particular the ATLAS project and examples of autonomous navigation projects are presented. Moreover the context of the problem and the objectives to be achieved are carried out.

Chapter 2 focuses on some of the more theoretical aspects of this dissertation with a brief introduction to path planning, a literature review related to local navigation algorithms with a special section centred on MPC control strategy for obstacle avoidance and tracking.

In Chapter 3, the theory of Model Predictive Control is discussed in detail to highlight working principle and its characteristics (tuning parameters, stability and robustness). In particular for this work we used an advanced control strategy based on this paradigm called Adaptive MPC that uses a fixed model structure, but allows the model parameters to evolve with the time.

Chapter 4 is used to present the Obstacle Avoidance method that we have developted. First we introduced the case-study model, then we designed the decision making alghorithm and the controller. Finally we verified the control strategy in different scenarios.

Chapter 5 focuses on the Lane Following alghoritm based on Adaptive Model Predictive Control. We considered a different model in which we have applied the bicycle model of lateral vehicle dynamics and approximated the longitudinal dynamics using a first order transfer function. Then we developed the overall control scheme considering different paths to follow.

2

Literature Review

In this chapter we introduce the literature survey of the various algorithms used for robot navigation. The navigation methods are divided into two kinds of control: global path planning and local motion control. First we analyze these two types of strategies and then we describe the current situation of Model Prediction which is the control method that we have adopted.

2.1 Global Path Searching Method

Global path planning uses information about a priori model or a map of the enviroment to evaluate the shortest path that allows the motion from a starting point to the goal position. There are a lot of path planning algorithms, like cell decomposition, road map or potential field, where the calculation of a complete trajectory from a start position to one or more goal positions can be computed off-line. However they produce a reliable path if and only if the map of the enviroment is already available. So, this prior knowledge of the space where the robot can move, must be accessible. The most famous method developed for global navigation is the Dijkstra algorithm [20]. In recent years, an evolution of this method called A* is one of the most used [21]: this algorithm gives a complete and optimal global path in static enviroments. However, it was upgraded in D* [22] for efficient online searching of a dynamic enviroment, which gives sequences of path points in the known or partially known space. The basic idea of these two methods is the following: minimizing their cost functions, these strategies have the capability of a quickly redesign when the conditions of the space change, to guarantee an optimal solution of the trajectory from the starting point to the goal location. Moreover it has been shown that the D* algorithm is much more efficent than the A* method. Now we will analyze the general structure of these two algorithms and the practical operation based on a representation of an environment grid around the vehicle. However these methods are not sufficient to create a complete robot system navigation. We will see later that if the information of the map is not yet available, the system has to use onboard sensors to define a path and avoid obstacles.

2.1.1 The A* Algorithm

The algorithm A*, based on a graph representation, examines, step by step, the nodes that have the best score. A* uses the following data structures to keep track of the execution status: the first one is a list of nodes already visited while the second one is a priority queue containing the nodes to be visited. During execution, each node is associated with multiple values: gScore, hScore, fScore. In mathematical terms, given the current node n, the starting node p and the solution node s, we define the values:

- $gScore = g(p, n)$ that evaluates the actual cost of the path that separates the p (start) and n (current) nodes.
- $hScore = h(s, n)$ that calculates an estimate of the cost of the path between the s (solution) and n (current) nodes; this is also called Euristic function because it is associated to an Euristic algorithm that allows A* to find rapidly a solution.
- $fScore = gScore + hScore$ which is the total cost.

The structure of the A* method can be summarized in 8 steps as follows:

1. Insertion in the queue of the starting node with priority equal to the fScore;
2. If the queue is empty, the algorithm returns that the solution cannot be found;
3. Extraction of the best node to visit (priority with lower value);
4. If the extracted node has zero $hScore$, the algorithm terminates: solution found;
5. Creation of child nodes;
6. Deletion of child nodes already visited and suboptimal;
7. Inserting the remaining nodes in the queue with priority equal to the fScore;
8. Return to step 2.

Finally it is possible to state that the major limitation of this algorithm is in the absence of constraints on the search depth.

2.1.2 The D* Algorithm

Before describing how the D* algorithm works, we have to introduce how the nodes are marked (infact also this method mantains a list of nodes):

- NEW, it has never been placed on the list;
- OPEN, it is currently on the list;
- CLOSED, it is no longer on the list;
- RAISE/LOWER, its cost is higher/lower than the last time it was on the list;

This method works by iteratively choosing a node from the list and evaluating it; then the node's changes are propagated to all of the neighboring nodes and they are placed on the list. This first process is called expansion. In particular we can emphasize that the D* algorithm begins by searching backwards from the goal node. After being expanded, each node has a backpointer which refers to next node leading to the target where the exact cost is known by each node. When an obstacle is detected along the path, all the points that are thwarted, are again placed on the list marked as RAISED. Hence, the method examines RAISED's neighbors to try to reduce the node's cost; otherwise it increases the cost of the RAISED node. If not, the nodes' descendants, which have backpointers, are marked as RAISE. These nodes are calculated and the RAISE state is propagated forming a wave. If it is possible to reduce a RAISED node, its backpointer is updated and the LOWER state is

passed to its neighbors. These two waves cannot touch other points so the algorithm worked only on the nodes that are affected by the cost change.

2.2 Local Motion Control

Local Motion Control is related to the real-time motion of the robot inside in unknown enviroment, where monitoring with the sensors, it can detect where are the obstacles and create a motion to avoid the collision with them. One of the advantages of the local navigation systems is the ability of generating a new path every time the space changes, for example when multiple moving obstacles are identified thanks to the sensors that captured the information in the enviroments. These methods can be divided into directional and velocity space-based approaches.

The most famous directional approaches (generate a direction for the robot/vehicle) are:

- Potential field method [23], where the robot is represented as a particle and it is subject to forces that are produced by the surrounding environment;
- Virtual Force Field which expands to Vector Field Histogram [24], where it utilizes a statistical representation of the vehicle's space through the so-called histogram grid;
- Nearness Diagram algorithm [25], which performs a high level information extraction and interpretation of the environment, used to generate the motion commands;

while the velocity space approaches, that manage the robot/vehicle considering translation and rotation velocities, are:

- Curvature Velocity method [26] which formulates the problem as one of constrained optimization in velocity space;

- Dynamic Window method [27] which is divided in two main phases; in first one, it generates a valid search space while in the second one it selects an optimal solution in the search space.

Now we will analyze the general structure of some of these algorithms. In particular it is important to underline that to create a complete navigation system it is necessary that there are both local algorithms and global methods such as those seen previously.

2.2.1 Potential Field Method

The first local method that we analyzed is the Potential Field; in this strategy the robot/vehicle is represented as a particle that moves in the workspace and it is affected by attractive and repulsive forces that are generated by the surrounding environment. In particular the target propagates an attractive force while the obstacles transmit a repulsive forces. The latter can be evaluated considering only the distance from the obstacles and the instantaneous vehicle velocity and accelerations. Combining these two forces it is possible to compute the trajectory of the robot at every time instant. The total force can be used as input to control the trajectory of the vehicle.

2.2.2 Curvature Velocity Method

Another obstacle avoidance problem can be solved applying the Curvature Velocity Method where the problem becomes a constrained optimization in the velocity space of a syncro-drive steered robot. This strategy tries to maximize the following function:

$$f(tv, rv) = \alpha_1 \text{speed}(tv) + \alpha_2 \text{dist}(tv, rv) + \alpha_3 \text{head}(rv) \tag{2.1}$$

where tv and rv are respectively the robot's translational and rotational velocities while speed(tv) represents the speed of the vehicle, head(rv) is the heading of the robot and dist(tv, rv) is distance to a set of obstacles within a given limit. These last three terms can be expressed as follows:

$$\text{speed}(tv) = \frac{tv}{tv_{\text{max}}} \qquad \text{head}(tv) = 1 - \frac{\theta_g - rvT_c}{\pi} \qquad \text{dist}(tv, rv) = \frac{D_{\text{limit}}(tv, rv, OBS)}{\text{limit}}. \tag{2.2}$$

At each sampling time T_c, the constrained optimization problem is solved choosing the transational and rotational velocities to adjuste α_1, α_2 and α_3 values. In order to limit the search space of possible curvatures of the vehicle in velocity space, the value D_{limit} is bounded and discretized. Finally we can classify these intervals in disjoint, contained by, contains or overlapping and they are fundamentals to compute optimal values for the distance function and consequently for the equation (2.1).

2.2.3 Dynamic Window method

The last obstacle avoidance method that we have analyzed is called Dynamic Window approach; this is a velocity-based local planner that evaluates the optimal speed for a robot rquired to reach the target without colliding. In this algorithm the cartesian goal is translated into a velocity command for a vehicle. First, we can evaluate the desired velocity to the target based on our current position, and the destination; then we select the ammissible linear and angular velocities given the robots dynamics. From all the allowable velocities we determine the closest obstacle for the proposed vehicle velocity. If the distance to the nearest obstacle is within the robots breaking distance and the vehicle is not able to stop in time, we will reject this proposed robot speed. Otherwise, we can evaluate

the required values for the cost function. If the calculated cost is better than anything else so far, then we set this as our best option. Finally, the desired trajectory is set to the best proposed velocity. Hence, this method calculate a valid velocity search space and select the best speed that maximizes the robots clearance and obtains the closest heading to the target.

2.3 MPC in Autonomous Driving

The algorithms we have developed in this project have a different approach than those mentioned above. In this section we have reported some algorithms based on the MPC used in the field of autonomous navigation. This literature review was fundamental to understand what methods exist and the strategies adopted to solve certain problems. In [28] the authors addressed the problem of real-time obstacle avoidance on low-friction road surfaces using spatial Nonlinear Model Predictive Control (NMPC). They considered a nonlinear four wheel vehicle dynamics model that includes load transfer and they also used the ACADO Code Generation tool to generate NMPC algorithms based on the real-time iteration scheme for dynamic optimization. An obstacle avoidance problem solved with an MPC approach that integrates the artificial potential field method is discussed in [29] where the controller is combined with the feedback linearization in order to manage the control problem of a single robot with unicycle kinematics and collision avoidance function. However in many traffic emergency situations a collision cannot be prevented by braking alone. This is the reason why in [30] the authors proposed an obstacle avoidance method based on the NMPC paradigm that simultaneously optimizes steering and braking. In [31] a path planning and tracking

framework is considered to maintain a collision-free path for autonomous vehicles. In particular to track the planned trajectory, the proposed controller formulated the tracking task as a multiconstrained model predictive control (MMPC) problem and evaluated the steering angle to prevent the robot from colliding with a moving obstacle vehicle. Finally we considered another driver assistance method for obstacle avoidance based on constrained model predictive control. The algorithm also in this case was reduced to a convex quadratic programming problem. [32]. All these methods, based on the MPC paradigm, allowed us to define two different situations where the use this type of controller is required. The first situation we have faced is a collision avoidance problem considering moving obstacles. In the second scenario we have instead built a system that allows tracking of the lane. These two approaches, faced earlier, have been solved through a new type of MPC called Adaptive MPC.

3

Model Predictive Control

In this chapter, the theory of Model Predictive Control is discussed in detail to highlight working principle. In particular for this work we used an advanced control strategy based on this paradigm called Adaptive MPC that uses a fixed model structure, but allows the model parameters to evolve with the time.

3.1 Generic Model Predictive Control problem

Model Predictive Control (MPC), also known as Moving Horizon Control (MHC) or Receding Horizon Control (RHC), is a popular method for the control of slow dynamical systems, to generate the required control inputs that are calculated at each sampling instance k, using the current state as initial conditions to solve a finite optimal control problem. Some of the advantages of using MPC are:

- the ability to handle unstable, time variable, non-minimum phase systems;
- robustness feature with the uncertainties in the nonlinear systems;
- built in feed-forward control to handle disturbances in the processes;
- enhanced tuning features to achieve the best response including transient responses;
- the possibility to introduce constraints in a natural form;
- if the references are known in advance, they can be used in order to optimize the reference tracking.

The methodology of all the controllers belonging to the MPC family is characterized by the following strategy, represented in Figure 3.1. The future outputs for a determined horizon, called the prediction horizon, are predicted at each instant k using the process model. These predicted outputs depend on the known values up to instant k (past inputs and outputs) and on the future control signals which are those to be sent to the system and calculated. The set of future control signals is calculated by optimizing a determined criterion to keep the process as close as possible to the reference trajectory. This criterion usually takes the form of a quadratic function of the errors between

the predicted output signal and the predicted reference trajectory. The control effort is included in the objective function in most cases.

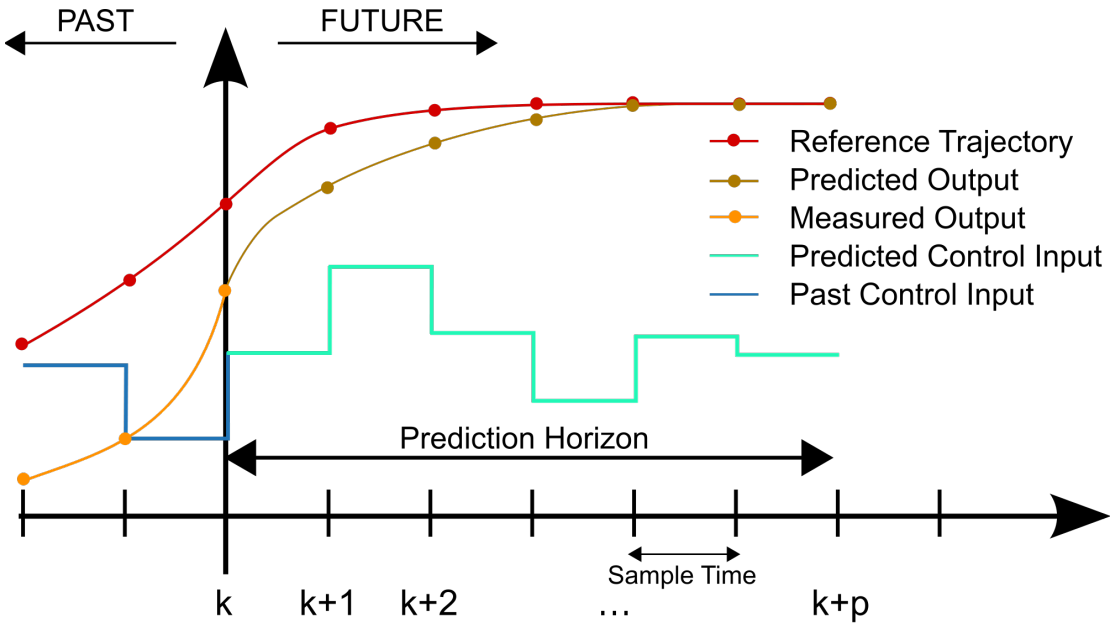


Figure 3.1: A discrete Model Predictive Control scheme adapted from [33].

An explicit solution can be obtained if the criterion is quadratic, the model is linear, and there are no constraints; otherwise an iterative optimization method has to be used. Some assumptions about the structure of the future control law are also made in some cases, such as that it will be constant from a given instant. Only the current control signal is send to the process. At the next sampling instant the measured output is evaluated and the sequence is repeated and all the steps brought up to date. Thus the predicted control input is then calculated using the receding horizon concept.

MPC is typically formulated in the state space form. For a given discrete linear time-invariant

(LTI) system:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \tag{3.1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$, $\boldsymbol{u}(k) \in \mathbb{R}^m$ are the state and the input, respectively. The central idea in the Model Predictive Control is to minimize some cost function, while still ensuring that some constraints are fulfilled. The generic MPC problem can be written as follows:

$$\begin{aligned}
\underset{\mathrm{u}}{\text{minimize}} \quad & J(\boldsymbol{x}(k), \mathrm{u}) \\
\text{subject to} \quad & \boldsymbol{x}_{k+i+1} = \boldsymbol{A}\boldsymbol{x}_{k+i} + \boldsymbol{B}\boldsymbol{u}_{k+i} \quad \forall i = 0, \dots N-1; \\
& \boldsymbol{x}_{k+i} \in \mathbb{X} \quad \forall i = 0, \dots N-1; \\
& \boldsymbol{u}_{k+i} \in \mathbb{U} \quad \forall i = 0, \dots N-1; \\
& \boldsymbol{x}_{k+N} \in \mathbb{X}_f; \quad \boldsymbol{x}_k = \boldsymbol{x}(k).
\end{aligned} \tag{3.2}$$

where $\mathrm{u} = (\boldsymbol{u}_k, \dots, \boldsymbol{u}_{k+N-1})$ is a sequence of control inputs, $\boldsymbol{x}_{k+i}$ is the state at time $k+i$ as predicted at time k, and N is the prediction horizon. The sets $\mathbb{X} \in \mathbb{R}^n$ and $\mathbb{U} \in \mathbb{R}^m$ define the constraints on the state and the input, respectively. Finally, the set $\mathbb{X}_f \subseteq \mathbb{X}$ defines the terminal constraint on the state. If we consider a regulation problem, the system (3.1) should be steered to the origin and the cost function $J(\boldsymbol{x}(k), \mathrm{u})$ could be in a quadratic form as follows:

$$J(\boldsymbol{x}(k), \mathrm{u}) = \boldsymbol{x}_{k+N}^\intercal \boldsymbol{P}_f \boldsymbol{x}_{k+N} + \sum_{i=1}^{N} \left(\boldsymbol{x}_{k+i}^\intercal \boldsymbol{Q} \boldsymbol{x}_{k+i} + \boldsymbol{u}_{k+i}^\intercal \boldsymbol{R} \boldsymbol{u}_{k+i} \right) \tag{3.3}$$

where $\boldsymbol{P}_f$, $\boldsymbol{Q} \geq 0$ (positive semi-definite) and $\boldsymbol{R} > 0$ (positive definite) are weighting matrices.

Instead if we consider a servo problem, like tracking of a reference signal, the cost function is changed as follows:

$$\begin{aligned} J(\boldsymbol{x}(k), \mathrm{u}) =& (\boldsymbol{x}_{k+N} - \boldsymbol{x}_{k+N}^{\text{ref}})^\intercal \boldsymbol{P}_f (\boldsymbol{x}_{k+N} - \boldsymbol{x}_{k+N}^{\text{ref}}) \\ &+ \sum_{i=1}^{N} \left((\boldsymbol{x}_{k+i} - \boldsymbol{x}_{k+i}^{\text{ref}})^\intercal \boldsymbol{Q} (\boldsymbol{x}_{k+i} - \boldsymbol{x}_{k+i}^{\text{ref}}) + \boldsymbol{u}_{k+i}^\intercal \boldsymbol{R} \boldsymbol{u}_{k+i} \right) \end{aligned} \tag{3.4}$$

where $\boldsymbol{x}_{k+i}^{\text{ref}}$, $\boldsymbol{x}_{k+N}^{\text{ref}}$ describe the reference trajectory. The standard MPC algorithm can be summarized by the following steps:

Algorithm 1 Basic Model Predictive Control loop

1: Measure the current state $\boldsymbol{x}(k)$;
2: Solve the optimization problem (3.2) with $\boldsymbol{x}(k)$ as initial state, where $\boldsymbol{u}(k)$ is calculated;
3: Apply the first control of the optimal control sequence;
4: Wait one sampling time and repeat steps 1-3;

An MPC has many strengths. Given that the model is discrete and linear it handles multivariable problems very well. Also mathematical convexity is an important part of the resulting problem formulation of an MPC. In fact there exists efficient solvers for convex optimization problems but it is therefore desirable that the MPC problem (3.2) is convex which is ensured if:

1. the cost function is convex;
2. the prediction model is linear;
3. the constraint sets $\mathbb{X}$, $\mathbb{U}$ are convex.

The optimization handles actuator constraints and state constraints naturally in the optimization which allows for the process to be operated much closer to the hard constraints, which improves

control performance and efficiency. Because of its predictive nature it is able to solve a variety of problems and handle disturbances smoothly.

3.1.1 Tuning Parameters

The two most important parameters to tune in order to satisfy the control objectives are the diagonal matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ that can be used to weight the system state matrix and the control inputs respectively. The response of the system that is too slow can be influenced by adding high weighting values in the $\boldsymbol{Q}$ matrix, whereas the control gains are damped with high weighing values in the $\boldsymbol{R}$ matrix. Find an optimal trade-off is a fundamental aspect for the constroller behaviour.

3.1.2 Stability of MPC controller

A limited horizon on the MPC problem affects the stability of the controllers; in order to avoid this problem it is possible to set an infinite horizon, impose end point constraints, terminal cost function or use other techniques. To obtain a stable controller, the parameters to tune are: the terminal cost, prediction horizon and constraints. Also the weights on the cost function can be tuned to ensure a stabilizing solution.

3.1.3 Robustness

If the stability can be guaranteed and the performance specifications are met with respect to a certain set of uncertainties, the system is said to be robust; in particular a controller with this property

has to ensure that the constraints are never violated for any admissible disturbance realization. The uncertainties in a system are due to external disturbances, measurement noise, inaccurate values of the model parameters, non-linearities etc... The most common type of uncertainties considered in the literature is additive disturbance because usually the current state of the system can be measured hence there is no noise in the measurements.

3.2 Adaptive Model Predictive Control

We understood that Model Predictive Control is an advanced method that predicts future behavior using a linear-time-invariant (LTI) dynamic model. These predictions are not exact and a good strategy is to make MPC insensitive to prediction errors. If the plant is strongly nonlinear or its characteristics vary dramatically with time, MPC performance might become unacceptable because LTI prediction accuracy degrade [33]. A method that can address this degradation by adapting the prediction model for changing operating conditions is called Adaptive MPC: this control strategy uses a fixed model structure, but allows the model parameters to evolve with time. Ideally, whenever the controller requires a prediction, it uses a model appropriate for the current conditions. At each control interval, the adaptive MPC controller updates the plant model and nominal conditions. Once updated, the model and conditions remain constant over the prediction horizon. The plant model used as the basis for the adaptive MPC must be an LTI discrete-time, state-space model with a struc-

ture as follows:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}_u\boldsymbol{u}(k) + \boldsymbol{B}_v\boldsymbol{v}(k) + \boldsymbol{B}_d\boldsymbol{d}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}_v\boldsymbol{v}(k) + \boldsymbol{D}_d\boldsymbol{d}(k) \end{aligned} \tag{3.5}$$

where the matrices $\boldsymbol{A}$, $\boldsymbol{B}_u$, $\boldsymbol{B}_v$, $\boldsymbol{B}_d$, $\boldsymbol{C}$, $\boldsymbol{D}_v$ and $\boldsymbol{D}_d$ can vary with time. The other parameters in the previous expression (3.5) are:

- k is the time index/current control interval;
- $\boldsymbol{x}$ are the plant model states;
- $\boldsymbol{u}$ are the manipulated inputs that can be adjusted by the MPC controller;
- $\boldsymbol{v}$ are the measured disturbance inputs;
- $\boldsymbol{d}$ are the unmeasured disturbance inputs;
- $\boldsymbol{y}$ are the plant outputs, including both measured (necessary at least one) and unmeasured.

In the adaptive MPC control, there are additional requirements for the plant model, like the sample time T_s that has to be constant and identical to the MPC control interval. This control strategy prohibits direct feed-through from any manipulated variable to any plant output. Thus, $\boldsymbol{D_v} = \boldsymbol{0}$ in the above model. A traditional MPC controller includes a nominal operating point at which the plant model applies, such as the condition at which you linearize a nonlinear model to obtain the LTI approximation (equilibrium, reference trajectory and the most updated value) [33]. In adaptive MPC, as time evolves it should update the nominal operating point to be consistent with the updated plant model. It is possible to rewrite the plant model in terms of deviations from the nominal

conditions as follows:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \overline{\boldsymbol{x}} + \boldsymbol{A}(\boldsymbol{x}(k) - \overline{\boldsymbol{x}}) + \boldsymbol{B}(\boldsymbol{u}_t(k) - \overline{\boldsymbol{u}}_t) + \overline{\Delta \boldsymbol{x}} \\ \boldsymbol{y}(k) &= \overline{\boldsymbol{y}} + \boldsymbol{C}(\boldsymbol{x}(k) - \overline{\boldsymbol{x}}) + \boldsymbol{D}(\boldsymbol{u}_t(k) - \overline{\boldsymbol{u}}_t) \end{aligned} \tag{3.6}$$

where the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are updated with respect to time. The other parameters in the previous structure (3.6) are:

- $\boldsymbol{u}_t$ is the combined plant input variable, comprising $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{d}$ variables defined earlier;
- $\overline{\boldsymbol{x}}$ are the nominal states;
- $\overline{\Delta \boldsymbol{x}}$ are the nominal state increments;
- $\overline{\boldsymbol{u}_t}$ and $\overline{\boldsymbol{y}}$ are the nominal inputs and outputs.

The adaptive MPC uses a Kalman filter to update its controller states which include the plant, the disturbance and measurement noise model states. In particular this filter is linear-time-varying (LTV) because adjusts the gains at each control interval to maintain consistency with the updated plant model.

4

Moving Obstacle Avoidance

In this chapter we present an Obstacle Avoidance alghorithm based on Adaptive Model Predictive Control. First we introduced the case-study model, then we designed the controller and the method of decision-making. Finally we verified the proposed strategy in different scenarios.

4.1 Problem Formulation

The collision avoidance problem is very dependent on the vehicle modeling since it is a requirement for adaptive MPC law design. Figure 4.1 illustrates a typical scenario of overtaking of a moving obstacle. The maneuver from the driver is to temporarily move to another lane, drive past the obstacle, and move back to the original lane afterward.

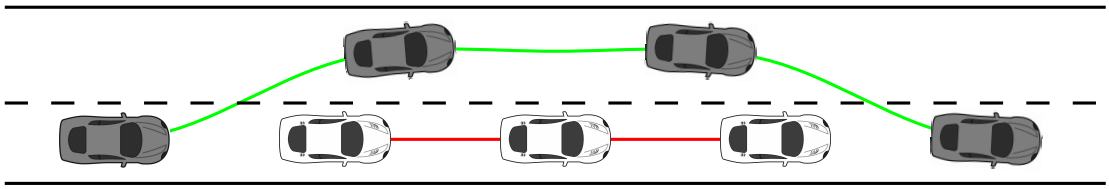


Figure 4.1: Problem description of collision avoidance on a road with only two lanes.

This is possible if and only if there is a free lane or sufficient space for overtaking the obstacle. In the case the road is busy with vehicles, the ATLASCAR2 must slow down and adapts its speed to that of the closest obstacle until the scenario changes.

4.1.1 Car-like Model

The model used in this part of the thesis should take into account the kinematic and dynamic aspects of the vehicle. Here, we present a non linear mathematical model of a vehicle used for the de-

velopment of a collision avoidance system. The model has four states and two inputs:

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix}, \qquad \boldsymbol{u} = \begin{bmatrix} T \\ \delta \end{bmatrix} \tag{4.1}$$

where (x, y) are the global coordinates of the contact point between the rear wheel and the ground, θ is the heading angle of the car body with respect to the x-axis and v is the linear speed of the car (positive). The manipulated variables are T the throttle (positive when accelerating/negative when braking) and δ the steering angle of the front wheel with respect to the vehicle (0 when aligned with car, counterclockwise positive). Figure 4.2 illustrates the applied nonlinear bicycle model and the related parameters.

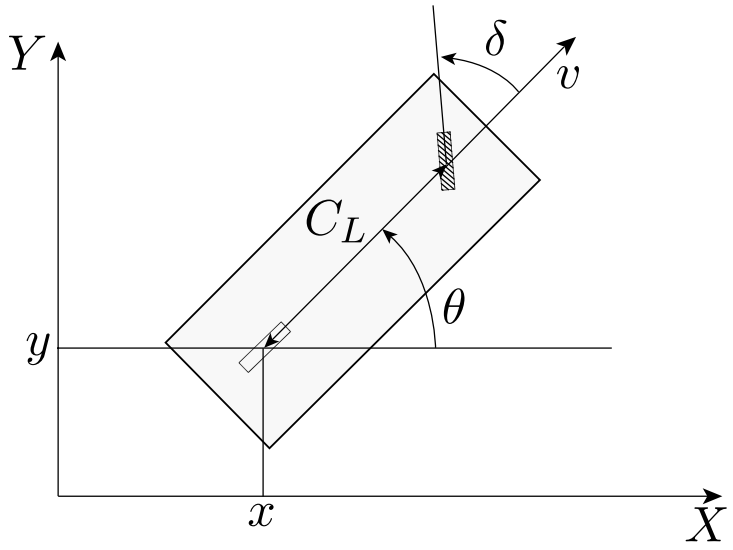


Figure 4.2: Bicycle model of a car (adapted from [34]).

The ATLASCAR2 can be modeled using the non-linear kinematic bicycle model described by the following equations of motion [35] [36]:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \dfrac{v}{C_L}\tan(\delta) \\ \dot{v} = 0.5 \cdot T \end{cases} \implies \begin{array}{l} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \end{array} \tag{4.2}$$

where C_L is the car length. When the velocity is zero then the rate of change of heading angle is zero, that is, it is not possible to change the vehicle's orientation when it is not moving. If the front wheel is orthogonal to the back wheel, i.e. the steering angle is $\frac{\pi}{2}$, the ATLASCAR2 cannot move forward and the model enters an undefined region. In order to simplify the model, it is assumed that only the front wheel can be steered. Moreover, in this chapter it is assumed that the ATLASCAR2 does not slip, so any slippage is thus considered as an external disturbance. Under this assumption, the slip angle is zero, meaning that the velocity is directed along the heading of the vehicle. In particular the rate of change of heading angle is referred to as turn rate which can be evaluated with a gyroscope. This yaw rate can also be deduced from the angular velocity of the wheels on the left- and right-hand sides of the vehicle which therefore rotate at different speeds because follow arcs of different radius. We can write the non-holonomic constraint which is an expression for velocity in the vehicle's y-direction in the world coordinate frame as follows:

$$\dot{y}\cos\theta - \dot{x}\cos\theta = 0. \tag{4.3}$$

This equation cannot be integrated to form a relationship between x, y and θ. In order to use an MPC controller, the state space model needs to be linearized with a first order approximation and also re-written in a more compact form:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}_c \boldsymbol{x} + \boldsymbol{B}_c \boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}_c \boldsymbol{x} + \boldsymbol{D}_c \boldsymbol{u} \end{aligned} \tag{4.4}$$

where the matrices $\boldsymbol{A}_c$, $\boldsymbol{B}_c$, $\boldsymbol{C}_c$ and $\boldsymbol{D}_c$ are obtained as follows:

$$\boldsymbol{A}_c = \frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}} = \begin{bmatrix} 0 & 0 & -v\sin(\theta) & \cos(\theta) \\ 0 & 0 & v\cos(\theta) & \sin(\theta) \\ 0 & 0 & 0 & \tan(\delta)/C_L \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{B}_c = \frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \frac{v}{C_L}\big(\tan(\delta)^2 + 1\big) \\ 0.5 & 0 \end{bmatrix}, \tag{4.5}$$

$$\boldsymbol{C}_c = \frac{\partial \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}} = \mathbf{I}_4, \qquad \boldsymbol{D}_c = \frac{\partial \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}} = \mathbf{0}_{4\times 2}.$$

The simple linearized approximation of the system to describe the dynamics of the ATLASCAR2

will be evaluated at the operating conditions. Note also that the system we are considering is a linear state-space model whose dynamics vary as a function of certain time-varying parameters. The system to be controlled is usually modeled by a discrete state-space model in the MPC literature. Therefore, (4.4) is transformed into a discrete state-space model to be used by the Model Predictive Controller:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}_c\boldsymbol{x} + \boldsymbol{B}_c\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}_c\boldsymbol{x} + \boldsymbol{D}_c\boldsymbol{u} \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}_d\boldsymbol{x}(k) + \boldsymbol{B}_d\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}_d\boldsymbol{x}(k) + \boldsymbol{D}_d\boldsymbol{u}(k) \end{aligned} \tag{4.6}$$

where $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ are the state and control matrices for the discrete state-space equation, respectively, which can be calculated with the Euler method as

$$\boldsymbol{A}_d = e^{\boldsymbol{A}_c T_s}, \qquad \boldsymbol{B}_d = \int_{kT_s}^{(k+1)T_s} e^{\boldsymbol{A}_c[(k+1)T_s - \eta]}\boldsymbol{B}_c d\eta \tag{4.7}$$

where T_s is the sampling interval for the discrete state-space model. The matrices $\boldsymbol{C}_d$ and $\boldsymbol{D}_d$ are equivalent to those in the continuous case. For simplicity, we assume that all the states are measurable and the ATLASCAR2 drives east with a constant speed at the nominal operating point.

In the scenario that we are going to consider, the road is straight and our vehicle stays in the middle of the center lane when not passing. Without losing generality, the ATLASCAR2 passes an obstacle both to the right and to the left lane depending on where it is placed on the road. We create also a safe zone around the obstacles so that the vehicle does not get too close to the obstacle when passing it.

4.2 Design of Adaptive Model Predictive Control

We designed a Model Predictive Controller that can make the ATLASCAR2 maintain a desired velocity and stay in the middle of center lane. We used an Adaptive MPC controller because it handles the nonlinear vehicle dynamics more effectively than a traditional MPC controller; in fact, the latter uses a constant plant model but the former allows us to provide a new plant model at each control interval. Because the new model describes the plant dynamics more accurately at the new operating condition, an adaptive MPC controller performs better than a traditional MPC controller.

In practice, at each control interval, the adaptive MPC controller updates the plant model and the nominal conditions. Once updated, the model and the conditions remain constant over the prediction horizon. In motion planning that uses adaptive MPC, it is common to formulate the constrained control problem as a real-time optimization problem subject to hard constraints on plant variables and soft constraints on outputs; at the beginning, we specified the constraints for the manipulated variables: to prevent the ATLASCAR2 from accelerating or decelerating too quickly, we added an hard constraint on the throttle rate of change and another one on the steering angle rate of change. We used an approach that takes advantage of the ability of MPC to handle constraint explicitly. Figure 4.3 shows a conditional state machine designed for higher-level behavior planning.

When an obstacle is detected, it defines an area on the road (in terms of constraints) that the ATLASCAR2 must not enter during the prediction horizon. At the next control interval the area is redefined based on the new positions of the vehicle and the obstacle until passing is completed.

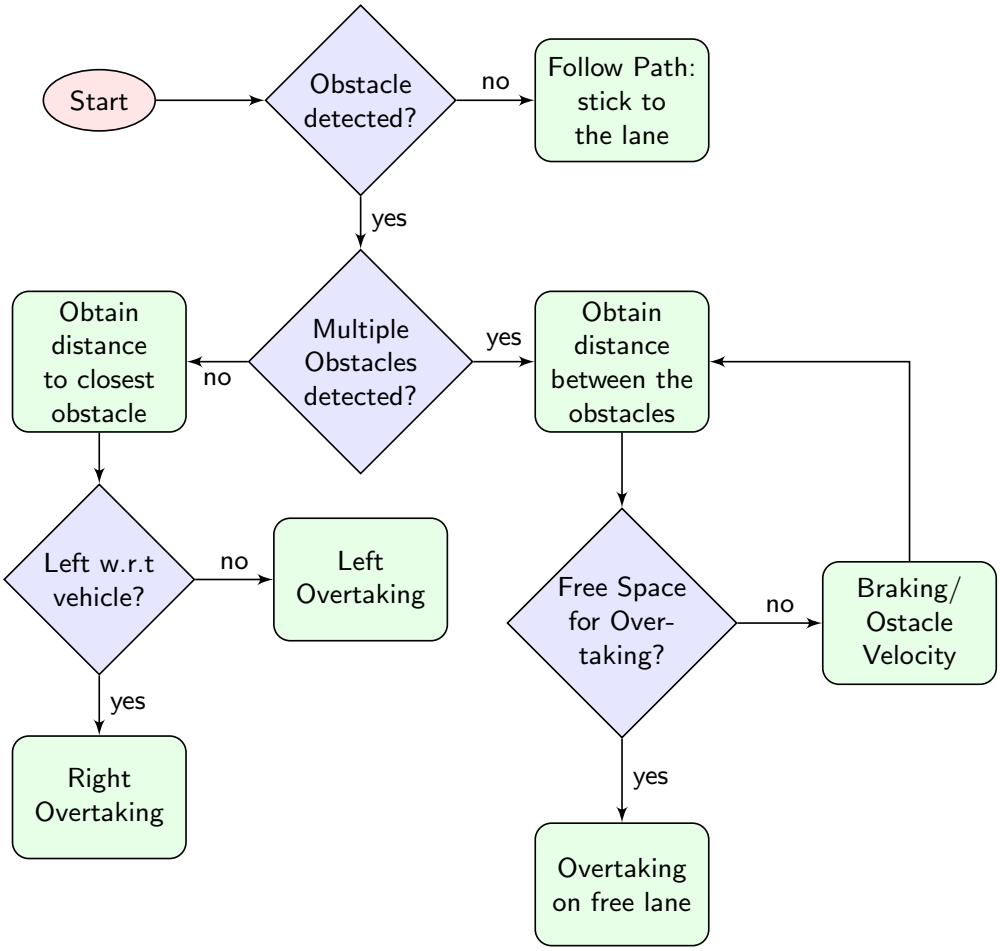


Figure 4.3: Behaviour planning conditional flowchart.

To define the area to avoid, we used the following mixed Input/Output constraints:

$$\boldsymbol{E}\boldsymbol{u} + \boldsymbol{F}\boldsymbol{y} \leq \boldsymbol{G} \tag{4.8}$$

where $\boldsymbol{u}$ and $\boldsymbol{y}$ are respectively the manipulated variable vector and the output variable vector, while $\boldsymbol{E}$, $\boldsymbol{F}$, $\boldsymbol{G}$ are the constraint matrices that can be updated when the controller is running.

Five constraints were defined:

1. upper bound on the y-coordinate (left boundary of the road);
2. lower bound on the y-coordinate (right boundary of the road);
3. constraint for obstacle avoidance; although no obstacle is considered in the nominal condition, we must add this virtual constraint here because we cannot change the dimensions of the constraint matrices at run time;
4. upper bound on the x-coordinate (position of the closest obstacle);
5. lower bound on the x-coordinate (position of the ATLASCAR2).

The matrices for the above inequality are the following:

$$\boldsymbol{E} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad \boldsymbol{F} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ cS & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{G} = \begin{bmatrix} W/2 \\ W/2 \\ -cI \\ x_{\max} \\ x_{\min} \end{bmatrix} \tag{4.9}$$

where W is the width of the road, cI and cS are the required parameters such that the ATLASCAR2 must be above the line formed from the vehicle to safe zone corner for left/right passing and finally $x_{\max}$ represents the position of the closest obstacle in the case there is not free space for the overtaking (otherwise $x_{\max} = +\infty$) while $x_{\min}$ depicts the location of our vehicle. Figure 4.4 illustrates the constraints that are computed at each T_s in the case of a left overtaking.

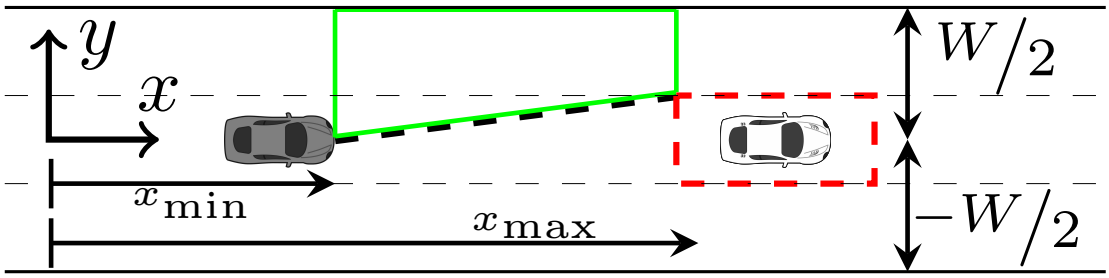


Figure 4.4: Constraints in the case of left overtaking.

4.3 Simulation Results

The performances of the proposed adaptive MPC based vehicle control method are demonstrated in four simulation examples. We tried to choose parameters that were as close as possible to a real situation: the sampling time used in the discretization of the system is $T_s = 0.02\,\mathrm{s}$ while the values of the prediction and the control horizon are respectively $p_H = 25$ and $c_H = 5$.

In all simulations, the distance between the front and rear axles is $C_L = 5\,\mathrm{m}$ and the width of the vehicle is $C_W = 2\,\mathrm{m}$. The saturation ranges of the control inputs are: the steering angle rate of change lies in $\left[-\frac{\pi}{30}, +\frac{\pi}{30}\right]$ rad/s while in order to prevent the ATLASCAR2 from accelerating or decelerating too quickly, we impose an hard constraint of $2.5\,\mathrm{m/s^2}$ on the throttle rate of change.

Moreover we are using a constant reference signal for the velocity of $v = 20\,\mathrm{m/s}\,(\approx 72\,\mathrm{km/h})$. Note that the blue paths in Figures 4.6, 4.8, 4.9, 4.11 and 4.13 are known only at the end of the simulations but we have decided to report the entire trajectory that the vehicle will perform in every frame. The overall framework for a multiple moving obstacle avoidance is depicted in the Figure 4.5.

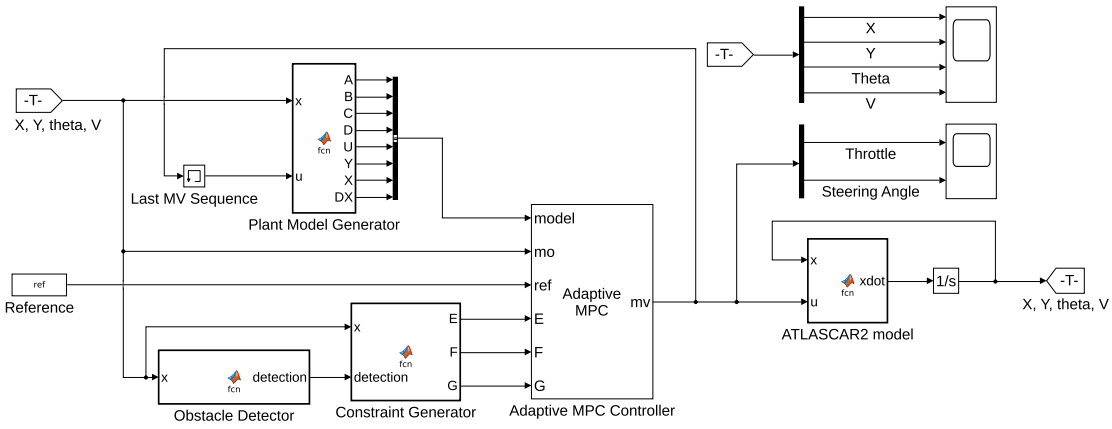


Figure 4.5: Overall procedure scheme moving obstacle avoidance.

All the simulations are done in MATLAB and Simulink. MATLAB is used for loop-shaping. Simulink is used as a testing environment. Moreover, the MATLAB Model Predictive Toolbox is used. The controller will be tested in the same Simulink environment as the classical controller. Note that all these blocks have been implemented through the S-functions (types of dynamically linked subroutines for Simulink).

4.3.1 One Moving Obstacle - Right Overtaking

In this first simulation (Figure 4.6) the ATLASCAR2 drives in the middle of the center lane while the road is completely free and when there is an obstacle, the vehicle passes it only using the right lane (the same simulation can be launched so that the car goes over to the left fast lane). In this example the obstacle has a constant speed of $8\,\mathrm{m/s}$ and moves in the same direction as the ATLASCAR2. We assume that the LIDAR sensor can detect an obstacle $30\,\mathrm{m}$ in front of the vehicle. The red dashed block around the obstacle represents a safe zone used to evaluate the restrictions so that,

even if there is a small margin of error in the manuever, there is always a safe distance between the ATLASCAR2 and the nearby vehicle.

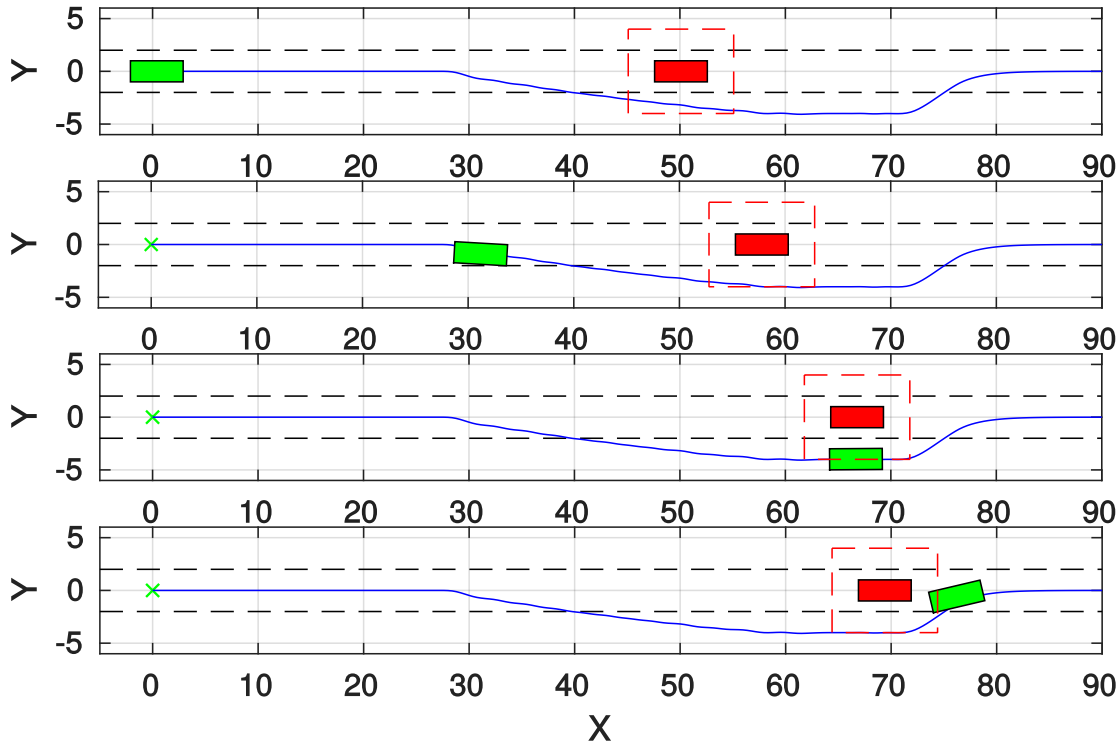


Figure 4.6: Simulation of right overtaking with one moving obstacle that moves in the same direction as the vehicle.

Algorithm 2 summarizes the main steps to compute custom constraints for the obstacle; when the vehicle detects the obstacle, the constraints are computed.

In other words the constraints are evaluated as follows:

- if the ATLASCAR2 is already in the adjacent lane, it uses the safety zone as the constraint (line 7);
- otherwise the vehicle must be above the line formed from the ATLASCAR2 to safe zone corner for right passing (lines 9 and 10);
- if the vehicle is parallel to the obstacle (line 14), it uses the safety zone as the constraint;

- finally if it has passed the obstacle, it uses the inactive constraint to go back to the center lane (line 16).

Algorithm 2 Right Overtaking if an obstacle is detected

```
1:  function RightOvertaking(car, obstacle, road)
2:      x_min ← carX, x_max ← +∞;
3:      obsYrr = Obstacle.RearRightSafeY;
4:      obsXrr = Obstacle.RearRightSafeX;
5:      if ATLASCAR2 is behind the obstacle then
6:          if ATLASCAR2 is in the adjacent lane then
7:              cS ← 0; cI ← obsYrr;
8:          else
9:              cS ← tan(atan2((obsYrr − carY)/(obsXrr − carX), 1));
10:             cI ← obsYrr − cS ∗ obsXrr;
11:         end if
12:     else
13:         if ATLASCAR2 is parallel to the obstacle then
14:             cS ← 0; cI ← obsYrr;
15:         else
16:             cS ← 0; cI ← W/2;
17:         end if
18:     end if
19:     return x_min, x_max, cI, cS
20:     Update matrices E, F and G
21: end function
```

In this simulation only the first three constraints are necessary because there is enough space for the overtaking without braking (the fourth and fifth constraints don't change). The first two constraints are constant and represent the left and right boundary of the road (upper and lower bound on the y-coordinate). The third constraint is useful to define a region where the vehicle can navigate; it consists of two parameters (cS and cI) that are calculated in the algorithm depending on where the vehicle is located with respect to the obstacle. After calculating these parameters, the matrices $\boldsymbol{E}$, $\boldsymbol{F}$ and $\boldsymbol{G}$ are updated so that the MPC can process the new optimal input.

Finally we can analyze time signals of the ATLASCAR2 in the simulation of right overtaking of one moving obstacle. Figures 4.7a and 4.7b illustrate the longitudinal and lateral position of the vehicle with respect to time, showing a constant trend with respect to the x component, while a shift from 0 can be noted regarding the y coordinate due to the overcoming of the obstacle. To confirm a uniform trend, the speed of the ATLASCAR2 remains fairly constant. Obviously it is possible to note from Figure 4.7c that the velocity decreases very little during the first part of the overtaking phase, and then come back to the cruising speed in the second part. Instead from the remaining Figures 4.7d, 4.7e and 4.7f we can observe that the throttle T and the steering angle δ respect the limits we imposed while the heading angle θ is used to describe the direction the ATLASCAR2 is pointing.

4.3.2 Multiple Moving Obstacles

For a second test, additional obstacles were added to make the scenario more complex like in a real highway.

Initially we assumed that the obstacles were located at a distance greater than the LIDAR detection range which is $30\,\mathrm{m}$. In this way every time the vehicle overtakes a car, it applies the inactive constraints to go back to the center lane.

This situation is represented in Figure 4.8 where there are three different scenarios with $N = 2, 3$ and 4 moving obstacles that drive with a constant velocity of $8\,\mathrm{m/s}$ in the opposite direction with respect to the ATLASCAR2. The vehicle is capable of overtaking the obstacles on the right or

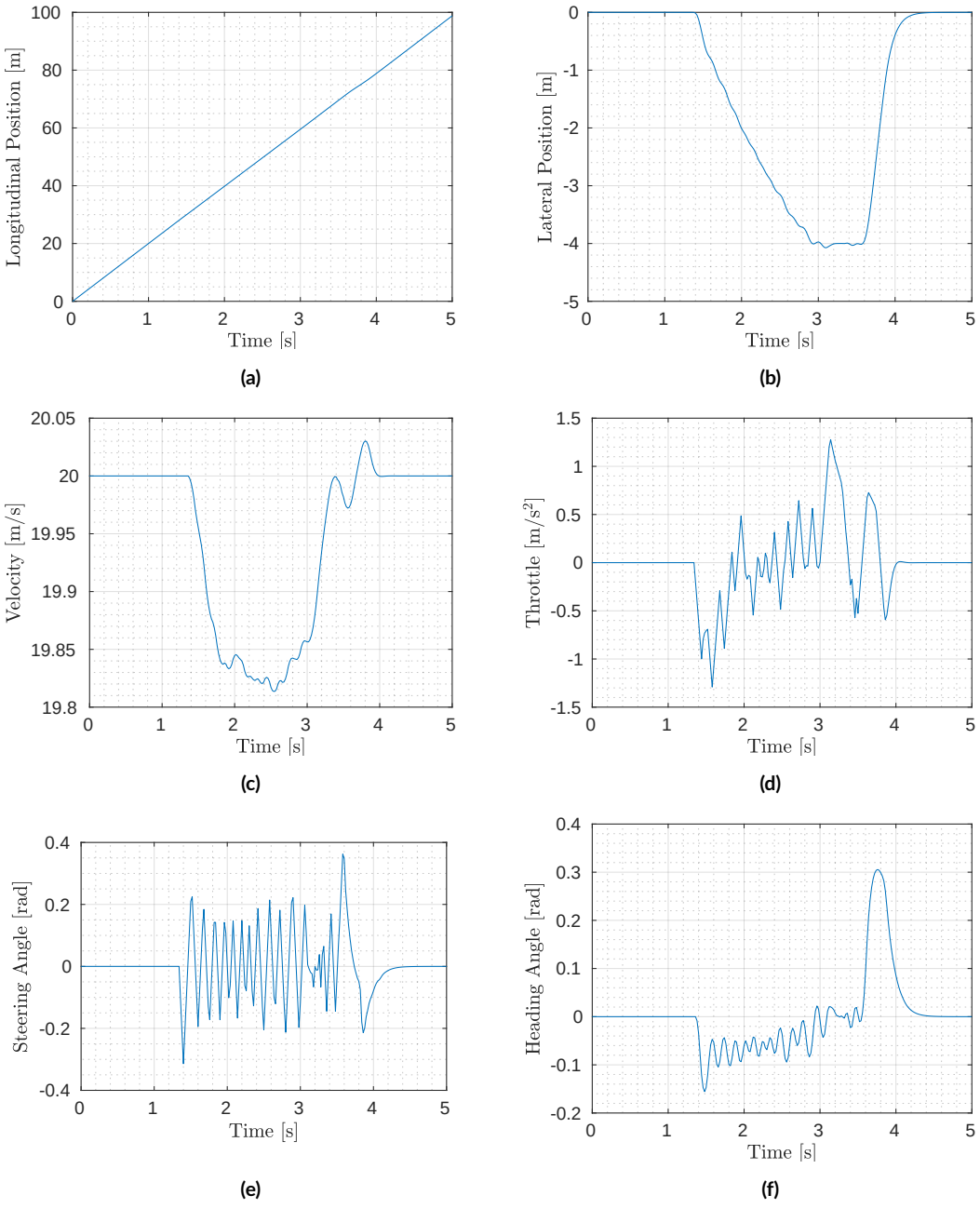


Figure 4.7: Time signals of the ATLASCAR2 in the simulation of right overtaking of one moving obstacle illustrated in Figure 4.6.

left depending on their positions with respect to the road. If the y-coordinate of the closest obstacle is greater than 0, then the vehicle overtakes to the right, otherwise the overtaking takes place on the left lane.

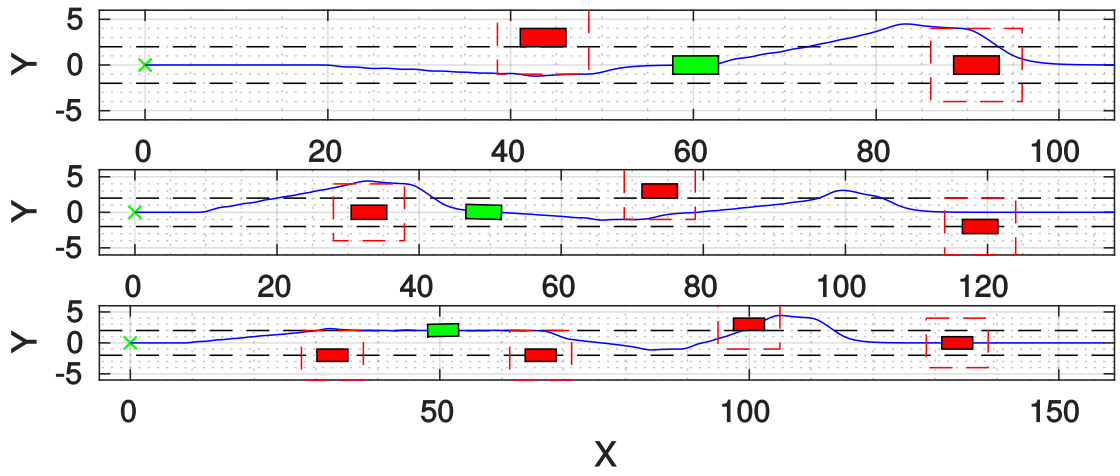


Figure 4.8: Simulations of overtaking with $N = 2, 3, 4$ moving obstacles that drive in the opposite direction with respect to the ATLASCAR2.

In the Figure 4.9 we also hypothesized there are six moving obstacles that can drive at different speeds but initially they are at a common distance; they can have a uniform motion or a uniformely accelerated motion. We also improved the code infact in case two obstacles are too close during the simulation and their distance is less than the detection range, which is 30 m, the ATLASCAR2 perceives the objects as a single entity and adapts to the situation. The same test can be done with the obstacles that drive in the same direction as the vehicle but to better assess and demonstrate results we decided to show a very unusual scenario.

Also in this case we reported time signals of the ATLASCAR2 in the simulation with six moving obstacles (Figure 4.10). The most articulated scenario makes the signals more complex with respect

to time. However, we can see that the speed is just under 20 $\mathrm{m/s}$ while the inputs continue to respect the imposed boundaries. Note that the vehicle manages to overcome all the obstacles without encountering any difficulty since there is always an empty lane.

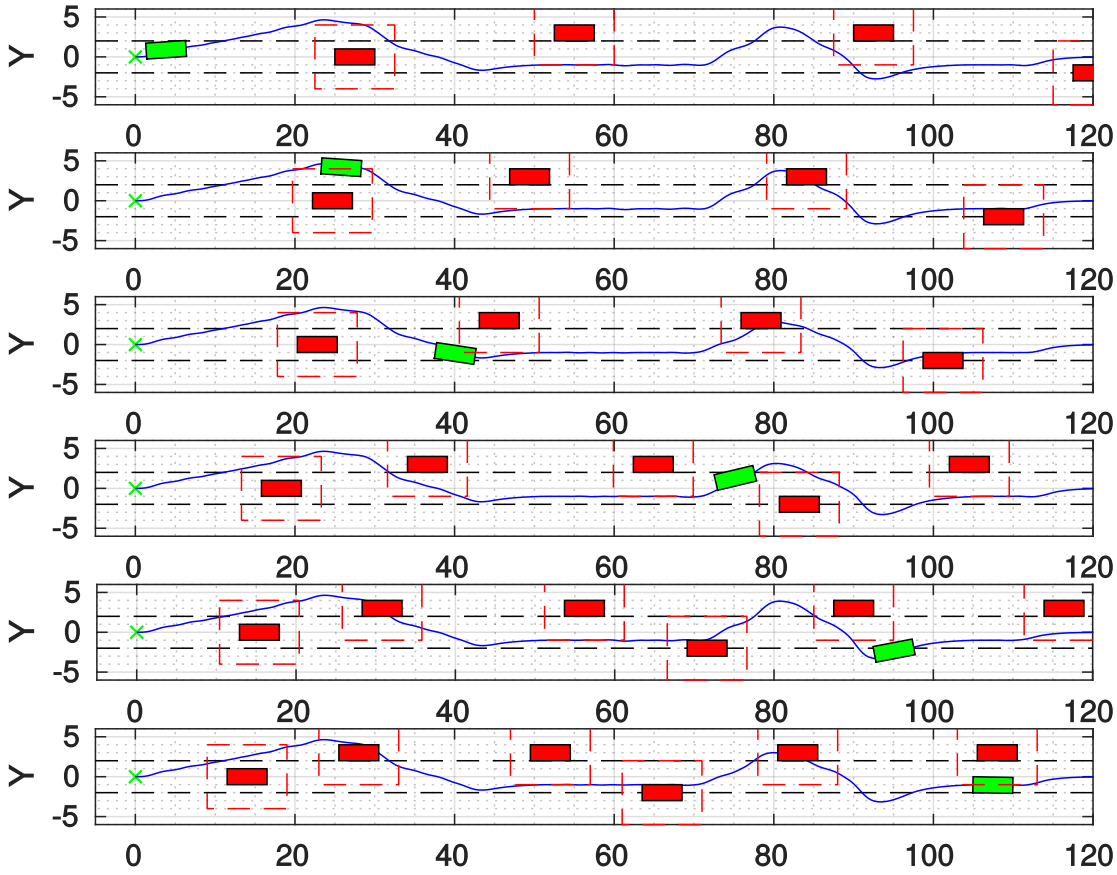


Figure 4.9: Simulation of overtaking with six moving obstacles that moves in the opposite direction with respect the vehicle.

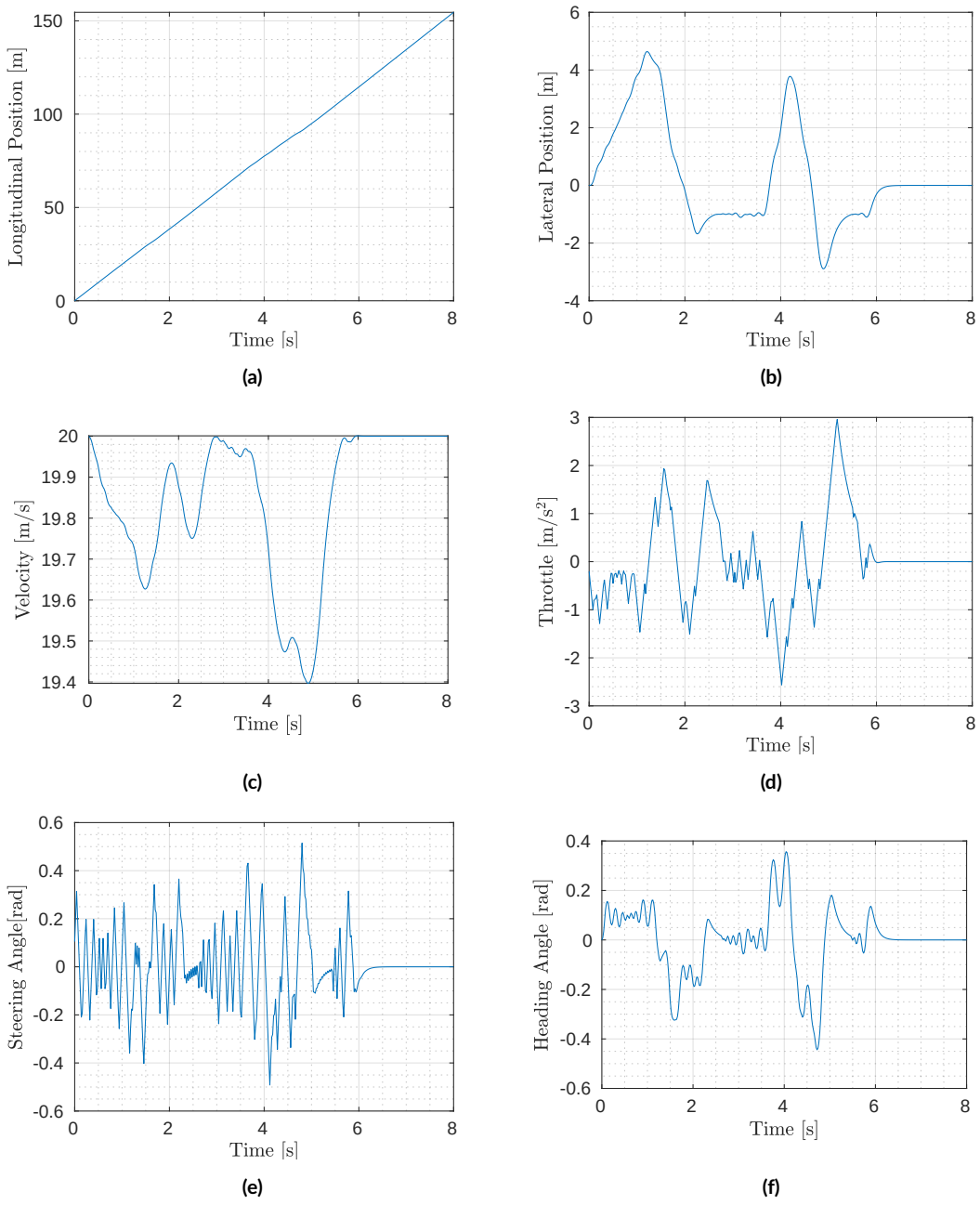


Figure 4.10: Time signals of the ATLASCAR2 in the simulation of overtaking with six moving obstacles illustrated in Figure 4.9.

4.3.3 No Overtaking

We have seen that as long as there is always a free lane, the obstacle avoidance algorithm works correctly. In the case that there are too many obstacles on the road, which do not allow overtaking, the vehicle must adapt to the situation and it must slow down in order not to collide. In the algorithm that we have developed it is possible to calculate two other restrictions which permit this deceleration in the case that it is strictly necessary. To better understand how these restrictions change the speed and deceleration of the ATLASCAR2, we simulated a scenario (Figure 4.11) in which there are 3 obstacles at an initial distance of $35\,\mathrm{m}$ occupying the 3 different lanes, all with the same x-coordinate and the same velocity of $8\,\mathrm{m/s}$. When these obstacles enter in the vehicle's detection range, it checks whether there is enough space to perform the overtaking maneuver by calculating the distance between the obstacles. Noting that it is not possible to overtake, our vehicle must decelerate in order not to crash into the nearest car. To do this, the MPC must take into account the position of the vehicle x_{min} and the position of the closest obstacle x_{max}.

Also in this simulation the reference speed of the car is $20\,\mathrm{m/s}$, but after $\simeq 1.5\,\mathrm{s}$, the obstacles enter in the detection range and as we can see from the Figure 4.12c the velocity decreases rapidly to adapt to that of the nearest vehicle. It is possible to observe that this decrease in speed, which subsequently remains constant at $8\,\mathrm{m/s}$, is due to a strong negative throttle impulse (Figure 4.12d). Finally, we note that the steering angle δ and heading angle θ remain at zero: there is only a very small oscillatory trend due to measurement errors as the vehicle always requires a minimum change of direction. Finally we imposed a safety distance that must always be respected between two vehi-

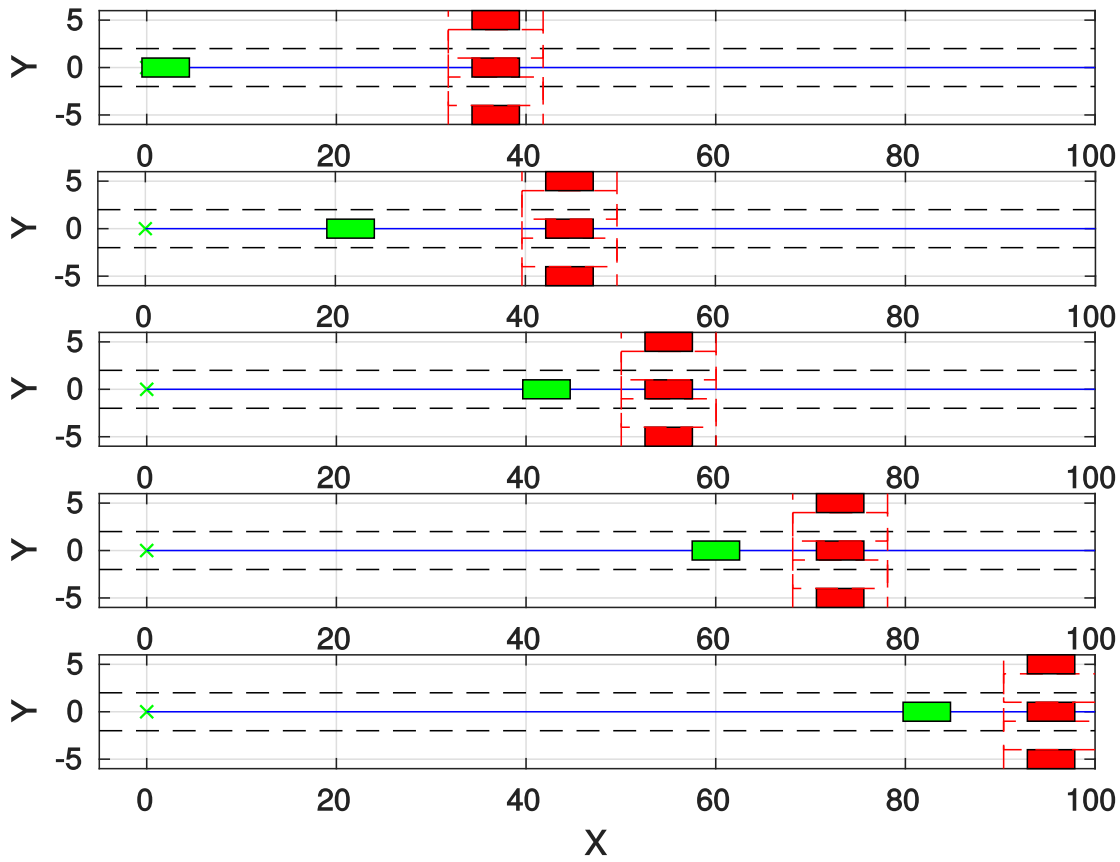


Figure 4.11: Simulation of no overtaking with three moving obstacles that move in the same direction as the vehicle.

cles so that if the obstacle brakes sharply the ATLASCAR2 can avoid the impact. It is very difficult to achieve reliable calculations of the braking distance as road conditions and the tyres' grip can vary greatly. An easy formula to calculate the braking distance is the following one:

$$d = \frac{v^2}{250 \times f} \tag{4.10}$$

where d is the braking distance in metres, v is the speed in km/h, 250 is the fixed figure which is always used and f is the coefficient of friction, approx. 0.8 on dry asphalt and 0.1 on ice.

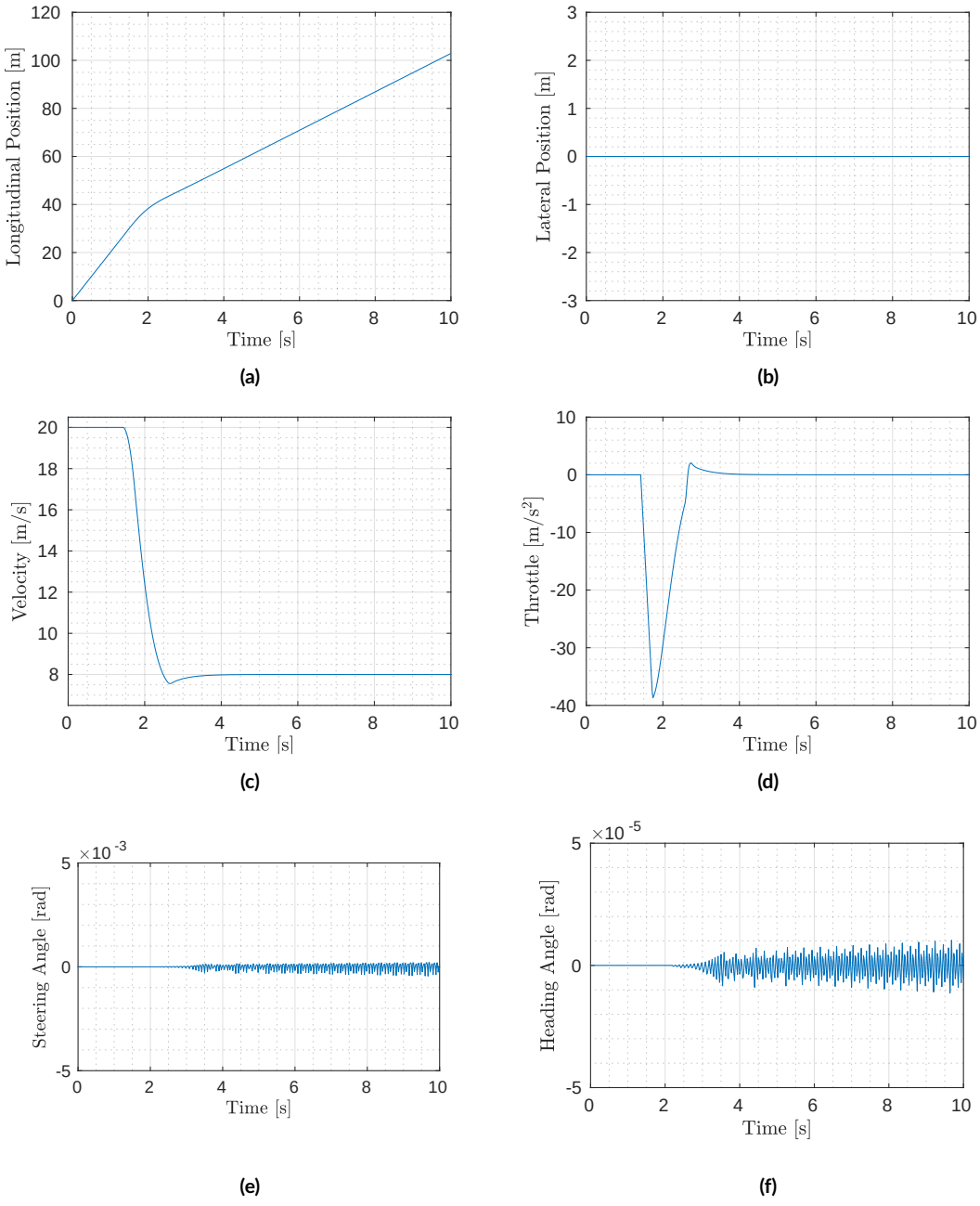


Figure 4.12: Time signals of the ATLASCAR2 in the simulation of no overtaking with three moving obstacles illustrated in Figure 4.11.

4.3.4 Vehicle Braking and Obstacles Overtaking

Finally we have improved the code related to the mixed Input/Output constraints. Figure 4.13 depicts a simulation in which there are 3 obstacles at the same x-coordinate. Two obstacles have a constant speed of $8\,\mathrm{m/s}$ while the one on the left lane of $15\,\mathrm{m/s}$.

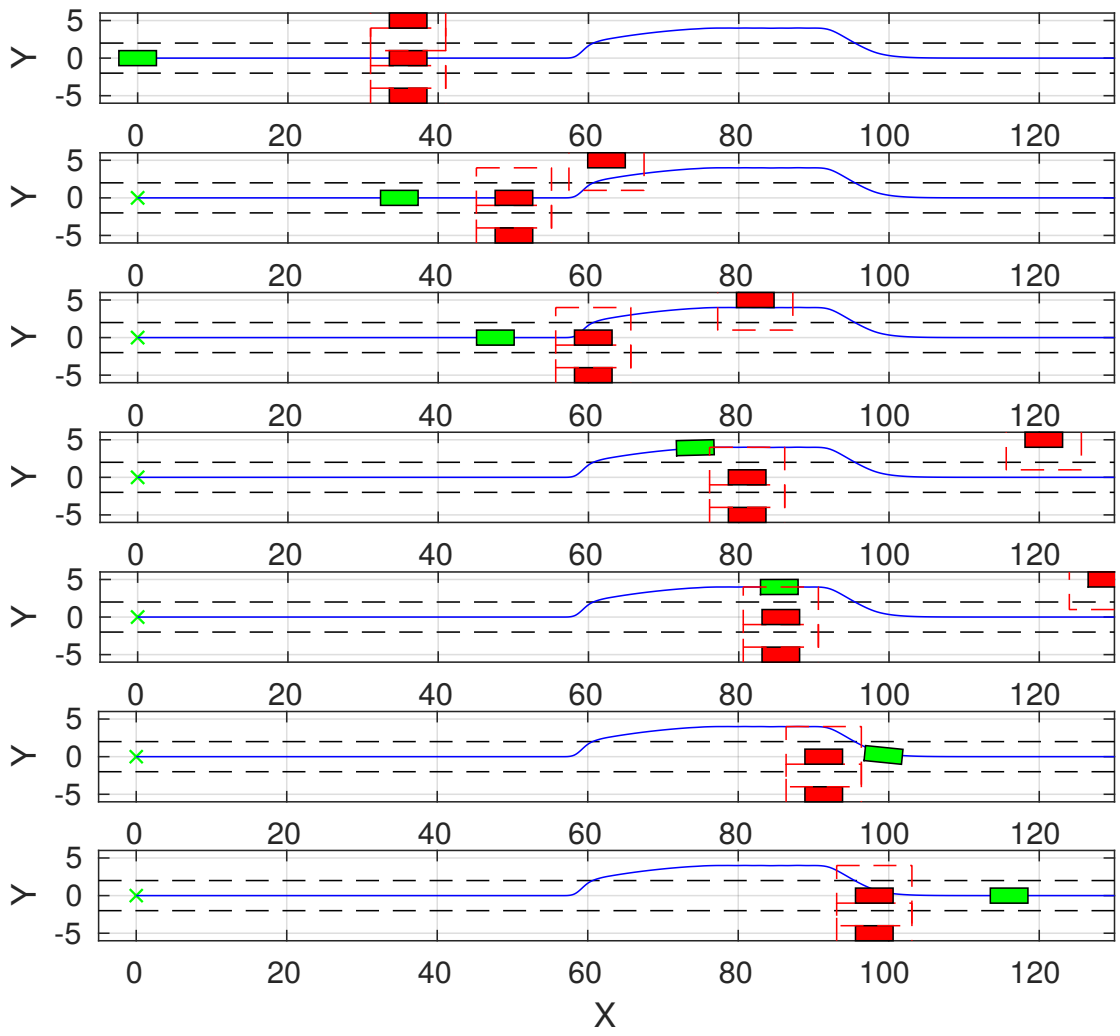


Figure 4.13: Simulation of braking and overtaking obstacles.

In this particular case it is essential to consider the fourth and the fifth restrictions in order to allow the ATLASCAR2 to slow down without colliding with the cars in front. The fifth constraint is simply the position of the ATLASCAR2 that updates at each interval, while the fourth constraint, until there is a free lane, is the position of the closest obstacle (both the positions are with respect to the global reference frame). In the calculation of the fourth constraint is necessary to consider the speed at which the vehicle and the obstacle are moving in order to keep a safe distance. At the beginning, the vehicle moves with the reference velocity of $v = 20\,\mathrm{m/s}$. When the ATLASCAR2 detects all the other cars on the road, checks if there is a free lane. In the first part of the simulation, the ATLASCAR2 brakes because there is not enough space for overtaking the cars as shown in Fig. 4.14c. A collision would happen if the vehicle continues to follow the initially planned path with the reference velocity. It is possible to notice that the speed decreases because the applied throttle is negative, so a consistent deceleration is set after $\approx 1.5\,\mathrm{s}$ as depicted in Fig. 4.14d. The velocity of the ATLASCAR2 for $\approx 2\,\mathrm{s}$ adapts to that of the closest obstacle. After a few seconds the fastest car moves and makes available the left lane for overtaking. Dramatic changes of steering angle in early stage are observed in Fig. 4.14e and consequently also on the heading angle in Fig. 4.14f. Then, the ATLASCAR2 returns to the reference velocity during the overtaking of the two obstacles (the applied throttle after $\approx 5\,\mathrm{s}$ is positive). It is seen that the ATLASCAR2 avoids the obstacles and returns to the road center line with a low overshoot.

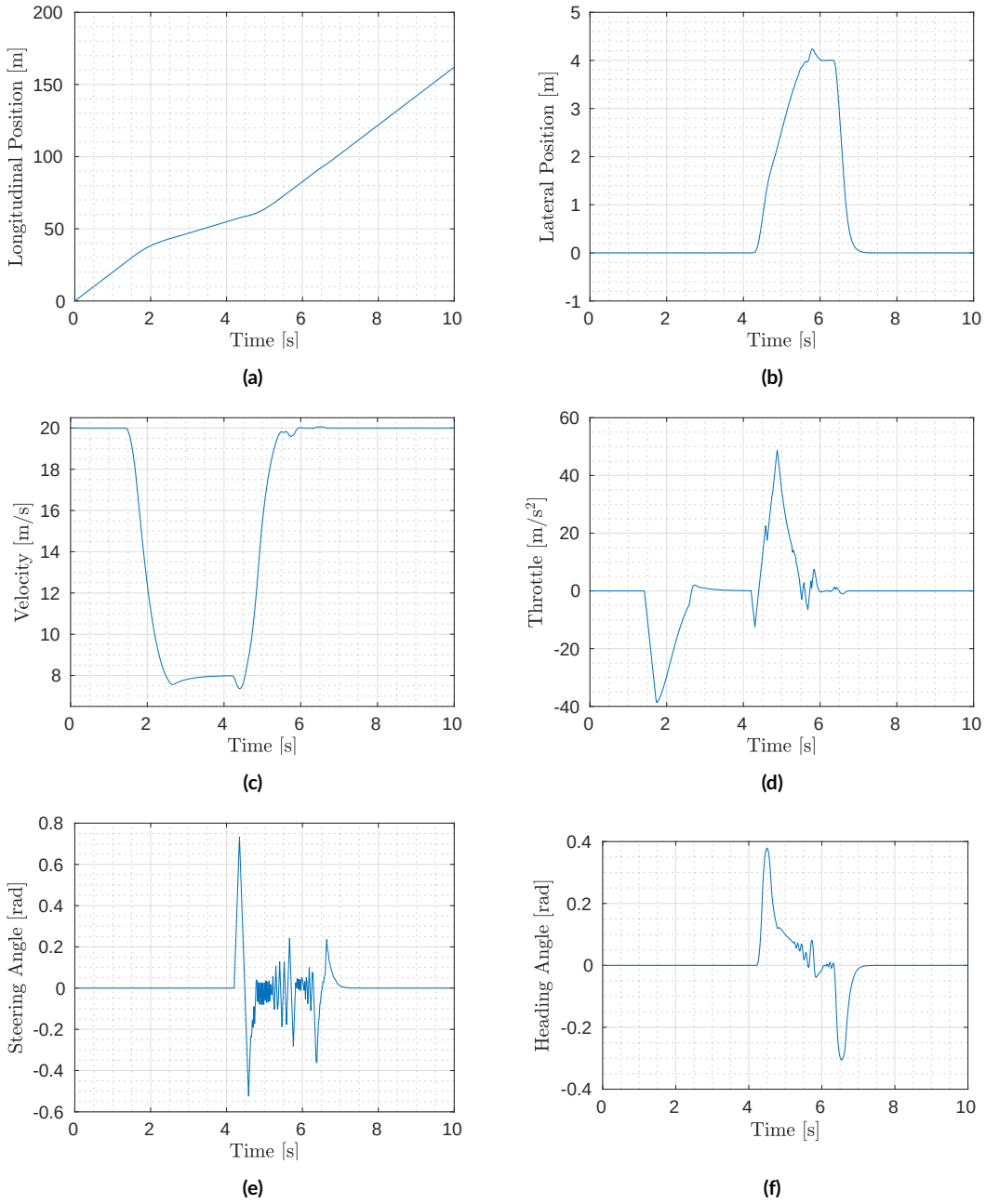


Figure 4.14: Time signals of the ATLASCAR2 in the simulation of braking and overtaking in the situation illustrated in Figure 4.13.

5

Lane Following

In this chapter we present a Lane Following alghorithm based on Adaptive Model Predictive Control. First we introduced the case-study model, then we designed the controller and the overall procedure scheme. Finally we verified the proposed strategy in different scenarios considering a double lane change, sinusoidal and elliptic/circular paths.

5.1 Problem Formulation

A lane-following system is a control system that keeps the vehicle traveling along the centerline of a highway lane, while maintaining a user-set velocity. Figure 5.1 illustrates a typical lane following scenario.

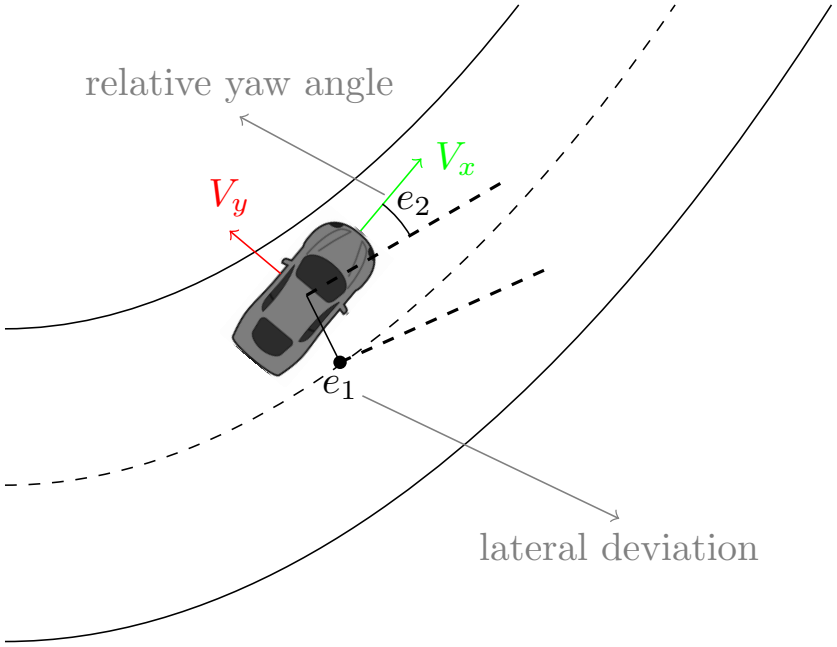


Figure 5.1: Problem description of a lane following system.

In a classic lane keeping assist, it is assumed that the longitudinal velocity is constant [37]. This restriction is relaxed in this model because the longitudinal acceleration varies in this MIMO control system. This lane-following system manipulates both the longitudinal acceleration and the front steering angle of the vehicle to keep the lateral deviation and the relative yaw angle small and the longitudinal velocity close to a driver set velocity. If these two goals cannot be met at the same moment,

the system tries to balance them. The model that we are considering contains many parameters. The first fundamental block describes the vehicle dynamics: we have applied the bicycle model of lateral vehicle dynamics and approximate the longitudinal dynamics using a time constant obtaining a linear model.

5.1.1 Longitudinal Dynamics

Dynamics of the powertrain are commonly modeled using a first-order transfer function, called the generalized vehicle longitudinal dynamic system in [38]:

$$\dot{V}_x = \frac{K}{\tau s + 1} a \tag{5.1}$$

with the system gain usually $K = 1$ and a time constant $\tau = 0.2$. This type of model was used in [38] for a predictive multi-objective vehicular ACC and in [39] for model predictive control of transitional maneuvers for adaptive vehicle cruise control. Moreover a similar modeling approach was applied in [40] to address trajectory tracking for autonomous vehicles, with the goal of developing a racing controller.

We can use the following state space to describe the longitudinal model:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\text{lon}} &= \boldsymbol{A}_m \boldsymbol{x}_{\text{lon}} + \boldsymbol{B}_m \boldsymbol{u}_{\text{lon}} \\ \boldsymbol{y}_{\text{lon}} &= \boldsymbol{C}_m \boldsymbol{x}_{\text{lon}} + \boldsymbol{D}_m \boldsymbol{u}_{\text{lon}} \end{aligned} \tag{5.2}$$

where the input is the desired acceleration and the states are the longitudinal velocity and the actual

acceleration which is also the only output of this system:

$$\boldsymbol{x}_{\text{lon}} = \begin{bmatrix} \dot{V}_x \\ \\ V_x \end{bmatrix}, \qquad \boldsymbol{u}_{\text{lon}} = a, \tag{5.3}$$

while the matrices $\boldsymbol{A}_m$,$\boldsymbol{B}_m$,$\boldsymbol{C}_m$ and $\boldsymbol{D}_m$ are the following:

$$\begin{gathered} \boldsymbol{A}_m = \begin{bmatrix} -\frac{1}{\tau} & 0 \\ \\ 1 & 0 \end{bmatrix}, \qquad \boldsymbol{B}_m = \begin{bmatrix} \frac{1}{\tau} \\ \\ 0 \end{bmatrix}, \\ \\ \boldsymbol{C}_m = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad \boldsymbol{D}_m = 0, \end{gathered} \tag{5.4}$$

where τ is time constant described above.

5.1.2 Lateral Dynamics

The vehicle is modelled at each axis through a bicycle model in which the two wheels are lumped into a single wheel. It will be assumed that a simple linear tire model can be used due to the considered speeds. It is also assumed that the small angles approximation can be used. The equations describing the yaw and lateral motion are:

$$\begin{aligned} m\dot{V}_y &= F_{yf} + F_{yr} \\ I_Z\ddot{\psi} &= l_F F_{yf} - l_R F_{yr} \end{aligned} \tag{5.5}$$

where F_{yf} and F_{yr} are the tire forces at the front and rear axles of the vehicle, which are a function of the cornering stifness of the tires C_F and C_R, the steering angle δ and the vehicle states. The ATLASCAR2 travels with a longitudinal velocity V_x and a lateral velocity V_y. These two velocities can be used to form a vector describing the resultant velocity and its direction. In this case it is also possible to find the slip angles, the difference between the actual travelling direction of the and where it is pointed (in the previous chapter we assumed that the ATLASCAR2 does not slip, so any slippage is thus considered as an external disturbance):

$$\alpha_f = \delta - \theta_{vf}, \qquad \alpha_r = -\theta_{vr} \tag{5.6}$$

where θ_{vf} and θ_{vr} represent the angles of the velocity vectors for the rear and front tires evaluated as follows:

$$\theta_{vf} = \text{atan}\left(\frac{V_y + l_F\dot{\psi}}{V_x}\right), \qquad \theta_{vr} = \text{atan}\left(\frac{V_y - l_R\dot{\psi}}{V_x}\right) \tag{5.7}$$

Finally the lateral tire forces acting on the wheels, F_{yf} and F_{yr}, are proportional to the slip angle, for small angles; so they can be written as:

$$F_{yf} = 2C_F(\delta - \theta_{vf}), \qquad F_{yr} = -2C_R\theta_{vr} \tag{5.8}$$

Combining (5.5) and (5.8) gives the following vehicle lateral model from parameters:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\text{lat}} &= \boldsymbol{A}_g \boldsymbol{x}_{\text{lat}} + \boldsymbol{B}_g \boldsymbol{u}_{\text{lat}} \\ \boldsymbol{y}_{\text{lat}} &= \boldsymbol{C}_g \boldsymbol{x}_{\text{lat}} + \boldsymbol{D}_g \boldsymbol{u}_{\text{lat}} \end{aligned} \tag{5.9}$$

where the input is the steering angle in radians, and the outputs are the lateral velocity in meters per second and yaw angle rate in radians per second:

$$\boldsymbol{x}_{\text{lat}} = \begin{bmatrix} V_y \\ \\ \dot{\psi} \end{bmatrix} \qquad \boldsymbol{u}_{\text{lat}} = \delta \tag{5.10}$$

while the matrices $\boldsymbol{A}_g$,$\boldsymbol{B}_g$,$\boldsymbol{C}_g$ and $\boldsymbol{D}_g$ are the following:

$$\begin{gathered} \boldsymbol{A}_g = \begin{bmatrix} -\dfrac{2C_F + 2C_R}{mV_x} & -\dfrac{2C_F l_F - 2C_R l_R}{mV_x} - V_x \\ -\dfrac{2C_F l_F - 2C_R l_R}{I_Z V_x} & -\dfrac{2C_F l_F^2 + 2C_R l_R^2}{I_Z V_x} \end{bmatrix}, \\ \boldsymbol{B}_g = \begin{bmatrix} 2C_F/m \\ \\ 2C_F l_F / I_Z \end{bmatrix}, \qquad \boldsymbol{C}_g = \begin{bmatrix} 1 & 0 \\ \\ 0 & 1 \end{bmatrix} = \boldsymbol{I}_2, \qquad \boldsymbol{D}_g = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix} = \boldsymbol{0}_{2\times 1}. \end{gathered} \tag{5.11}$$

Recapping the parameters in the previous matrices are:

- V_x is the longitudinal velocity of the car;
- m is the total mass parameter;
- I_Z is the yaw moment of inertia parameter;
- l_F is the longitudinal distances from center of gravity to front tire parameter;

- l_R is the longitudinal distances from center of gravity to rear tire parameter;
- C_F is the cornering stiffnesses of front tire parameter;
- C_R is the cornering stiffnesses of rear tire parameter;

5.1.3 Augmented Model for Lateral Dynamics

The goal for the driver steering model is to keep the vehicle in its lane and follow the curved road by controlling the front steering angle. Denote by e_1 the offset to the center line and e_2, the relative angle to the center line. This goal is achieved by driving the yaw angle error $e_2 = \psi - \psi_{\text{des}}$ and lateral displacement error e_1 to zero ($\dot{e}_1 = V_x e_2 + V_y$). We can incorporate these two paramenters in the augmented model:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\text{aug}} &= \boldsymbol{A}_a \boldsymbol{x}_{\text{aug}} + \boldsymbol{B}_a \boldsymbol{u}_{\text{aug}} \\ \boldsymbol{y}_{\text{aug}} &= \boldsymbol{C}_a \boldsymbol{x}_{\text{aug}} + \boldsymbol{D}_a \boldsymbol{u}_{\text{aug}} \end{aligned} \tag{5.12}$$

where the states and the inputs are:

$$\boldsymbol{x}_{\text{aug}} = \begin{bmatrix} V_y \\ \dot{\psi} \\ e_1 \\ e_2 \end{bmatrix}, \qquad \boldsymbol{u}_{\text{aug}} = \begin{bmatrix} \delta \\ \dot{\psi}_{\text{des}} \end{bmatrix} \tag{5.13}$$

while the matrices $\boldsymbol{A}_a$,$\boldsymbol{B}_a$,$\boldsymbol{C}_a$ and $\boldsymbol{D}_a$ are the following:

$$\boldsymbol{A}_a = \begin{bmatrix} \boldsymbol{A}_g & \boldsymbol{0}_{2\times 2} \\ & 0 \quad V_x \\ \boldsymbol{I}_2 & \\ & 0 \quad 0 \end{bmatrix}, \qquad \boldsymbol{B}_a = \begin{bmatrix} \boldsymbol{B}_g & \boldsymbol{0}_{2\times 1} \\ 0 & 0 \\ 0 & -1 \end{bmatrix}, \tag{5.14}$$

$$\boldsymbol{C}_a = \begin{bmatrix} \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_2 \end{bmatrix}, \qquad \boldsymbol{D}_a = \boldsymbol{0}_{2\times 2}.$$

5.1.4 Overall Model Dynamics

Combining (5.2) with (5.12) yields the state-space model that characterizes the Model Predictive Control problem:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\text{tot}} &= \boldsymbol{A}_f \boldsymbol{x}_{\text{tot}} + \boldsymbol{B}_f \boldsymbol{u}_{\text{tot}} \\ \boldsymbol{y}_{\text{tot}} &= \boldsymbol{C}_f \boldsymbol{x}_{\text{tot}} + \boldsymbol{D}_f \boldsymbol{u}_{\text{tot}} \end{aligned} \tag{5.15}$$

where the states and the inputs are:

$$\boldsymbol{x}_{\text{tot}} = \begin{bmatrix} \boldsymbol{x}_{\text{lon}} \\ \boldsymbol{x}_{\text{aug}} \end{bmatrix} = \begin{bmatrix} \dot{V}_x \\ V_x \\ V_y \\ \dot{\psi} \\ e_1 \\ e_2 \end{bmatrix}, \qquad \boldsymbol{u}_{\text{tot}} = \begin{bmatrix} \boldsymbol{u}_{\text{lon}} \\ \boldsymbol{u}_{\text{aug}} \end{bmatrix} = \begin{bmatrix} a \\ \delta \\ \dot{\psi}_{\text{des}} \end{bmatrix} \tag{5.16}$$

while the matrices $\boldsymbol{A}_f$,$\boldsymbol{B}_f$,$\boldsymbol{C}_f$ and $\boldsymbol{D}_f$ are the following:

$$\boldsymbol{A}_f = \begin{bmatrix} \boldsymbol{A}_m & \boldsymbol{0}_{2\times 4} \\ \boldsymbol{0}_{4\times 2} & \boldsymbol{A}_a \end{bmatrix}, \qquad \boldsymbol{B}_f = \begin{bmatrix} \boldsymbol{B}_m & \boldsymbol{0}_{2\times 2} \\ \boldsymbol{0}_{4\times 1} & \boldsymbol{B}_a \end{bmatrix},$$
$$\boldsymbol{C}_f = \begin{bmatrix} \boldsymbol{C}_m & \boldsymbol{0}_{1\times 4} \\ \boldsymbol{0}_{2\times 2} & \boldsymbol{C}_a \end{bmatrix}, \qquad \boldsymbol{D}_f = \boldsymbol{0}_{3\times 3}. \tag{5.17}$$

However the system to be controlled is usually modeled by a linear discrete state-space model:

$$\begin{aligned} \boldsymbol{x}_{\text{tot}}(k+1) &= \boldsymbol{A}\boldsymbol{x}_{\text{tot}}(k) + \boldsymbol{B}\boldsymbol{u}_{\text{tot}}(k) \\ \boldsymbol{y}_{\text{tot}}(k) &= \boldsymbol{C}\boldsymbol{x}_{\text{tot}}(k) + \boldsymbol{D}\boldsymbol{u}_{\text{tot}}(k) \end{aligned} \tag{5.18}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the state and control matrices for the discrete state-space equation, respectively, which can be calculated, also in this case, with the Euler method as:

$$\boldsymbol{A} = e^{\boldsymbol{A}_f T_s}, \qquad \boldsymbol{B} = \int_{kT_s}^{(k+1)T_s} e^{\boldsymbol{A}_f[(k+1)T_s - \eta]} \boldsymbol{B}_f d\eta \tag{5.19}$$

where T_s is the sampling interval for the discrete state-space model. The matrices $\boldsymbol{C}$ and $\boldsymbol{D}$ are equivalent to those in the continuous case.

5.2 Design of Adaptive Model Predictive Control

Before talking about the design of the controller, it is necessary to highlight which are all the elements that make up our control scheme. The overall framework for a lane keeping assist is depicted in the Figure 5.2.

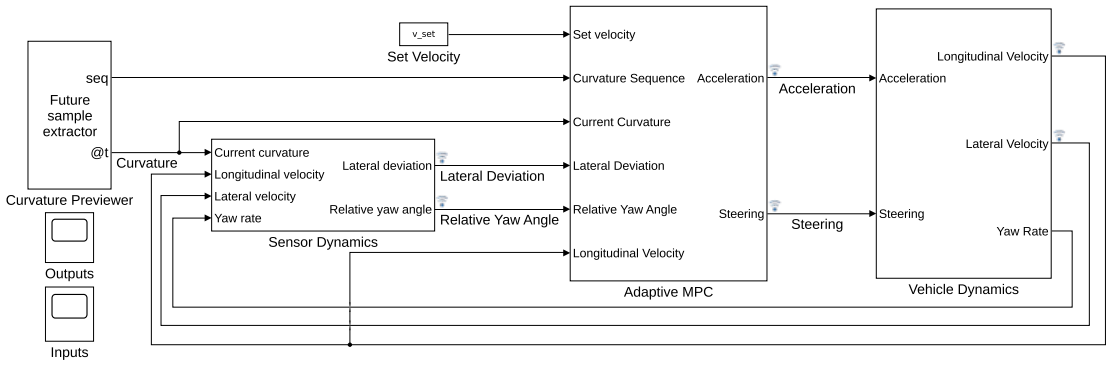


Figure 5.2: Overall procedure scheme lane following.

This control scheme is composed by four different blocks that are the Sensor Dynamics, the Vehicle Dynamics, the Adaptive MPC controller and the Curvature Previewer. Summarizing, the Vehicle Dynamics applies the bicycle model of lateral vehicle dynamics and approximate the longitudinal dynamics using a time constant according to what was seen previously; the Sensor Dynamics approximates a sensor, such as a camera or a laser, to calculate the lateral deviation and relative yaw angle; the Curvature Previewer detects the curvature at the current time step and the curvature sequence over the prediction horizon of the MPC controller and finally Adaptive MPC generates the optimal control inputs for a lane following system. The objective of the trajectory planning along specified path can be described as follows: given a path which the vehicle is expected to follow design a trajec-

tory of a car-vehicle configuration. In order to do this it is possible to derive the road curvature and its derivative. For a plane curve given parametrically in Cartesian coordinates as $\gamma(t) = (x(t), y(t))$, the signed curvature is

$$\kappa = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{\frac{3}{2}}} \tag{5.20}$$

where primes refer to derivatives $\frac{d}{dt}$ with respect to the parameter t. The resulting curvature (5.20) is a rational function, continuous except at the roots of the denominator, which is always nonzero provided that the interpolated positions do not contain subsequent duplicate values and that the points are spaced sufficiently [40]. The vehicle dynamics is represented by a Simulink model in Figure 5.3 where on the upper part we find the longitudinal dynamics while on the lower part we can see the lateral dynamics highlighted by the gains that are the components of the linear discrete state-space model (5.9).

We created an Adaptive MPC controller with a prediction model that has six states, three outputs (longitudinal velocity, lateral deviation, relative yaw angle), two manipulated signals (acceleration and steering) and one measured disturbance (desired yaw rate).

In order to design a valid lane keeping algorithm based on MPC we have set the constraints for manipulated variables and the scale factors. In particular the control variables are constrained as follows:

$$-3\,\mathrm{m/s^2} \leq a \leq 3\,\mathrm{m/s^2} \qquad -1.13\,\mathrm{rad} \leq \delta \leq 1.13\,\mathrm{rad}\ , \tag{5.21}$$

Moreover we have specified the weights in the standard MPC cost function. The third output, yaw angle, is allowed to float because there are only two manipulated variables to make it a square

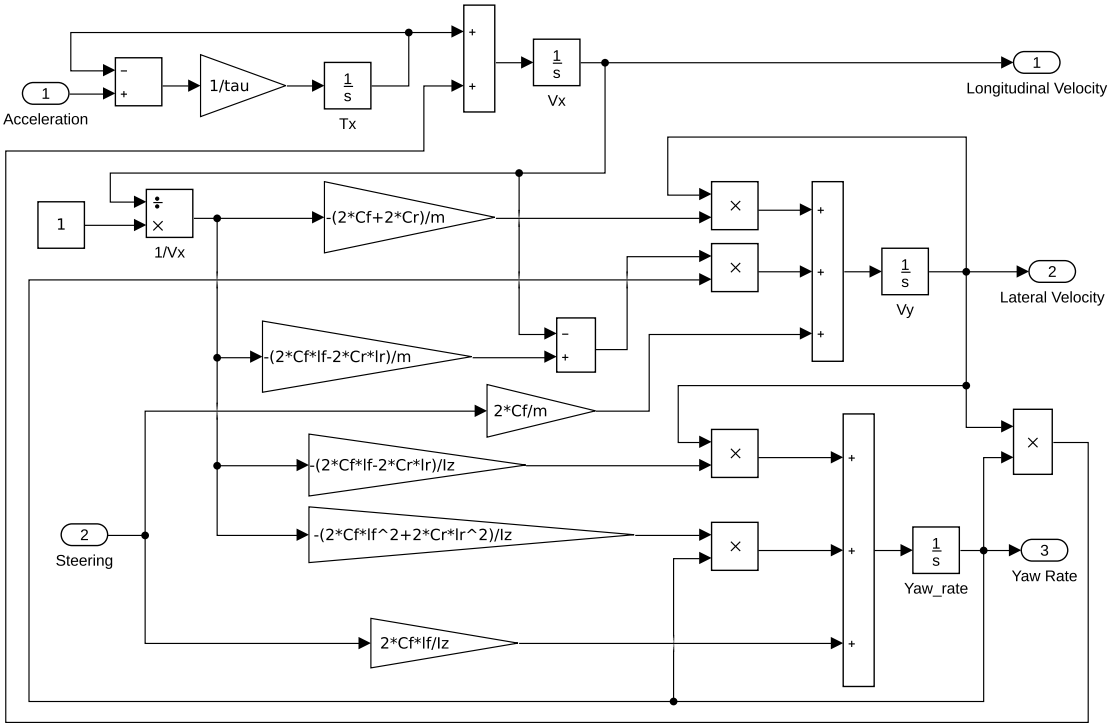


Figure 5.3: Vehicle Dynamics of the overall scheme lane following.

system. In this controller, there is no steady-state error in the yaw angle as long as the second output, lateral deviation, reaches 0 at steady state. Finally we have also penalized acceleration change more for smooth driving experience. This controller uses a linear model for the vehicle dynamics and updates the model online as the longitudinal velocity varies. The dynamics of the sensors (Figure 5.4) allows us to calculate two fundamental parameters to reach the goal of keeping the vehicle in its lane and following the curved road by controlling the front steering angle:

- e_1 which is the offset to the center line,
- e_2 which is the relative angle to the center line.

where we want to drive the yaw angle error and the lateral displacement error to zero to achieve

our goal. The product of the road curvature and the longitudinal velocity is modeled as a measured disturbance.

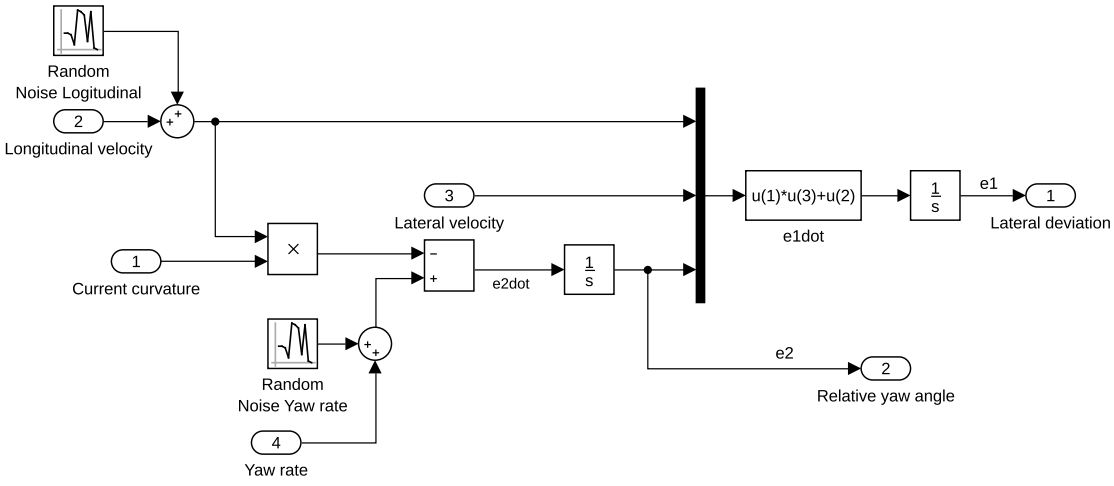


Figure 5.4: Sensor Dynamics of the overall scheme lane following.

5.3 Simulation Results

The proposed adaptive MPC algorithm is designed in the MATLAB/Simulink and validated through simulations considering different paths. The objective of these tests is to evaluate the behavior of the proposed control strategy in critical situations. The parameters used in the lane following simulations are summarized in table 5.1:

The paths considered for the evaluation of the proposed control strategy are:

- a *double lane change* curve (useful when a vehicle has to pass an obstacle),
- a *sinusoidal* path (slalom cones scenario),
- a *circular/elliptic* path (NASCAR race tracks).

Parameters	Values	Description
m	$1575\,\mathrm{kg}$	Mass of the vehicle
I_z	$2875\,\mathrm{kgm}^2$	Inertia Moment
l_F	$1.2\,\mathrm{m}$	Distance from COG to front axle
l_R	$1.6\,\mathrm{m}$	Distance from COG to rear axle
C_F	$19\,000\,\mathrm{N/rad}$	Front cornering stiffness
C_R	$33\,000\,\mathrm{N/rad}$	Rear cornering stiffness
τ	0.2	Time constant
V_0	$15\,\mathrm{m/s}$	Initial Velocity
V_{set}	$20\,\mathrm{m/s}$	Driver-set Velocity
T_s	$0.02\,\mathrm{s}$	Sampling Time

Table 5.1: Parameters of Vehicle Dynamics and Road Curvature.

5.3.1 Double Lane Change Path

In the first simulation the desired path is described in terms of the lateral position Y_{ref} as function of the longitudinal position X_{ref}. The equations (5.22) describe a double lane change that have been employed in different tests for different scenarios i.e. [41] [42] [43] as follow :

$$X_{\text{ref}} = V_x \cdot t, \quad \text{with} \quad t \in [0, 10]\text{s}$$

$$z_1 = \frac{2.4}{50}(X_{\text{ref}} - 27.19) - 1.2; \quad z_2 = \frac{2.4}{43.9}(X_{\text{ref}} - 56.46) - 1.2; \tag{5.22}$$

$$Y_{\text{ref}} = \frac{8.1}{2}(1 + \tanh(z_1)) - \frac{8.4}{2}(1 + \tanh(z_2)).$$

Figures 5.5a and 5.5b represent the first trajectory that the vehicle must perform and the curvature calculated with the previous formula 5.20.

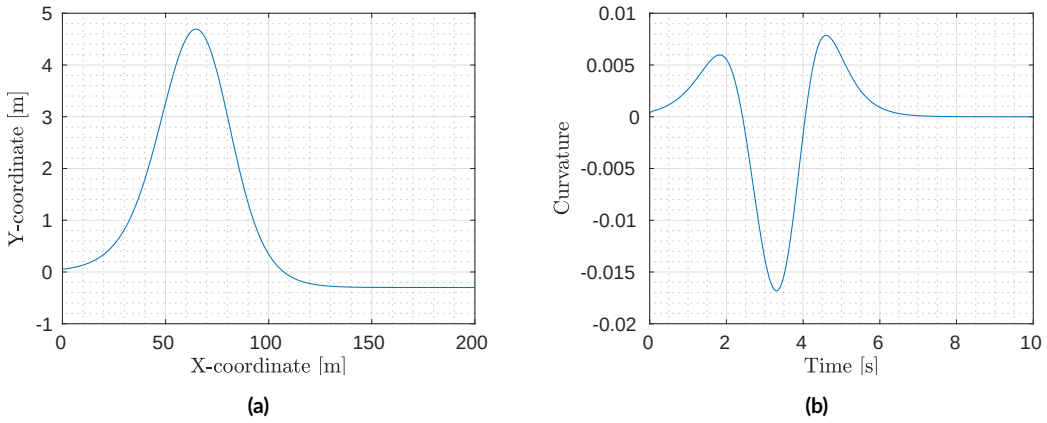
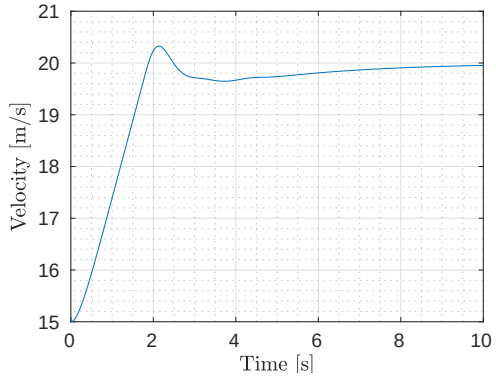
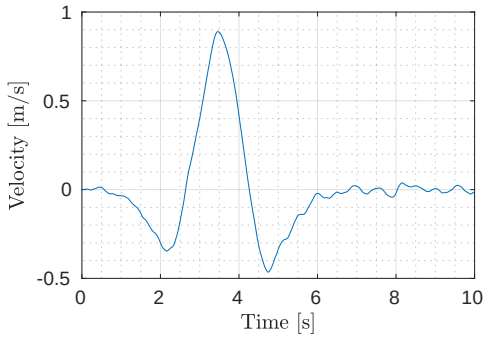


Figure 5.5: Desired double lane change path and its curvature of the ATLASCAR2 in a simulation of $10\,\mathrm{s}$ with initial position in $(0, 0)$.

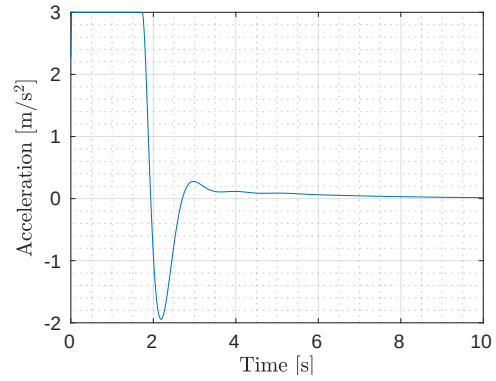
From the following figures, on the other hand, it can be deduced that the control system we have created allows the ATLASCAR2 to respect the imposed constraints and to reach the set objectives. The longitudinal and lateral components of the velocity are represented in figures 5.6a and 5.6b. In the first few seconds of the simulation V_x increases thanks to a constant acceleration imposed by the first input (Figure 5.6c). Later this acceleration decreases converging to zero so that at run time, we can note that V_x reaches the predefined value of $20\,\mathrm{m/s}$ and then it stabilizes near the driver set velocity because it continues to vary the steering angle to adapt to the path to be followed (Figure 5.6d). The lateral component V_y, on the other hand, only affects during the steering phase when we are forcing a change of direction. From Figures 5.6f and 5.6e we can notice that the lateral deviation and the relative yaw angle both converge to zero. That is, the ATLASCAR2 follows the road closely based on the previewed curvature.



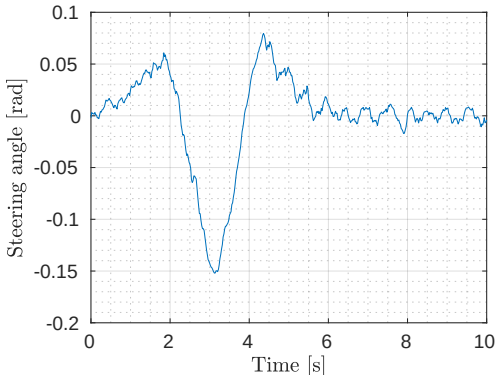
(a) Longitudinal velocity V_x w.r.t. time.



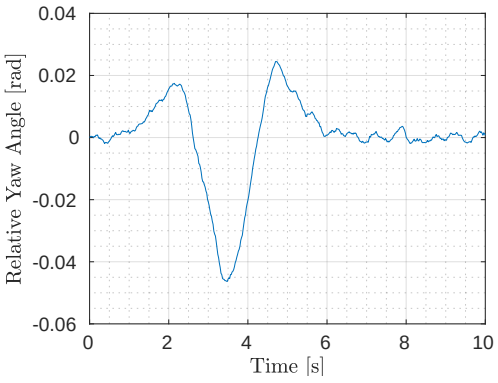
(b) Lateral velocity V_y w.r.t. time.



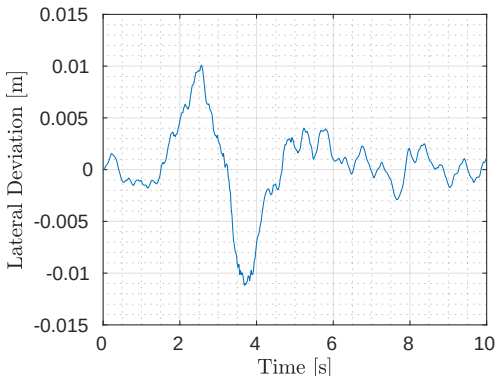
(c) Acceleration u_1 w.r.t. time.



(d) Steering angle u_2 w.r.t. time.



(e) Relative yaw angle e_2 w.r.t. time.



(f) Lateral deviation e_1 w.r.t. time.

Figure 5.6: Time signals of the ATLASCAR2 in the simulation with a double lane change path.

5.3.2 Sinusoidal Path

In the second simulation we assumed that the car must follow a sinusoidal path. We have extended the simulation time to 20 seconds in order to better understand the progress of the various signals. Figures 5.7a and 5.7b show the desired path and its curvature, where the former is described in terms of the lateral position Y_{ref} as function of the longitudinal position X_{ref} and the latter is derived according with [44]. The ATLASCAR2 is controlled to follow a sinusoidal trajectory which is given as follows:

$$\begin{aligned} X_{\text{ref}} &= V_x \cdot t, \quad \text{with} \quad t \in [0, 20]\text{s} \\ Y_{\text{ref}} &= 5 \sin\left(\frac{X_{\text{ref}}}{20}\right) \end{aligned} \tag{5.23}$$

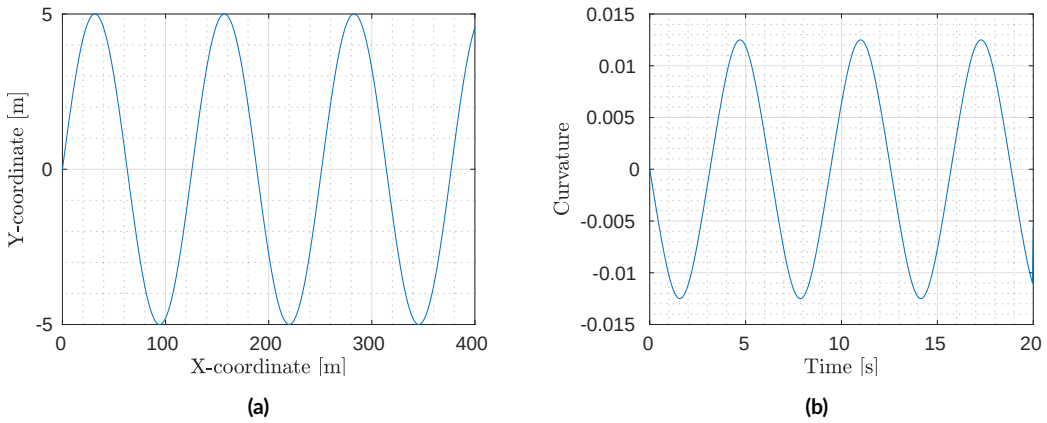


Figure 5.7: Desired sinusoidal path and its curvature of the ATLASCAR2 in a simulation of $20\,\text{s}$.

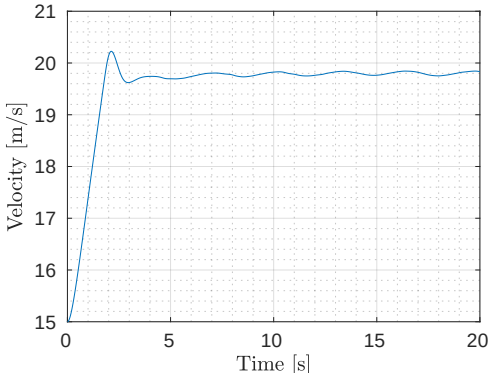
In practice this type of route can be followed by the vehicle when it has to make a slalom between cones or it must go up some hairpin bends in the mountains. Moreover the following figures show the trend of the main parameters confirming that the control strategy used allows the vehicle to fol-

low the path. In particular we simulated also a small error in the sensor dynamics in order to make the simulation more realistic: we added a 3 percent error to the longitudinal velocity and this is evident from the small noise in the graphs of the steering angle (Figure 5.8d) and the lateral deviation (Figure 5.8f). Figure 5.8a shows the evolution of the vehicle longitudinal velocity. Also in this simulation the velocity is equal to the initial condition for longitudinal velocity parameter V_0. At run time, we can note that V_x reaches the predefined value of $20\,\mathrm{m/s}$ and then it stabilizes near the cruising speed because it continues to vary the steering angle to adapt to the path to be followed. In this scenario the lateral deviation and the relative angle yaw do not converge to zero but oscillate. We can however assert that this type of oscillation is due to the type of path we are following. Furthermore the limits of this oscillation are very small indeed: in particular the lateral deviation is less than $1.5\,\mathrm{cm}$ while the maximum relative yaw angle is $0.04\,\mathrm{rad}$. We can therefore state that even in this case the vehicle correctly follows the lane.

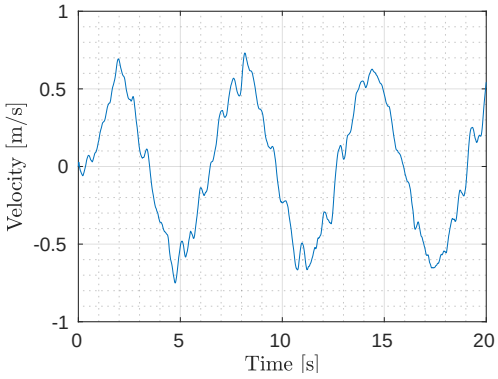
5.3.3 Circular/Elliptic Path

In the last simulation we considered that the vehicle must follow an elliptic path with a semi-major axis of $30\,\mathrm{m}$ and semi-minor axis of $15\,\mathrm{m}$ depicted in Figure 5.9a. Also in this case the simulation time is $20\,\mathrm{s}$. To describe the elliptic path and to evaluate its curvature, depicted in Figure 5.9b, the equations are as follows:

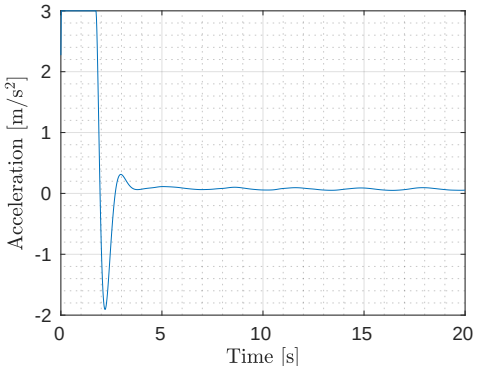
$$X_{\text{ref}} = 15\cos\left(\frac{V_x \cdot t}{60}\right) \qquad Y_{\text{ref}} = 30\sin\left(\frac{V_x \cdot t}{60}\right) \quad \text{with} \quad t \in [0, 20]\mathrm{s} \tag{5.24}$$



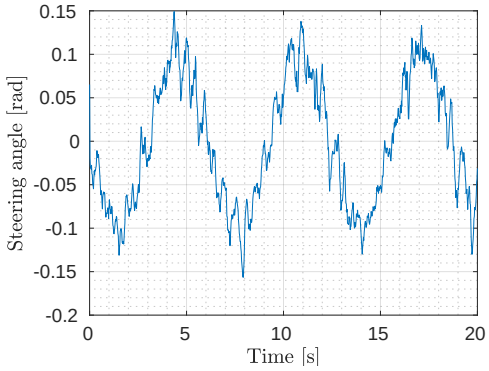
(a) Longitudinal velocity V_x w.r.t. time.



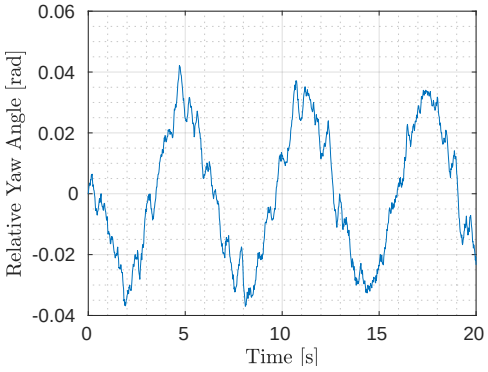
(b) Lateral velocity V_y w.r.t. time.



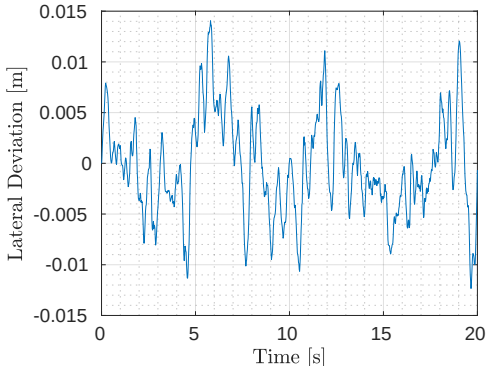
(c) Acceleration u_1 w.r.t. time.



(d) Steering angle u_2 w.r.t. time.



(e) Relative yaw angle e_2 w.r.t. time.



(f) Lateral deviation e_1 w.r.t. time.

Figure 5.8: Time signals of the ATLASCAR2 in the simulation with a sinusoidal path.

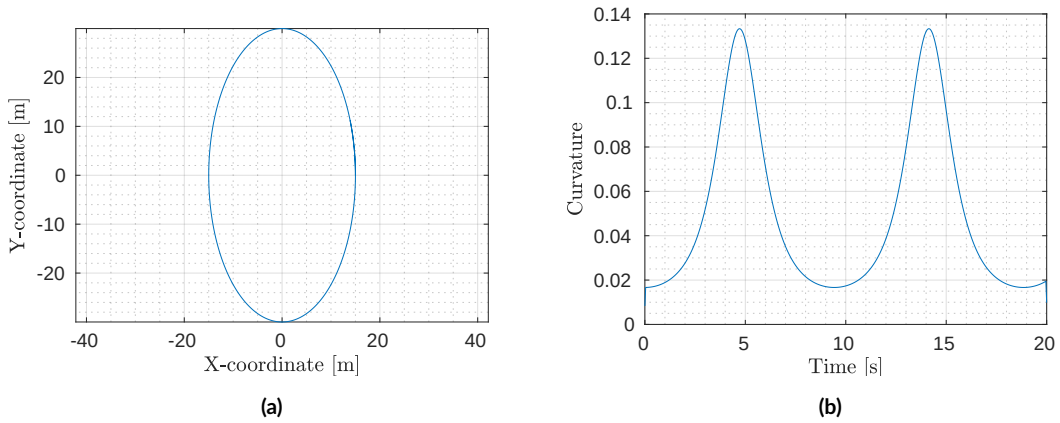


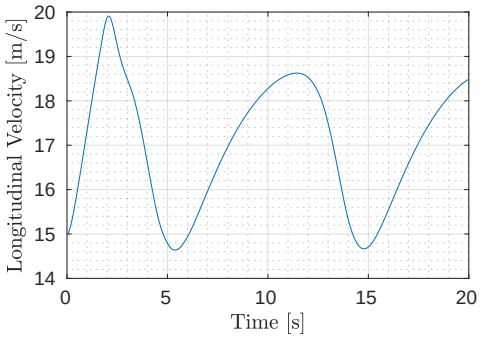
Figure 5.9: Desired elliptic path and its curvature of the ATLASCAR2 in a simulation of $20\,\mathrm{s}$ with initial position in $(0, 30)$.

In particular in Figure 5.10 are represented the time signals of the ATLASCAR2 in the simulation with the elliptic path. It is possible to notice that also in this case the vehicle follows the path with a minimal lateral deviation described in the Figure 5.10f.

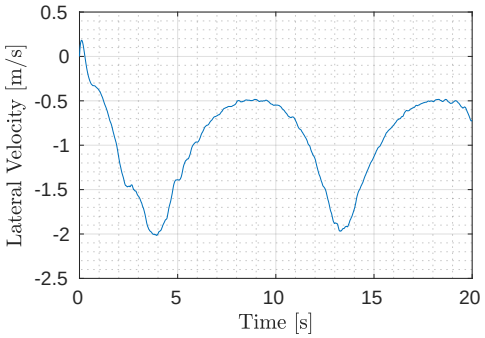
However, the minimum lateral displacement is at the expense of the costant cruise velocity; infact Figure 5.10a and 5.10b show an oscillatory pattern of both longitudinal speed and lateral velocity caused by the continuous change of direction of the road and the length of the latter.

Also in this situation we simulated a small error in the sensor dynamics in order to make the scenario more realistic: we added a 2 percent error to the longitudinal velocity and this is evident from the small noise in the graphs of the steering angle (Figure 5.10d) and the lateral deviation (Figure 5.10f).

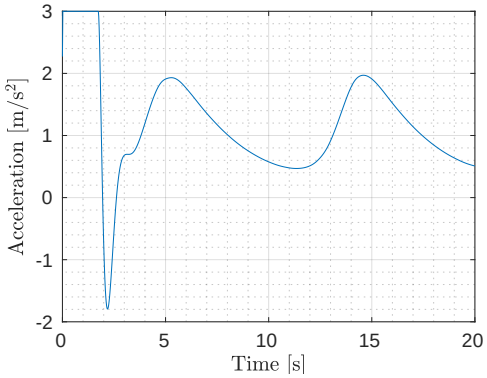
As previously stated, the vehicle follows the lane and allows us to state that the control system we have designed is efficient and robust.



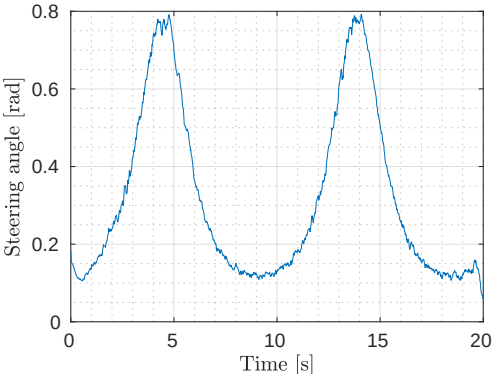
(a) Longitudinal velocity V_x w.r.t. time.



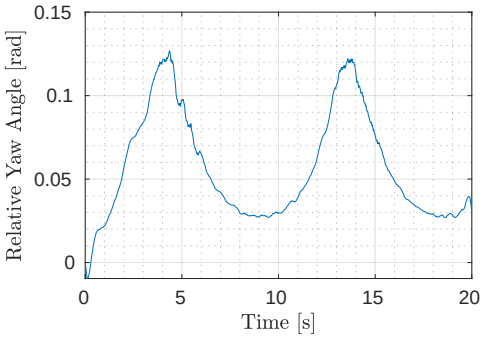
(b) Lateral velocity V_y w.r.t. time.



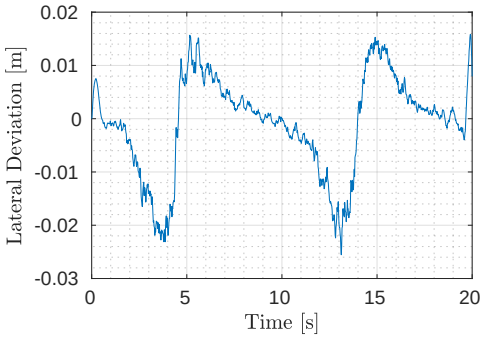
(c) Acceleration u_1 w.r.t. time.



(d) Steering angle u_2 w.r.t. time.



(e) Relative yaw angle e_2 w.r.t. time.



(f) Lateral deviation e_1 w.r.t. time.

Figure 5.10: Time signals of the ATLASCAR2 in the simulation with an elliptic path.

6

Conclusions and Future Work

This thesis proposes two advanced methods for short term motion planning of an autonomous car based on adaptive Model Predictive Control. The first component is an obstacle avoidance system that moves the vehicle around different moving obstacles in the lane using throttle and steering angle. This system updates both the predictive model and the mixed input/output constraints at each control interval. The vehicle is also able to brake in order to prevent collisions against closest

obstacles. Instead, in the second scheme we have developed a lane following system that keeps the ATLASCAR2 traveling along the centerline of the lanes on the road by adjusting the front steering angle of the car. The flexibility of the concepts used in the algorithms allows a multitude of refinements and extensions to this work. The future work includes the combination of these two control strategies in a way that they can operate simultaneously. In order to achieve this objective we have started to simulate a trajectory tracking framework using a slight modification of the linearized tracking error model in [45]. Next expected steps include the migration to ROS-Gazebo simulation environment and, later on, the usage of real data collected on board the ATLASCAR2 and, ultimately, test it in a real autonomous driving scenario.

References

[1] J. Skoda, "3D Navigation for Mobile Robots," Master's thesis, Charles University, Prague, 2016.

[2] J. Bai, "Robot Navigation Using Velocity Potential Fields and Particle Filters for Obstacle Avoidance," Master's thesis, University of Ottawa, Ottawa, 2015.

[3] D. Gruyer, V. Magnier, K. Hamdi, L. Claussmann, O. Orfila, and A. Rakotonirainy, "Perception, information processing and modeling: Critical stages for autonomous driving applications," *Annual Reviews in Control*, vol. 44, pp. 323–341, 2017.

[4] WHO, *Global Status Report on Road Safety 2015*. Nonserial Publication, World Health Organization, 2015.

[5] C. Katrakazas, M. Quddus, W. H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[6] V. Santos, J. Almeida, E. Avila, D. Gameiro, M. Oliveira, R. Pascoal, R. Sabino, and P. Stein, "ATLASCAR - technologies for a computer assisted driving system on board a common

automobile," in *13th International IEEE Conference on Intelligent Transportation Systems, Funchal, Madeira, Portugal, 19-22 September 2010*, pp. 1421–1427, 2010.

[7] J. F. Pereira, "Estacionamento autónomo usando perceção 3D Autonomous parking using 3D perception," dissertação de mestrado, Departamento de Engenharia Mecânica, Universidade de Aveiro, 2012.

[8] R. F. Cabral de Azevedo, "Sensor Fusion of LASER and Vision in Active Pedestrian Detection," dissertação de mestrado, Departamento de Engenharia Mecânica, Universidade de Aveiro, 2014.

[9] D. T. Vieira da Silva, "Calibração Multissensorial e Fusão de Dados Utilizando LIDAR e Visão," dissertação de mestrado, Departamento de Engenharia Mecânica, Universidade de Aveiro, 2016.

[10] J. D. Madureira Correia, "Unidade de Perceção Visual e de Profundidade Para o Atlascar2," dissertação de mestrado, Departamento de Engenharia Mecânica, Universidade de Aveiro, 2017.

[11] R. Silva, "ATLASCAR2 Local Navigation for Autonomous Driving and Driver Assistance," Master's thesis, Universidade de Aveiro, Aveiro, 2018.

[12] Wikipedia, "Self-driving car." `https://en.wikipedia.org/wiki/Self-driving_car`, 2019.

[13] K. Bimbraw, "Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology," *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics*, no. August, pp. 191–198, 2015.

[14] Waymo, "Waymo - journey." `https://waymo.com/journey/`, 2018.

[15] Waymo, "Waymo - technology." `https://waymo.com/tech/`, 2018.

[16] A. Broggi, P. Medici, E. Cardarelli, P. Cerri, A. Giacomazzo, and N. Finardi, "Development of the control system for the vislab intercontinental autonomous challenge," in *13th International IEEE Conference on Intelligent Transportation Systems*, pp. 635–640, Sep. 2010.

[17] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari, "Extensive tests of autonomous driving technologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1403–1415, 2013.

[18] R. Zhang and M. Pavone, "Control of robotic mobility-on-demand systems: A queueing-theoretical perspective," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 186–203, 2016.

[19] V. Santos, "ATLASCAR: A sample of the quests and concerns for autonomous cars," in *Informatics in Control, Automation and Robotics: 14th International Conference, ICINCO 2017 Madrid, Spain, July 26-28, 2017 Revised Selected Papers* (O. Gusikhin and K. Madani, eds.), Lecture Notes in Electrical Engineering, Springer International Publishing, 2019.

[20] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, Dec. 1959.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.

[22] A. Stentz and I. C. Mellon, "Optimal and efficient path planning for unknown and dynamic environments," *International Journal of Robotics and Automation*, vol. 10, pp. 89–100, 1993.

[23] O. Khatib, *The Potential Field Approach And Operational Space Formulation In Robot Control*, pp. 367–377. Boston, MA: Springer US, 1986.

[24] J. Borenstein and Y. Koren, "The vector field histogram – fast obstacle avoidance for mobile robots," *IEEE JOURNAL OF ROBOTICS AND AUTOMATION*, vol. 7, pp. 278–288, 1991.

[25] J. Minguez and L. Montano, "Nearness diagram navigation: a new real time collision avoidance approach," vol. 3, pp. 2094 – 2100, February 2000.

[26] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3375–3382 vol.4, April 1996.

[27] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, pp. 23–33, March 1997.

[28] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *2013 European Control Conference (ECC)*, pp. 4136–4141, July 2013.

[29] M. F. Manzoor and Q. Wu, "Control and obstacle avoidance of wheeled mobile robot," in *2015 7th International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 235–240, June 2015.

[30] M. Werling and D. Liccardo, "Automatic collision avoidance using model-predictive online optimization," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 6309–6314, Dec 2012.

[31] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 952–964, Feb 2017.

[32] N. Wada and T. Matsumoto, "Driver assistance for collision avoidance by constrained mpc," in *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pp. 90–93, Sep. 2017.

[33] A. Bemporad, M. Morari, and N. L. Ricker, "Model predictive control toolbox: Getting started guide, the math works." `https://www.mathworks.com/help/pdf_doc/mpc/mpc_ug.pdf`, 2018.

[34] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st ed., 2008.

[35] T. Xu and H. Yuan, "Autonomous vehicle active safety system based on path planning and predictive control," in *2016 35th Chinese Control Conference (CCC)*, pp. 8889–8895, July 2016.

[36] W. Xi and J. S. Baras, "Mpc based motion control of car-like vehicle swarms," in *2007 Mediterranean Conference on Control Automation*, pp. 1–6, June 2007.

[37] M. Bujarbaruah, X. Zhang, H. E. Tseng, and F. Borrelli, "Adaptive MPC for autonomous lane keeping," *CoRR*, vol. abs/1806.04335, 2018.

[38] S. Li, K. Li, R. Rajamani, and J. Wang, "Model predictive multi-objective vehicular adaptive cruise control," *IEEE Transactions on Control Systems Technology*, vol. 19, pp. 556–566, May 2011.

[39] V. L. Bageshwar, W. L. Garrard, and R. Rajamani, "Model predictive control of transitional maneuvers for adaptive cruise control vehicles," *IEEE Transactions on Vehicular Technology*, vol. 53, pp. 1573–1585, Sep. 2004.

[40] J. Filip, "Trajectory tracking for autonomous vehicles," Master's thesis, Czech Technical University, 2018.

[41] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "A model predictive control approach for combined braking and steering in autonomous vehicles," in *2007 Mediterranean Conference on Control Automation*, pp. 1–6, June 2007.

[42] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on Control Systems Technology*, vol. 15, pp. 566–580, May 2007.

[43] T. Keviczky, P. Falcone, F. Borrelli, J. Asgari, and D. Hrovat, "Predictive control approach to autonomous vehicle steering," in *2006 American Control Conference*, pp. 6 pp.–, June 2006.

[44] D. Wang and F. Qi, "Trajectory planning for a four-wheel-steering vehicle," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 4, pp. 3320–3325, May 2001.

[45] H. Yang, M. Guo, Y. Xia, and L. Cheng, "Trajectory tracking for wheeled mobile robots via model predictive control with softening constraints," *IET Control Theory Applications*, vol. 12, no. 2, pp. 206–214, 2018.