UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in

"Tecnologie dell'Informazione"

Ciclo XXXVI

Trajectory Generation Strategies for Safe Autonomous Driving in Urban Scenario

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Ph.D. Ing. Alessandro Rucco

Chiar.mo Prof. Massimo Bertozzi

Dottorando: *Francesco Laneve*

Anni Accademici 2020/2021 - 2022/2023

Alla mia famiglia

"Suppose that we are wise enough to learn and know – and yet not wise enough to control our learning and knowledge, so that we use it to destroy ourselves? Even if that is so, knowledge remains better than ignorance. It is better to know? even if the knowledge endures only for the moment that comes before destruction? than to gain eternal life at the price of a dull and swinish lack of comprehension of a universe that swirls unseen before us in all its wonder. That was the choice of Achilles, and it is mine, too."

– Isaac Asimov

Abstract

In this dissertation we develop novel strategies based on nonlinear optimal control techniques for trajectory generation of autonomous vehicles. These strategies are designed to enable the development of autonomous vehicles that navigate dynamic environments while enhancing safety and passenger comfort.

In the first part of the work, we introduce a family of reduced-order car models suited for trajectory generation strategies. We derive the equations of motion for both kinematic and dynamic bicycle models, the latter of which includes tire modeling for a more realistic representation of vehicle behavior. We re-write the kinematic model in terms of longitudinal and lateral coordinates, aligning them with the way humans perceive and control vehicle motion.

In the second part, we propose an optimization-based strategy to address merging maneuvers in busy intersections. We describe vehicle dynamics in terms of longitudinal and transverse coordinates and introduce a "virtual target vehicle" constrained to move within the target lane for merging. We formulate an optimal control problem in terms of longitudinal and lateral coordinates, including the kinematic position error between the autonomous vehicle and the virtual target vehicle. We also use obstacle predictions to enforce suitable kinematic constraints for generating collision-free trajectories. We show the efficacy of the proposed strategy through a set of numerical computations and highlight the main features of the generated trajectories.

In the third part, we present a real-time maneuver generation algorithm. Given a planar road geometry with static and moving obstacles along it, we are interested in finding collision-free maneuvers that satisfied

the vehicle dynamics and subject to physical and comfort limits. Based on longitudinal and transverse coordinates, we propose a novel collision avoidance constraint and formulate a suitable optimal control problem. The optimization problem is solved by using a nonlinear optimal control technique that generates (local) optimal trajectories. We demonstrate the efficacy of the proposed algorithm by providing numerical computations on a simulated scenario. Experimental results are presented to demonstrate the efficiency of the proposed algorithm both in terms of computational effort and dynamic features captured.

In the fourth part, we address the lane change maneuver using a parametric model predictive control approach. We recognize that successful lane changes involve both the decision of when to initiate these maneuvers and the generation of collision-free trajectories. Our approach combines decision-making and planning tasks, guiding the low-level policy through upper-level policy search. Additionally, we incorporate self-supervised learning techniques to adapt to dynamic, online scenarios, ensuring the vehicle can handle unexpected changes in its environment. We provide numerical results that highlight the effectiveness of this approach in improving vehicle maneuvering in dynamic environments.

Keywords: Nonlinear optimal control, trajectory generation, model predictive control, reinforcement learning, autonomous vehicles

Contents

List of Figures

List of Tables

Introduction

Motivation of the Work

Autonomous vehicles represent a groundbreaking innovation in the field of transportation, poised to transform the way we move people and goods. The promise of autonomous vehicles lies in their ability to offer a multitude of benefits, ranging from enhancing safety on the roads, to significantly reducing travel times, and even contributing to a more sustainable future by minimizing energy consumption. These potential advantages underscore the transformative power of autonomous vehicles in reshaping our mobility landscape. However, navigating through the complex and dynamic environments of real-world roads is not a simple task. To achieve these ambitious goals, autonomous vehicles must perform different challenge maneuvers. For example, these maneuvers include: lane changes, overtaking slower vehicles, and merging into traffic. Achieving these tasks necessitates the development and implementation of advanced algorithms that empower an autonomous vehicle to interact with its environment intelligently and safely. Typically, at the heart of this autonomous driving capability lies a chain of sub-tasks. First, the autonomous vehicle needs to perceive its surroundings. This perception is not limited to static objects like road signs and traffic lights but extends to dynamic entities such as pedestrians, cyclists, and other vehicles. Then,

the autonomous vehicles must predict the future intentions and actions of these various traffic agents. This predictive capability is crucial for anticipating potential hazards and ensuring a smooth and safe driving. With this information in hand, the autonomous vehicle's high-level planning module comes into play. It is responsible for generating a feasible trajectory that allows the autonomous vehicle to navigate its surroundings while avoiding obstacles, maintaining safe distances, and adhering to traffic rules. This planning phase is a critical step, as it determines the vehicle's path through the environment. Once the high-level plan is established, the autonomous vehicle's low-level controller module takes over. This module is responsible for executing the planned trajectory, ensuring that the vehicle stays on course and responds dynamically to any unexpected changes in the environment. Together, these components of perception, prediction, planning, and control form the intricate web of capabilities that enable an autonomous vehicle to operate effectively and safely in real-world driving scenarios. The ultimate goal is to achieve human-level reliability, where autonomous vehicles navigate with a level of safety and efficiency that rivals or surpasses human drivers. In this dissertation, we aim to develop efficient tools for designing autonomous vehicles in dynamic environments and assess their effectiveness through simulation and real field tests. We follow three perspectives, aimed at reaching the aforementioned objectives.

First, we focus on the development of a family of car vehicle models. These models serve as the foundational building blocks for designing trajectory generation strategies and evaluating their effectiveness through simulation. In order to do that, it is crucial to find the right balance between mathematical complexity and the model's ability to capture dynamic effects.

Second, we are interested in enhancing safety and passenger comfort by generating reference trajectories that can be used as reference for low-level controllers. We propose trajectory generation strategies based on nonlinear optimal control techniques. These strategies are designed to compute collision-free maneuvers, which are critical for the safe operation of autonomous vehicles, especially when encountering dynamic obstacles such as human-driven vehicles or pedestrians. These trajectory generation techniques are useful when navigating through complex scenarios like busy intersections or overtaking slower-moving vehicles. It is worth noticing that our strategies incorporate vehicle dynamics and appropriate constraints, ensuring that the generated trajectories are not only safe but also dynamically feasible.

Third, we are interested in improving human-level reliability in autonomous vehicles. To accomplish this, we combine the advantages of model-based optimal control with model-free reinforcement learning. This integration allows us to develop a decision-making and planning schema that combine predictability of deterministic modeling with the flexibility and adaptability of learning-based techniques. This integration empowers autonomous vehicles to make informed decisions, thus increasing their ability to navigate in dynamic environments.

Contributions

The main contributions of this dissertation are as follows.

Trajectory optimization strategy for merging maneuvers. We develop an optimization-based trajectory generation strategy, based on nonlinear optimal control technique introduced in [2], to compute collision-free merging maneuvers. Based on the idea detailed in [3], we introduce the

concept of *virtual target vehicle*, which is constrained to move along the target lane into which the autonomous vehicle intends to merge. Our strategy exploits the extra degree of freedom of the virtual target vehicle to generate a time parametrized reference. This reference trajectory helps to find the optimal space-time gap necessary for a safe merging maneuver. Moreover, based on the obstacles' predictions, we enforce suitable kinematic coordinates constraints to take into account dynamic obstacles. To show the effectiveness of the proposed method, we present a set of numerical computations and highlighting the main features of the generated trajectories.

Real-time collision-free maneuver generation As main contribution of this dissertation, we develop an optimal control based strategy for computing collision-free maneuvers. Based on the idea in develop in [4], our approach involves rewriting the system dynamics in terms of longitudinal and transverse coordinates, with the longitudinal coordinate use as the independent variable, instead of the time. Then, we embed the time variable into the problem formulation and introduce a novel collision avoidance constraint within this new set of variables. The main advantages with respect to the original problem are: we avoid the need for approximations for describing road geometry and we handle both static and dynamic obstacles. The proposed strategy is based on the constrained Projection Operator Newton method for Trajectory Optimization, [1]. The main idea is the following. We start with a feasible vehicle maneuver that satisfy the vehicle's dynamics and state/input constrains. These constraints are handled through a barrier function approach. Then, we iteratively apply the Projection Operator Newton method to solve the relaxed optimal control problem. In each iteration, we adjust the barrier function parameter, effectively increasing the barrier's influence and pushing the optimal maneuvers towards the constraint boundaries. One notable advantage of

our approach is that it leads to convergence towards optimal maneuvers in an interior point fashion. This means that even in scenarios where computational resources are limited, our algorithm can reliably produce sub-optimal (intermediate) solutions. To validate the effectiveness of our algorithm, we provide numerical simulations that demonstrate its performance in various scenarios. Furthermore, we integrate our algorithm into an autonomous driving stack developed by Ambarella[1] to highlighting its practical applicability in real-world scenarios.

Combining policy search and model predictive control for lane change maneuvers. We design a strategy for lane change maneuvers based on model predictive control combined with reinforcement learning policy search. In contrast to existing methodology, which often decouple decision-making and planning tasks, leading to performance bottlenecks and conservative solutions, our approach takes a hybrid perspective. Specifically, we address the challenge of capturing the lane change decision through an upper-level policy search. This upper-level policy then guides the low-level policy of the model predictive control. Our proposed method employs a weighted maximum likelihood approach for policy learning, thereby effectively optimizing the lane change strategy. Furthermore, we integrate self-supervised learning techniques to adapt to dynamic, online scenarios, enhancing the autonomous vehicle's capacity to respond effectively to unexpected changes in its environment

[1] https://www.ambarella.com/applications/automotive/

Outline

The dissertation is structured as follows.

In Chapter 1 we provide a family of car-like vehicle models. First, we introduce the well-known bicycle model, which serves as fundamental framework for predicting the vehicle's motion for trajectory planning strategies. Then, we re-write the model using curvilinear coordinates, which are particularly useful when dealing with curved or complex trajectories. Second, we develop a dynamic bicycle model, which includes tire modeling. This model provides a more realistic representation of vehicle behavior during simulation, accounting for tire forces and their influence on vehicle dynamics.

In Chapter 2 we address the merging problem and present an optimization based strategy for generating optimal collision-free trajectory. The proposed strategy leverages the extra degree of freedom of a *virtual target vehicle* to generate a time-parametrized reference, which helps to find the right space-time gap to perform a safe merging maneuver. We show the efficacy of the proposed strategy through a set of numerical computations and highlighting the main features of the generated trajectories.

In Chapter 3 we develop a real-time collision-free maneuver generation algorithm. Based on longitudinal and transverse coordinates, we propose a novel collision avoidance constraint and formulate a suitable maneuver regulation optimal control problem. The optimization problem is solved by using a nonlinear optimal control technique that generates (local) optimal trajectories. We validate the proposed algorithm through experimental results, demonstrating its efficiency in terms of computational effort and its ability to capture dynamic features.

In Chapter 4 we address the lane change problem and introduces a parametric model predictive control formulation for decision-making and generating these maneuvers. This formulation combines upper-level policy search with model predictive control policies to generate collision-free lane change maneuvers. We illustrate the efficiency of this approach and highlight important characteristics of the executed maneuvers.

In the appendices, we provide supplementary material to enhance the understanding of the content presented in this work.

- *Appendix A.* We recall the projection operator-based Newton method, which is the fundamental framework for the strategies developed in Chapter 3.

- *Appendix B.* We delve into the direct multiple shooting method, which is a tool for solving optimal control problems. This method plays a crucial role in the generation of optimal maneuvers as discussed in Chapter 2 and is also essential for solving parametric model predictive control in Chapter 4.

Symbols

The following symbols are consistently used throughout this dissertation. A rigorous definition of the symbols will be given in the following chapters.

x, y $[m]$	vehicle longitudinal, later position point rear wheel
ψ $[rad]$	vehicle heading
$\dot{\psi}$ $[rad/s]$	vehicle yaw rate
$\bar{x}_{cl}(s), \bar{y}_{cl}(s)$ $[m]$	archlength parametrized centre-line
$\bar{\kappa}_{cl}$ $[1/m]$	centre-line curvature
$\bar{\psi}_{cl}$ $[rad]$	centre-line heading
s, w $[m]$	centre-line longitudinal, transverse coordinates
μ $[rad]$	relative course heading , $\mu = \psi - \bar{\psi}_{cl}$
$\bar{x}_{tl}(s_{tl}), \bar{y}_{tl}(s_{tl})$ $[m]$	archlength parametrized virtual target vehicle
s_{tl} $[m]$	virtual target vehicle longitudinal coordinate
$\bar{\kappa}_{tl}$ $[1/m]$	virtual target vehicle curvature
$\bar{\psi}_{tl}$ $[rad]$	virtual target vehicle heading
e_x, e_y $[m]$	virtual target vehicle longitudinal, lateral error coordinates
e_ψ $[rad]$	virtual target vehicle relative heading , $e_\psi = \psi - \bar{\psi}_{tl}$

r $[m]$	instantaneous center of rotation radius
L $[m]$	vehicle wheelbased
δ $[rad]$	front wheel steer angle
v $[m/s]$	vehicle velocity point rear wheel
κ $[1/m]$	vehicle curvature point rear wheel
x_{GoG}, y_{CoG} $[m]$	vehicle logitudinal, lateral position Center of Gravity
$\dot{x}_{CoG}, \dot{y}_{CoG}$ $[m/s]$	vehicle longitudinal, lateral velocity Center of Gravity
$F_x^{f,r}, F_y^{f,r}, F_z^{f,r}$ $[N]$	longitudinal, lateral ,and normal tire forces Center of Gravity
I_z, I_w $[kg\ m^2]$	wheel inertia (w.r.t z-axis)
r_w $[m]$	vehicle wheels radius
$\omega^{f,r}$ $[rad/s]$	front, rear wheel angular velocity
$T^{f,r}$ $[N\ m]$	front, rear wheel torque
$F_l^{f,r}, F_c^{f,r}$ $[N]$	front, rear wheel tire force
a, b $[m]$	distance bettween front wheel and Center of Gravity
$\beta^{f,r}$ $[rad]$	distance bettween rear wheel and Center of Gravity
$v_c^{f,r}, v_l^{f,r}$ $[m/s]$	lonfitudinal, laterl wheels velocity
$v_x^{f,r}, v_y^{f,r}$ $[m/s]$	longitudinal, lateral components wheels velocity
g $[m/s^2]$	gravity acceleration
a $[m/s^2]$	vehicle longitudinal acceleration point rear wheel
$\dot{\kappa}$ $[1/m\ s]$	vehicle curvature rate point rear wheel
v_{vtv} $[m/s]$	virtual target vehicle velocity

Chapter 1

Vehicle Models

In this chapter, we present a family of car-like vehicle models that we will use throughout this dissertation. These models play a crucial role in generating feasible trajectories and validating the proposed approaches through simulation. We start introducing the well-known bicycle model, which provides a simplified representation of a vehicle's motion. Then, we develop a dynamic bicycle model designed specifically for simulation purposes.

1.1 Introduction

Accurate vehicle modeling plays a crucial role in various applications such as trajectory generation [5, 6, 7], real-time model predictive control [8, 9, 10], and simulation [11]. The level of detail in a model requires a

trade-off between the dynamics features one wants to capture and the computational efficiency required for practical implementation.

In the literature, various models with different levels of complexity have been introduced to capture the vehicle's behavior [12, 13, 14]. Typically, there are two approaches to modeling vehicle motion: kinematic modeling and dynamic modeling.

Kinematic modeling focuses on the geometric constraints that define the vehicle's motion, making it particularly suitable for capturing motion at low velocities where accelerations are negligible. On the other hand, dynamic modeling takes into account the forces and moments acting on the vehicle, offering more accurate estimation of vehicle motion throughout its operating range. However, employing highly complex vehicle models often results in intensive computation, leading to high computation times.

The next part of the chapter is dedicated to the development of two reduced-order car vehicle models, namely the *kinematic bicycle* model and the *dynamic bicycle* model. The kinematic bicycle model will be used to generate feasible trajectories in Chapters 2, 3 and 4, while the dynamic bicycle model will be used to validate the proposed maneuver generation approach through simulation in Chapter 3.

In Section 1.2, we introduce different types of coordinate frames used throughout this dissertation. In Section 1.3, we present the kinematic bicycle model and then we rewrite the vehicle model with respect to the newly introduced coordinate frames. In Section 1.4, we develop a dynamic bicycle model.

1.2 Coordinate Systems

Before we present the vehicle models, it is important to introduce the different coordinate frames used throughout this dissertation.

The first coordinate frame we consider is the body frame, which is positioned at a key location on the vehicle, such as the center of gravity (CoG) or the center point of the rear axle. As the vehicle moves, the body frame undergoes both translational and rotational motion with respect to a fixed inertial frame.

The second coordinate frame we define is the inertial or global coordinate frame. We introduce three different inertial coordinate frames.

First, we introduce the Cartesian frame, which is a fixed reference frame attached to the Earth. Common representations include the East North Up (ENU) or North East Down (NED) frame, with respect to a nearby reference point. In order to provide a visual representation of these concepts, Figure 1.1 illustrates a Cartesian coordinate system with the body frame attached to the vehicle.

Next, we introduce alternative inertial frame coordinates. Especially for trajectory generation, it is often more convenient and practical to describe the motion of the vehicle using curvilinear coordinate systems instead of traditional Cartesian coordinates [15]. Curvilinear coordinate systems are particularly useful when dealing with motion along a curved path, as they align with the natural geometry of the motion.

Specifically, we focus on the Serret-Frenet coordinate system [16, 17], which is a widely used framework for describing motion along a path. The Serret-Frenet coordinate system provides a local, tangent-based representation of the motion, allowing to capture the vehicle's position, orientation, and curvature with respect to a desired path. One significant advantage of using curvilinear coordinate systems, such as the Serret-

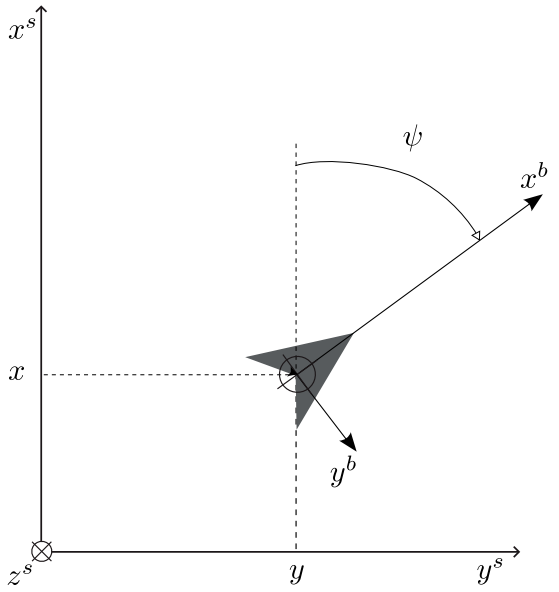


Figure 1.1: Inertial and vehicle body frames. The inertial frame, represented in North East Down (NED) convention, serves as a fixed reference frame relative to the Earth. The vehicle body frame is attached to the vehicle represented by the bold triangle.

Frenet coordinate system, is that they naturally handle the vehicle's motion along curved paths, enabling simplified descriptions of vehicle dynamics. Taking into account this new frame, we can transform complex motion equations into simpler, more intuitive forms, which facilitate the development of the proposed trajectory generation approaches.

Furthermore, we introduce the concept of the Virtual Target Vehicle (VTV), a powerful tool used to address certain constraints and singularities that may arise in trajectory planning and control. The VTV coordinate system allows us to treat the vehicle as if it is following a virtual target point with specific motion characteristics, which provides

more flexibility in generating feasible trajectories.

1.2.1 Longitudinal and Transverse Coordinates

As mentioned above, it is convenient to introduce a new coordinate system, namely the Serret-Frenet frame, to describe the vehicle's position with respect to a given curve, usually the lane center-line. This setup is particularly well-suited for urban road environments, allowing for an opportune representation of the vehicle's dynamics in relation to the road geometry instead of using Cartesian coordinates.

As shown in Figure 1.2, the longitudinal coordinate s represents the position along the center-line, while the lateral coordinate w denotes the displacement transverse to the center-line. The angle μ is the vehicle's heading with respect to the center-line direction. It is defined positively clockwise with $\mu = 0$ when the x-axis is aligned with the tangent of the center-lane.

Given the road geometry, we assume that the lane has a reasonably smooth (at least C^2) arc-length parametrized center-line, $(\bar{x}_{cl}(s), \bar{y}_{cl}(s))$. The course heading $\bar{\psi}_{cl}(s)$ and the curvature $\bar{\kappa}_{cl}(s)$ are related by differentiation:

$$\begin{aligned} \frac{d\bar{x}_{cl}(s)}{ds} &= \bar{x}'_{cl}(s) = \cos\bar{\psi}_{cl}(s)\,, \\ \frac{d\bar{y}_{cl}(s)}{ds} &= \bar{y}'_{cl}(s) = \sin\bar{\psi}_{cl}(s)\,, \\ \frac{d\bar{\psi}_{cl}(s)}{ds} &= \bar{\psi}'_{cl}(s) = \bar{\kappa}_{cl}(s)\,, \end{aligned} \tag{1.1}$$

where the bar symbol indicates that the variable is expressed as a function of the longitudinal coordinate s, and the prime symbol denotes differentiation with respect to s. Using the longitudinal coordinates s and the lateral displacement w, each point in the environment is then of the form:

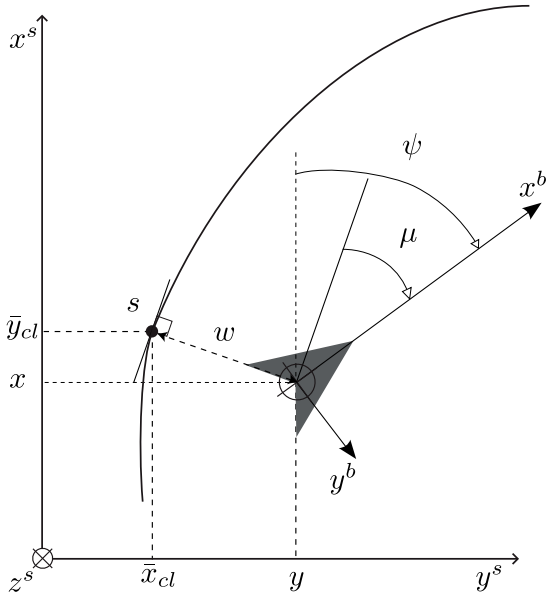


Figure 1.2: Local coordinates around the geometry path. The bold triangle represents the ego-vehicle, while the solid line denotes the center-line of the lane.

$$\begin{aligned}\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} &= \begin{bmatrix} \bar{x}_{cl}(s) \\ \bar{y}_{cl}(s) \end{bmatrix} + \begin{bmatrix} -\sin\bar{\psi}_{cl}(s) \\ \cos\bar{\psi}_{cl}(s) \end{bmatrix} w(t) \\ &= \begin{bmatrix} \bar{x}_{cl}(s) \\ \bar{y}_{cl}(s) \end{bmatrix} + R_z(\bar{\psi}_{cl}(s)) \begin{bmatrix} 0 \\ w(t) \end{bmatrix} \end{aligned}, \tag{1.2}$$

where

$$R_z(\bar{\psi}_{cl}(s)) = \begin{bmatrix} \cos\bar{\psi}_{cl}(s) & -\sin\bar{\psi}_{cl}(s) \\ \sin\bar{\psi}_{cl}(s) & \cos\bar{\psi}_{cl}(s) \end{bmatrix}$$

is the rotation matrix transforming vectors from the velocity frame into the inertial frame.

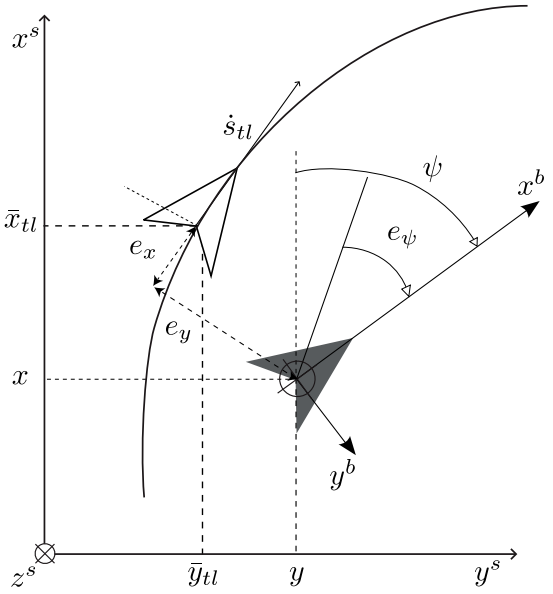


Figure 1.3: Local coordinates around the geometry path. The bold and the empty triangles indicate the ego-vehicle and the VTV, respectively. The solid line indicates the center-line of the target lane.

Equation (1.2) is used to re-write the vehicle's kinematic using the new coordinates frame in Section 1.3.1.

1.2.2 Virtual Target Vehicle

The Serret-Frenet coordinate system, as previously described, is attached to the point on the path closest to the vehicle. In certain scenarios, it can be convenient to relax this assumption by introducing the concept of VTV. In such a case, the Serret-Frenet coordinate system, which can be viewed as the body frame of a "virtual vehicle", moves along the desired path according to a desired velocity, see Figure 1.3.

Following the idea described in [3] for unmanned aerial vehicles, we assume that the desired target reference path has a smooth arc-length parametrized center-line, $(\bar{x}_{tl}(s_{tl}), \bar{y}_{tl}(s_{tl}))$ and we constrain the VTV to move along the center-line of this target lane, see Figure 1.3, so that the VTV's position can be described by simply integrating its velocity v_{vtv}, i.e., $\dot{s}_{tl} = v_{vtv}$. Similarly to (1.1), we can relate the heading of the target center-line, i.e., $\bar{\psi}_{tl}(s_{tl})$ and the curvature $\bar{\kappa}_{tl}(s_{tl})$, by the following differential equations:

$$\begin{aligned} \bar{x}'_{tl}(s_{tl}) &= \cos\bar{\psi}_{tl}(s_{tl})\,, \\ \bar{y}'_{tl}(s_{tl}) &= \sin\bar{\psi}_{tl}(s_{tl})\,, \\ \bar{\psi}'_{tl}(s_{tl}) &= \bar{\kappa}_{tl}(s_{tl})\,. \end{aligned}$$

Given the VTV's position along the target lane, s_{tl}, we can now relate Cartesian and VTV coordinates as follows,

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \bar{x}_{tl}(s_{tl}) \\ \bar{y}_{tl}(s_{tl}) \end{bmatrix} + R_z(\bar{\psi}_{tl}(s_{tl})) \begin{bmatrix} e_x(t) \\ e_y(t) \end{bmatrix}, \tag{1.3}$$

where e_x and e_y are the longitudinal and lateral error coordinates, respectively, and $R_z(\bar{\psi}_{tl})$,

$$R_z(\bar{\psi}_{tl}(s_{tl})) = \begin{bmatrix} \cos\bar{\psi}_{tl}(s_{tl}) & -\sin\bar{\psi}_{tl}(s_{tl}) \\ \sin\bar{\psi}_{tl}(s_{tl}) & \cos\bar{\psi}_{tl}(s_{tl}) \end{bmatrix}$$

is the rotation matrix transforming vectors from the error frame into the inertial frame. We use (1.3) in Section 1.3.2 to derive the vehicle's motion with respect to the VTV.

1.3 Kinematic Bicycle Model

The well-known kinematic bicycle model has long been used as a suitable control-oriented model for representing vehicles because of its simplicity,

see, e.g., [18, 19]. Such a model mimics well the vehicle dynamics under mild driving conditions where wheel slip can be neglected and without considering the forces that affect the vehicle. Indeed, the equations of motion are derived from the geometric relationships that govern the system.

Next we describe the vehicle's motion using Cartesian coordinate introduced in Section 1.2. Let us consider a bicycle model of a car-like vehicle as shown in Figure 1.4. We assume the vehicle operates on a 2D horizontal plane. In the proposed model, the front wheel at point A represents the front right and left wheels of the vehicle. Similarly, the rear wheel at point B represents the rear right and left wheels. Such a model is called front-wheel-only steering vehicle, as the front wheel orientation

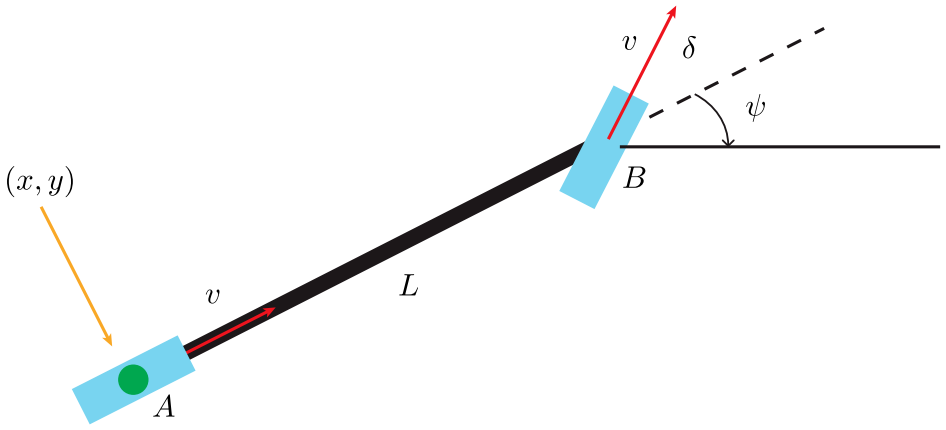


Figure 1.4: Kinematic bicycle model. The vehicle is represented as a simplified bicycle model with two wheels. The reference point, A, is located on the rear axle. The variables (x, y) and ψ denote the position and orientation of the vehicle, while δ and v represent the steering angle and velocity, respectively. The wheelbase is denoted as L.

can be controlled relative to the vehicle's heading ψ. In order to derive the equations of motion, we need to define some additional variables. Let us define δ as the steering angle for the front wheel, measured relative to the forward bicycle's direction. The wheelbase of the vehicle, namely the distance between the two wheels, is denoted by L. Moreover, in order to analyze the kinematics of the bicycle model, we need to select a reference point, (x, y), on the vehicle. This point can be located at the center of gravity (CoG), at the center of the rear axle, or at the center of the front axle. Such a choice affect the kinematic equations. We locate the reference point on the rear wheel for convenience. The velocity is denoted with v.

Now we are ready to derive the kinematic model starting form the following observation: the vehicle is constraint to move forward because its wheels point in this direction. This constraint, namely the non-holonomic constraint, restricts the rate of change of the vehicle's position. Specifically, the non-holonomic constraint impose that the vehicle cannot move sideways without performing a turning maneuver. We use this constraint to derive the vehicle's equation of motion.

First, we write expression for the nonholonomic constraint equation as follows,

$$\dot{x}(t)\sin\psi(t) - \dot{y}(t)\cos\psi(t) = 0\,. \tag{1.4}$$

In the following equations, we will omit the explicit time dependency when it is evident from the context that the variable is time-dependent.

Second, by rearranging (1.4) we can construct a the system of equations for the vehicle's motion:

$$\begin{aligned} \dot{x} &= v\cos\psi\,, \\ \dot{y} &= v\sin\psi\,. \end{aligned} \tag{1.5}$$

Third, we need to relate the rate of change of the heading ψ and the steering angle δ. The main assumption is that the velocity at A and B

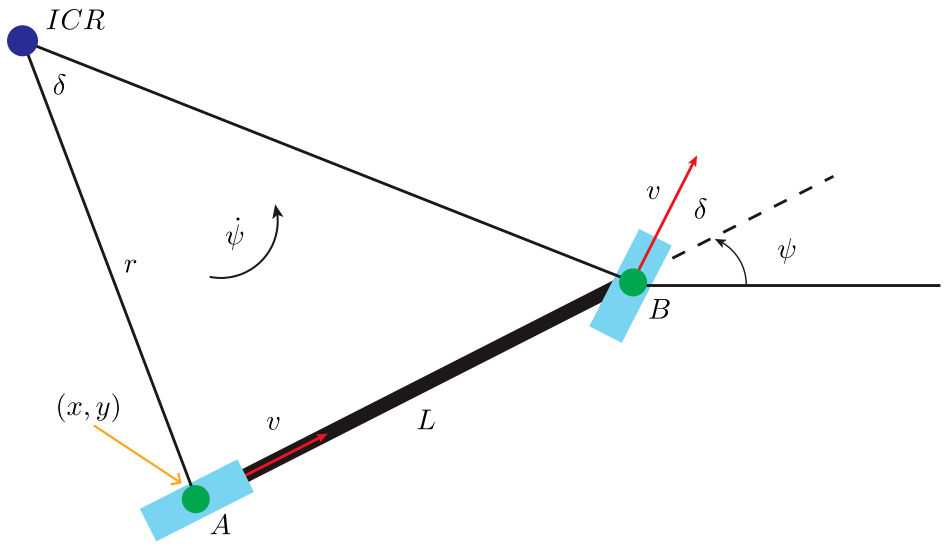


Figure 1.5: Instantaneous Center of Rotation (ICR). The ICR is the point around which the vehicle's motion can be approximated as pure rotation at a specific moment in time. The ICR is denoted by the blue dot, and its location changes as the vehicle moves and steers.

aligns with the orientation of the front and rear wheels, respectively. This is equivalent to assume that the slip angles at both the front and rear wheels are zero. This is a reasonable assumption at low velocities, where the later forces generated by the tires are small. Enforcing the no-slip condition implies that the vehicle can moves only in the direction normal to the axis of the driving wheel. In other words, the vehicle satisfies the pure rolling and no-slip condition. By considering this assumption, we can establish a relationship between the rotational rate of the bicycle $\dot{\psi}$ and the velocity v. Specifically, $\dot{\psi}$ is equal to the velocity over by the instantaneous center of rotation ICR, which has a radius r:

$$\dot{\psi} = \frac{v}{r}\,. \tag{1.6}$$

As depicted in Figure.1.5, the similar triangles formed by L and r, and v and δ, result in the relationship:

$$\tan\delta = \frac{L}{r}\,. \tag{1.7}$$

By combining (1.6) and (1.7), we derive the relation that holds between the rotation rate of the vehicle, $\dot{\psi}$, and the steering angle, δ, as follows:

$$\dot{\psi} = \frac{v}{r} = \frac{v\tan\delta}{L}\,. \tag{1.8}$$

Finally, we can now write kinematic bicycle model for the rear axle reference point, using (1.5) and (1.8), as follows:

$$\begin{aligned} \dot{x} &= v\cos\psi\,, \\ \dot{y} &= v\sin\psi\,, \\ \dot{\psi} &= \frac{v\tan\delta}{L}\,. \end{aligned} \tag{1.9}$$

In order to reduce nonlinear terms in (1.9), it is convenient to use the curvature κ instead of the steering angle δ. By knowing that

$$\kappa = \frac{1}{r} = \frac{\tan\delta}{L}\,,$$

we can rewrite the previous (1.9) as follows:

$$\begin{aligned} \dot{x} &= v\cos\psi\,, \\ \dot{y} &= v\sin\psi\,, \\ \dot{\psi} &= v\kappa\,. \end{aligned} \tag{1.10}$$

In this formulation, the state of the system, $\mathbf{x} = [x, y, \psi]$, includes the positions x, y, and the orientation ψ, while the input vector, $\mathbf{u} = [\kappa, v]$, includes the curvature κ and the velocity v. These equations satisfy the pure rolling and no-slip conditions, which are fundamental assumptions in deriving the proposed vehicle model.

1.3.1 Longitudinal and Transverse Coordinate Formulation

Next, we rewrite the kinematic bicycle model (1.10) with respect to longitudinal and transverse coordinates (s, w).

We know that each point in the environment can be expressed as follows, see Section 1.2.1:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \bar{x}_{cl}(s) \\ \bar{y}_{cl}(s) \end{bmatrix} + R_z(\bar{\psi}_{cl}(s)) \begin{bmatrix} 0 \\ w \end{bmatrix} . \tag{1.11}$$

First, following the calculation in [15], we differentiate (1.11) and we get:

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} &= R_z(\bar{\psi}(s)) \begin{bmatrix} \dot{s} \\ 0 \end{bmatrix} + R_z(\bar{\psi}(s)) \begin{bmatrix} 0 \\ \dot{w} \end{bmatrix} + R_z'(\bar{\psi}(s)\bar{\kappa}(s)\dot{s} \begin{bmatrix} 0 \\ w \end{bmatrix} \\ &= R_z(\bar{\psi}(s)) \begin{bmatrix} \dot{s} \\ 0 \end{bmatrix} + R_z(\bar{\psi}(s)) \begin{bmatrix} 0 \\ \dot{w} \end{bmatrix} \\ &+ \begin{bmatrix} -\sin\bar{\psi}(s) & -\cos\bar{\psi}(s) \\ \cos\bar{\psi}(s) & -\sin\bar{\psi}(s) \end{bmatrix} \bar{\kappa}_{cl}(s)\dot{s} \begin{bmatrix} 0 \\ w \end{bmatrix} \\ &= R_z(\bar{\psi}(s)) \begin{bmatrix} \dot{s} \\ 0 \end{bmatrix} + R_z(\bar{\psi}(s)) \begin{bmatrix} 0 \\ \dot{w} \end{bmatrix} - R_z(\bar{\psi}(s))\bar{\kappa}_{cl}(s)\dot{s} \begin{bmatrix} 0 \\ w \end{bmatrix} . \end{aligned} \tag{1.12}$$

Equation (1.12) can be rewritten in compact form as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = R_z(\bar{\psi}(s)) \begin{bmatrix} (1 - w\bar{\kappa}_{cl}(s))\dot{s} \\ \dot{w} \end{bmatrix} . \tag{1.13}$$

Second, by using (1.10) and (1.13) , we obtain:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = R_z(\bar{\psi}(s)) \begin{bmatrix} (1 - w\bar{\kappa}_{cl}(s))\dot{s} \\ \dot{w} \end{bmatrix} = R_z(\psi) \begin{bmatrix} v \\ 0 \end{bmatrix} ,$$

from which we find that:

$$\begin{bmatrix} (1 - w\bar{\kappa}_{cl}(s))\dot{s} \\ \dot{w} \end{bmatrix} = R_z(\psi - \bar{\psi}(s)) \begin{bmatrix} v \\ 0 \end{bmatrix} . \tag{1.14}$$

It is important to highlight that (1.14) expresses the rotational invariance of the planar kinematics (a result well-known in differential geometry): only the relative heading $\psi - \bar{\psi}(s)$ and the center-lane curvature $\bar{k}_{cl}(s)$ are needed to fully capture and describe the vehicle's motion using longitudinal and transverse coordinate system. This rotational invariance property simplifies the representation and calculation of the vehicle's dynamics, making it more efficient and easier to work with in various trajectory planning and control tasks.

Finally, we describe the ego-vehicle position with respect to the (s, w) coordinates.

$$\begin{aligned} \dot{s} &= \frac{v \cos \mu}{1 - w\bar{\kappa}_{cl}(s)} , \\ \dot{w} &= v \sin \mu , \\ \dot{\mu} &= v\kappa - \bar{\kappa}_{cl}(s)\dot{s} , \end{aligned} \tag{1.15}$$

where $\mu = \psi - \bar{\psi}_{cl}(\mathrm{s})$ is the local heading error.

It is worth noting that the inverse of the map $(s, w) \mapsto (x, y)$ is well-defined only if the following condition is satisfied:

$$1 - w\bar{\kappa}_{cl}(s) > 0 ,$$

which means that the ego-vehicle position must lie inside a tube around the center-line of the lane. This condition ensures the validity of the mapping between road-compliant coordinates and Cartesian coordinates and is essential to guarantee that the transformation is meaningful and accurate for the vehicle's position within the lane.

1.3.2 Virtual Target Vehicle Formulation

In this section, we rewrite the vehicle kinematics (1.10) using the VTV formulation introduced in Section 1.2.2.

We know that each point in the environment can be expressed as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \bar{x}_{tl}(s_{tl}) \\ \bar{y}_{tl}(s_{tl}) \end{bmatrix} + R_z(\bar{\psi}_{tl}(s_{tl})) \begin{bmatrix} e_x \\ e_y \end{bmatrix} . \tag{1.16}$$

First, we differentiate (1.16) with respect to the time t and we obtain

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \bar{x}_{tl} \\ \bar{x}_{tl} \end{bmatrix} \dot{s}_{tl} + \begin{bmatrix} -\sin\bar{\psi}_{tl} & -\cos\bar{\psi}_{tl} \\ \cos\bar{\psi}_{tl} & -\sin\bar{\psi}_{tl} \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix} \bar{\kappa}_{tl}(s_{tl})\dot{s}_{tl} + R_z(\bar{\psi}_{tl}) \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \end{bmatrix} .$$

By using the kinematic of the vehicle (1.10) and VTV local coordinates (1.16), we can re-arrange the previous equation as follows:

$$R_z(\psi_{tl}) \begin{bmatrix} v \\ 0 \end{bmatrix} = R_z(\bar{\psi}_{tl}) \begin{bmatrix} \dot{s}_{tl} \\ 0 \end{bmatrix} - R_z(\bar{\psi}_{tl}) \begin{bmatrix} e_y \\ -e_x \end{bmatrix} \bar{\kappa}_{tl}(s_{tl})\dot{s}_{tl} + R_z(\bar{\psi}_{tl}) \begin{bmatrix} \dot{e}_x \\ -\dot{e}_y \end{bmatrix} ,$$

that is

$$\begin{bmatrix} (1 - e_y\bar{\kappa}_{tl}(s_{tl}))\dot{s}_{tl} + \dot{e}_x \\ e_x\bar{\kappa}_{tl}\dot{s}_{tl} + \dot{e}_y \end{bmatrix} = R_z(\bar{\psi}_{tl})^T R_z(\psi) \begin{bmatrix} v \\ 0 \end{bmatrix} .$$

Defining the local heading error as $e_\psi = \psi - \bar{\psi}_{tl}$, now it is straightforward to compute the expressions for $\dot{e}_x$ and $\dot{e}_y$:

$$\begin{aligned} \dot{e}_x &= v\cos e_\psi - (1 - e_y\bar{\kappa}_{tl}(s_{tl}))v_{vtv} , \\ \dot{e}_y &= v\sin e_\psi - e_x\bar{\kappa}_{tl}(s_{tl})\dot{s}_{tl} . \end{aligned}$$

Finally, we can describe the ego vehicle dynamics (1.10) as follows

$$\begin{aligned} \dot{s}_{tl} &= v_{vtv} , \\ \dot{e}_x &= v\cos e_\psi - (1 - e_y\bar{\kappa}_{tl}(s_{tl}))v_{vtv} , \\ \dot{e}_y &= v\sin e_\psi - e_x\bar{\kappa}_{tl}(s_{tl})v_{vtv} , \\ \dot{e}_\psi &= v\kappa - \bar{\kappa}_{tl}(s_{tl})v_{vtv} , \end{aligned}$$

where $\mathbf{x} = [s_{tl}, e_x, e_y, e_\psi]$ and $\mathbf{u} = [\kappa, v, v_{vtv}]$ represent the state and control vectors, respectively.

It is important to highlight that in (1.15), the longitudinal error between the vehicle and the Serret-Frenet frame is equal to 0 for all t because the point on the desired path is defined by the projection of the actual vehicle on the path. As a result, the velocity of the VTV is given by

$$\dot{s} = \frac{v \cos \mu}{1 - w\bar{\kappa}_{cl}(s)}.$$

However, a singularity appears when $w = \frac{1}{\bar{\kappa}_{cl}}$. This singularity imposes a constraint on the position of the actual vehicle to be inside a "tube" around the desired path. Such a constraint can be overly conservative and may restrict the exploration of trajectories for the vehicle. To overcome this limitation, the VTV formulation can be employed. By considering the velocity of the virtual target v_{vtv} as a new control input, the VTV approach effectively eliminates the singularity and enables more flexibility in trajectory planning and exploration. This allows the vehicle to navigate a wider range of paths, enhancing its maneuverability and path-following capabilities.

1.3.3 Spatial Formulation

In this section, we rewrite the vehicle's kinematics using the longitudinal coordinate s as the independent variable.

First, we recall that the vehicle kinematics with respectn to transverse and longitudinal coordinates has the following expression, see Sec-

tion 1.3.1,

$$\begin{aligned} \dot{s} &= \frac{v\cos\mu}{1-w\bar{\kappa}_{cl}(s)}\,, \\ \dot{w} &= v\sin\mu\,, \\ \dot{\mu} &= v\kappa - \bar{\kappa}_{cl}(s)\dot{s}\,. \end{aligned} \tag{1.17}$$

Then, let us denote with $\bar{t}(s) : \mathbb{R}_0^+ \longmapsto \mathbb{R}_0^+$ the inverse of $s(t) : \mathbb{R}_0^+ \longmapsto \mathbb{R}_0^+$, which satisfies $\bar{t}(s(t)) = t$. Given that $s(t)$ is invertible, any function of time $\alpha(t)$ can be expressed as a function of s as $\alpha(\bar{t}(s))$. By defining $\bar{\alpha} = \alpha(\bar{t})$, we have $\alpha(t) = \bar{\alpha}(s(t))$ and thus, in our case, it holds the following equivalences:

$$\begin{aligned} t &= \bar{t}(s), \quad w = \bar{w}(s), \quad \mu = \bar{\mu}(s), \\ v &= \bar{v}(s), \quad k = \bar{k}(s), \quad a = \bar{a}(s)\,. \end{aligned} \tag{1.18}$$

By differentiating with respect to time (1.18), we see that

$$\begin{aligned} t &= \bar{t}'(s), \quad w = \bar{w}'(s)\dot{s}, \quad w = \bar{w}'(s)\dot{s} \\ v &= \bar{v}'(s)\dot{s}, \quad k = \bar{k}'(s)\dot{s}, \quad a = \bar{a}'(s)\dot{s}\,. \end{aligned}$$

Thus following that (1.17) became

$$\begin{aligned} \bar{w}'(s) &= \frac{v\sin\mu}{\dot{s}}\,, \\ \bar{\mu}'(s) &= \frac{v\kappa}{\dot{s}} - \bar{\kappa}_{cl}(s)\,, \end{aligned}$$

so that now variables depend on time only through $s(t)$. In this way, we can obtain a description of the vehicle kinematics as a function of the longitudinal coordinates s.

Formally, we obtain the spatial vehicle kinematics in terms of longi-

tudinal coordinate s, as follows:

$$\begin{aligned} \bar{w}'(s) &= (1 - \bar{\kappa}_{cl}(s)\bar{w}(s)) \tan \bar{\mu}(s)\,, \\ \bar{\mu}'(s) &= \frac{1 - \bar{\kappa}_{cl}(s)\bar{w}(s)}{\cos \bar{\mu}(s)} \bar{\kappa}(s) - \bar{\kappa}_{cl}(s)\,, \\ \bar{v}'(s) &= \frac{1 - \bar{\kappa}_{cl}(s)\bar{w}(s)}{\bar{v}(s) \cos \bar{\mu}(s)} \bar{a}(s)\,, \end{aligned} \tag{1.19}$$

which requires that the vehicle moves with $\bar{v} > 0$ and $|\bar{\mu}| < \pi/2$. In (1.19) the state and the control vector are $\bar{\mathbf{x}}$=$[\bar{w},\ \bar{\mu}]$ and $\bar{\mathbf{u}}$=$[\bar{\kappa},\ \bar{v}]$ respectively. For the sake of notational simplicity, from now on we will neglect the explicit dependence from s if it is clear from the context.

1.4 Dynamic Bicycle Model

The dynamic bicycle model is a fundamental representation of a vehicle's motion, accounting for both the translational and rotational dynamics. It provides a more detailed description of a vehicle's behavior compared to the kinematic bicycle model, as it considers the forces and moments acting on the vehicle during its motion. As a result, it offers a more accurate description of a vehicle's behavior, making it a crucial tool in various applications, particularly for vehicle simulation.

In the dynamic bicycle model, see Figure 1.6, the vehicle is typically represented as a rigid body with a fixed center of mass. The model considers the longitudinal and lateral forces acting on the vehicle's Center of Gravity (CoG), as well as the rotating moment around the vertical axis. As for the kinematic bicycle model in Section 1.3, we assume a simplified representation where the vehicle's front wheels and the two rear wheels are lumped into a single wheel located at the points A and B, respectively. The variables $(x, y, \psi)_{CoG}$ denote the longitudinal position,

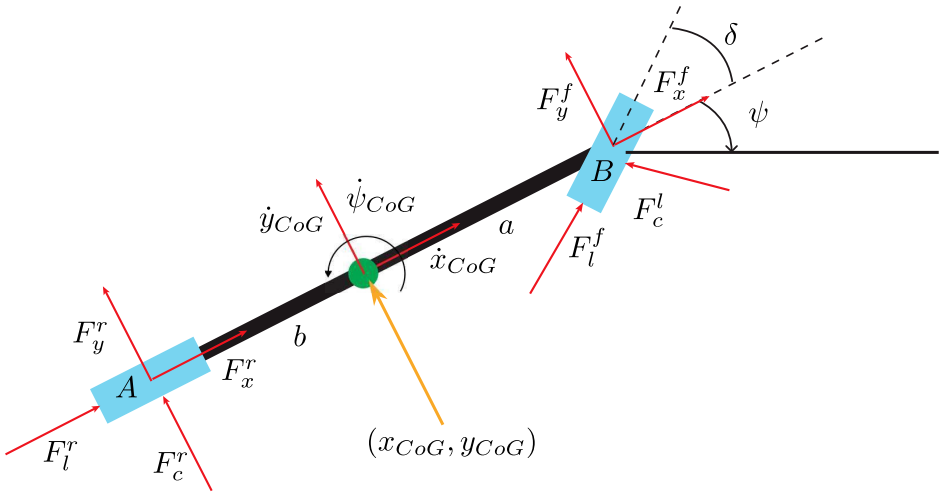


Figure 1.6: Dynamic bicycle model. The vehicle is represented as a simplified bicycle model with two wheels. The variables $(x, y, \psi)_{CoG}$ denote the longitudinal position, lateral position, and heading of the vehicle, while $(\dot{x}, \dot{y}, \dot{\psi})_{CoG}$ represent the longitudinal velocity, lateral velocity, and yaw rate, respectively. $F_l^{f,r}$ and $F_c^{f,r}$ represent the longitudinal and cornering (lateral) tire forces acting on the wheels. The components of these forces along the longitudinal and lateral vehicle's axes are denoted as $F_x^{f,r}$ and $F_y^{f,r}$. The parameter δ denotes the front wheel steering angle, and a and b are the distances from the CoG to the front and rear axles, respectively.

lateral position, and heading of the vehicle, while $(\dot{x}, \dot{y}, \dot{\psi})_{CoG}$ represent the longitudinal velocity, lateral velocity, and yaw rate, respectively. In this model, we consider the forces $F_l^{f,r}$ and $F_c^{f,r}$ that represent the longitudinal and cornering (lateral) tire forces acting on the wheels. The components of these forces along the longitudinal and lateral vehicle's axes are denoted as $F_x^{f,r}$ and $F_y^{f,r}$. The variable δ denotes the front wheel steering angle, and a and b are the distances from the CoG to the front and rear axles, respectively. The vehicle dynamics can be derived by con-

sidering the equation of motion about the CoG and the coordinate transformation between the body frame and the inertial frame. The equations are as follows:

$$\begin{aligned}
\dot{x}_{CoG} &= \dot{x}_{CoG}\cos\psi - \dot{y}_{CoG}\sin\psi\,, \\
\dot{y}_{CoG} &= \dot{x}_{CoG}\sin\psi + \dot{y}_{CoG}\cos\psi\,, \\
\dot{\psi}_{CoG} &= \omega\,, \\
m\ddot{x}_{CoG} &= mv_y\dot{\psi}_{CoG} + 2F_x^f + 2F_x^r\,, \\
m\ddot{y}_{CoG} &= -mv_x\dot{\psi}_{CoG} + 2F_y^f + 2F_y^r\,, \\
I_z\dot{\omega} &= 2aF_y^f - 2bF_y^r\,, \\
I_w\dot{\omega}^f &= T^f + r_wF_x^f\,, \\
I_w\dot{\omega}^r &= T^r + r_wF_x^r\,,
\end{aligned} \tag{1.20}$$

where m and I_z represent the vehicle mass and the moment of inertia about the vertical axis, respectively, while I_w denotes the wheel inertial.The wheel angular velocity is represented by $\omega^{f,r}$, while r_w denotes the wheel radius. T^f and T^r are the tractive and braking torques applied to the front and rear wheels, respectively. Positive torque denotes driving torque, while negative torque represents braking torque.

Under the assumption that only the steering angle at the front wheel can be controlled (i.e., $\delta_f = \delta$ and $\delta_r = 0$), the longitudinal and lateral component of the tire force can be computed as follows:

$$\begin{aligned}
F_x^f &= F_l^f\cos\delta - F_c^f\sin\delta, & F_y^f &= F_l^f\sin\delta + F_c^f\cos\delta\,, \\
F_x^r &= F_l^r, & F_y^r &= F_c^r\,.
\end{aligned}$$

The longitudinal and cornering tire forces $F_l^{f,r}$ and $F_c^{f,r}$ can be effectively modeled using Pacejka's magic formula [20]. This complex, semi-empirical formula is well-suited for capturing the tire behavior across a wide range of operating conditions, including linear and nonlinear ranges

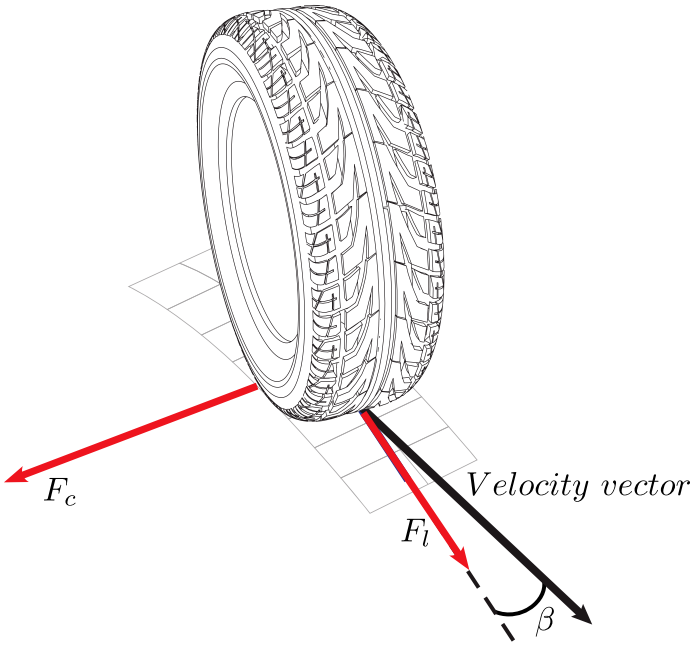


Figure 1.7: Tire model.

of slip ratio $s_r^{f,r}$ and tire slip angle $\beta^{f,r}$. The tire forces can then be expressed as:

$$F_l^{f,r} = f_l(\beta^{f,r}, s_r^{f,r}, F_z^{f,r}),$$
$$F_l^{f,r} = f_c(\beta^{f,r}, s_r^{f,r}, F_z^{f,r}),$$

where $F_z^{f,r}$ represents the normal force acting on the front and rear tires. The functions $f_l(\cdot)$ and $f_l(\cdot)$ in the magic formula incorporate multiple coefficients that are determined through experimental tire testing. As shown in Figure 1.7, the tire slip angles $\beta^{f,r}$ represent the angle between the tire velocity and its longitudinal direction. They can be computed as follows:

$$\beta^{f,r} = \arctan \frac{v_c^{f,r}}{v_l^{f,r}},$$

where $v_l^{f,r}$ and $v_c^{f,r}$ are the longitudinal and cornering wheel velocities, respectively. These velocities are then transformed in the wheel frame as follows:

$$\begin{aligned} v_l^f &= v_y^f \cos\delta + v_x^f \cos\delta, & v_l^r &= v_y^f\,, \\ v_c^f &= v_y^f \cos\delta - v_x^f \sin\delta, & v_c^r &= v_y^f\,, \end{aligned}$$

where $v_x^{f,r}$ and $v_y^{f,r}$ are the longitudinal and lateral components of the wheels velocity, respectively. These velocities can be expressed as a function of the CoG velocity and yaw rate as follows:

$$\begin{aligned} v_x^f &= \dot{x}_{CoG}, & v_x^r &= \dot{x}_{CoG}\,, \\ v_y^f &= \dot{y}_{CoG} + a\dot{\psi}\dot{x}_{CoG}, & v_x^r &= \dot{y}_{CoG} + b\dot{\psi}_{CoG}\,. \end{aligned}$$

The slip ration $s_r^{s,r}$ is a critical parameter that characterizes the interaction between the tires and the road surface. It is defined by

$$s_r^{f,r} = \begin{cases} \frac{r_w\omega^{f,r}}{v_l^{f,r}} - 1, & \text{if } v_{lf,r} > r_w\omega^{f,r},\ \ v_l^{f,r} \neq 0 \text{ for breaking}\,, \\ 1 - \frac{v_l^{f,r}}{r_w\omega^{f,r}}, & \text{if } v_l^{f,r} < r_w\omega^{f,r},\ \ \omega^{f,r} \neq 0 \text{ for throttle} \end{cases},$$

where $\omega^{f,r}$ is the rotational velocity of the wheel and $r^{f,r}$ represent the wheel radius.

Finally, in order to estimate the normal forces acting on the front and real wheels, we use a static weight distribution approach to approximate these forces as follows:

$$F_z^f = \frac{bmg}{(a+b)}, \quad F_z^r = \frac{amg}{(a+b)}\,,$$

where g is the acceleration due to gravity.

In (1.20), $\mathbf{x} = [x_{CoG}, y_{CoG}, \psi_{CoG}, \dot{x}_{CoG}, \dot{y}_{CoG}, \dot{\psi}_{CoG}, \omega^f, \omega^r]^T$ represents the state vector and it is composed of longitudinal and lateral coordinates

in inertial frame $(x, y)_{CoG}$, the heading angle ψ_{CoG}, the longitudinal and lateral velocities in the body frame $(\dot{x}, \dot{y})_{CoG}$, the yaw rate $\dot{\psi}_{CoG}$ and the rotational velocities at the two wheels, (ω^f, ω^r). The input vector is denoted as $\mathbf{u} = [\delta, T^f, T^r]^T$, where δ is the steering angle, $T^{f,r}$ are the tractive and braking torques applied to the front and rear wheels, respectively. Positive torque denotes driving torque, while negative torque represents braking torque.

1.4.1 Tire model

In this section we give more details about the Pacejka's magic formula used in Section 1.4 for tire modeling. As mentioned in the previous section, the tire forces $F_l^{f,r}$ and $F_c^{f,r}$ depend on the the slip angles $\beta^{f,l}$, the slip ratio $s_r^{f,l}$, and the normal tire forces $F_z^{f,l}$. In order to compute the tire forces, we make the following reasonable assumption: $F_l^{f,r}$ and $F_c^{f,r}$ depend linearly to the normal forces $F_z^{f,l}$, that is

$$\begin{aligned} F_l^{f,r} &= F_z^{f,r} \mu_l^{f,r}(\beta^{f,r}, s_r^{f,r}) \,, \\ F_c^{f,r} &= F_z^{f,r} \mu_c^{f,r}(\beta^{f,r}, s_r^{f,r}) \,, \end{aligned}$$

where $\mu_l^{f,r}$ and $\mu_c^{f,r}$ are the combined longitudinal and cornering force coefficients, respectively.

We model the tire force by using a suitable version of Piajeka's magic formula, [21]. The combined longitudinal and corning force coefficient are give by:

$$\begin{aligned} \mu_l^{f,r}(\beta^{f,r}, s_r^{f,r}) &= \mu_{l0}(s_r^{f,r}) g_{l,\beta}(\beta^{f,r}, s_r^{f,r}) \,, \\ \mu_c^{f,r}(\beta^{f,r}, s_r^{f,r}) &= \mu_{c0}(\beta^{f,r}) g_{c,\beta}(\beta^{f,r}, s_r^{f,r}) \,. \end{aligned}$$

The pure longitudinal slip is given by

$$\mu_{l0}(s_r) = d_l \sin\{c_l \arctan[b_l k - e_l(b_l s_r - \arctan b_l s_r)]\} \,,$$

and the pure cornering slip by

$$\mu_{c0}(\beta) = d_c \sin\{c_c \arctan[b_c\beta - e_c(b_c\beta - \arctan b_c\beta)]\} .$$

The pure longitudinal and lateral slip are then combined by functions g_{l,s_r} and $g_{c,\beta}$ defined as follows:

$$g_{l,\beta}(\beta, s_r) = \cos\left[c_{l,\beta} \arctan\left(\beta \frac{h_{bl_1}}{1 + h_{bl_2}^2 s_r^2}\right)\right] ,$$
$$g_{c,\beta}(\beta, s_r) = \cos\left[c_{c,\beta} \arctan\left(s_r \frac{h_{bc_1}}{1 + h_{bc_2}^2 \beta^2}\right)\right] .$$

We have developed a family of reduced-order car models designed for different applications. These models include a kinematic bicycle model, which enables us to predict vehicle dynamics for trajectory generation. Notably, we have provided this kinematic bicycle model in three different coordinate sets to accommodate various needs. Additionally, we have developed a dynamic bicycle model that takes into account both translational and rotational dynamics, providing a more detailed description of a vehicle's behavior. This dynamic model considers the various forces and moments acting on the vehicle and incorporates a nonlinear tire model based on the well-established Pacejka model. We plan to utilize the reformulated models (1.15) and (1.3) in Chapter 2 to develop a merging strategy. Additionally, Chapter 4 will rely on Model (1.15) to formulate effective lane change strategies. In the context of Chapter 3, the reformulated model presented in Equation (1.19) will play a key role in the proposed maneuver generation algorithm. Furthermore, Model (1.20) will be used in the same chapter to validate the proposed algorithm through simulation.

Chapter 2

Optimal Control-based Strategy for Merging Maneuvers

In this chapter we address the merging problem for Autonomous Vehicles and develop an optimal control-based strategy in order to generate feasible trajectories.

2.1 Introduction

In the context of autonomous driving, one of the most critical maneuvers is the merging maneuver. As an Autonomous Vehicle (AV) approaches a busy intersection, it needs to perceive the presence of surrounding vehicles, predict their future intentions, and find the right space-time gap for a successful merge. However, accomplishing this task makes the analysis and design of the planning strategies particularly challenging. Finding the “right" space-time gap requires a suitable strategy to ensure safe and efficient merges under dynamic traffic conditions.

Many approaches have been proposed in the literature to tackle the challenging problem of autonomous merging on various road layouts. These methods range from lane changes along straight lanes, as demonstrated in [22], to more complex scenarios involving intersections with turning maneuvers, as investigated in [23], and merging into roundabouts, as proposed in [24].

A common assumption in these approaches is their reliance on inter-vehicle communications, enabling vehicles to exchange information and coordinate their actions efficiently. While this communication-based paradigm offers promising results in well-connected environments, it fails in handling situations where vehicles are partially or fully disconnected.

Addressing this limitation, [7] presents a Model Predictive Control (MPC) scheme tailored for the merging problem in a partially disconnected motorway scenario. This approach optimizes the longitudinal motion of the merging vehicle by generating smooth acceleration and deceleration profiles. Additionally, in order to take into account the unpredictability of the main lane vehicle's motion (the obstacle from the merging vehicle's perspective), the MPC scheme utilizes a penalty function for collision avoidance.

Recently, the coordination of autonomous vehicles (AVs) at intersections was presented in [25], specifically focusing on fixed-order crossing scenarios. The proposed algorithm addresses several critical challenges, including handling nonlinear dynamics, economic objective functions, and scenarios involving turning vehicles. In order to generate merging maneuvers, the algorithm formulates the problem as a constrained optimal control problem (OCP). By doing so, it ensures that each AV can traverse conflict zones in a mutually exclusive fashion, taking into account fixed-order crossing and collision avoidance. Notably, the proposed algorithm is designed to work even in situations where inter-vehicle com-

munication is limited or absent. Instead of relying on communication-dependent coordination, the authors model non-cooperative agents as uncertain systems. The addition of suitable constraints to the optimization problem, as previously explored by the same authors in [6] and [26], further enhances the algorithm's ability to manage uncertainties and ensure safety. This enables the algorithm to handle scenarios where vehicles might not be connected.

In this chapter, the merging problem is addressed from a different perspective. We assume there is no inter-vehicle communication and we focus on the trajectory generation of the AV: we propose an optimization-based strategy in order to compute collision-free merging trajectories with the right trade-off between trajectory-tracking and maneuver-regulation behaviors. *Trajectory tracking* refers to the control strategy employed to follow a predefined trajectory. A trajectory typically consists of a series of desired positions and velocities parametrized over time. *Maneuver regulation* is a path-following problem that is concerned with the design of control laws to reach and follow a geometric path. A secondary goal is to satisfy some additional requirements such as to follow the path with some desired velocity. Please refer to [27, 28, 29] for a discussion on these two approaches.

The rest of this chapter is dedicated to the development of a trajectory generation strategy for merging maneuvers. In Section 2.2 we describe the merging problem and describe how to use curvilinear coordinate, see Chapter 1.2.1, and Virtual Target Vehicle (VTV), see Chapter 1.2.2 to formulate this problem. Specifically, given the nominal road geometry, the vehicle dynamics is initially described in terms of longitudinal and lateral coordinates. Then, we introduce the use of the VTV that is constrained to move along the lane into which the ego-vehicle has to merge. In Section 2.3, we formulate the OCP. In particular, we set up a constrained OCP

in terms of the longitudinal and lateral coordinates and the kinematic position error between the ego-vehicle and the VTV. Moreover, based on the obstacles' predictions, we enforce suitable kinematic coordinates constraints to generate collision-free trajectories. Finally, in Section 2.4, we provide numerical computations highlighting some interesting features captured by the proposed strategy.

2.2 Problem Formulation

In the following section, we present the merging scenario, the car vehicle model (slightly modified in comparison to the one discussed in Chapter 1.3), and formulate the merging problem in terms of longitudinal and transverse coordinates and VTV.

2.2.1 The Motivating Scenario

Let us consider the merging scenario represented in Figure 2.1. The intersection is composed of two incoming lanes, called the “ego lane" and the “target lane", and a crossing zone, i.e., the merging zone.

The ego-vehicle is traveling along the ego lane and is supposed to merge into the target lane while performing a right turn. Moreover, the ego-vehicle must yield the right-of-way to (human-driven) vehicles traveling along the target lane. From now on, we call “obstacles" the (human-driven) vehicles.

We assume that

(i) the obstacles do not cooperate with the ego-vehicle, meaning they do not adjust their paths to facilitate merging. The ego-vehicle must find a suitable time-space gap to cross the intersection without causing a collision with any obstacles.

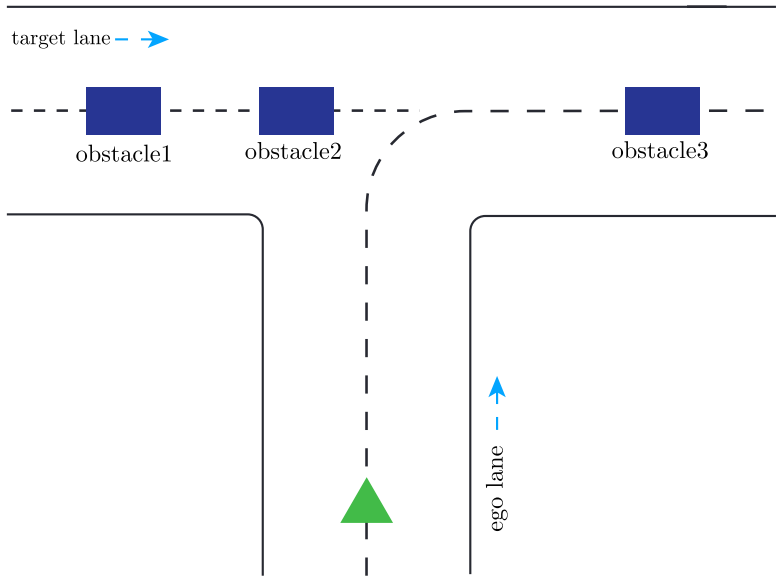


Figure 2.1: The merging scenario. The ego-vehicle is approaching an intersection where it has to yield the right-of-way to the obstacles. Travel directions are indicated by light blue arrows.

(ii) the ego-vehicle has access to the current state (position and velocity) of the obstacles as well as their predicted future positions, typically obtained from a motion forecasting module.

In such a scenario, we are interested in generating a feasible trajectory for the ego-vehicle that best approximates a desired one with road boundary, collision avoidance, and input control constraints. It is worth noting that, in a typical hierarchical motion planner framework, the generated trajectory can be used as a reference trajectory for a low-level controller.

2.2.2 Constrained Ego-vehicle Model

The equations of motion are based on the kinematic bicycle model we developed in Chapter 1.3 and recapitulated below for reference:

$$\begin{aligned} \dot{x} &= v\cos\psi\,, \\ \dot{y} &= v\sin\psi\,, \\ \dot{\psi} &= v\kappa\,, \end{aligned}$$

where (x, y) are the longitudinal and lateral coordinates with respect to the inertial frame and ψ is the heading angle. The control inputs are the velocity v and the curvature κ.

In practical application, direct manipulation of the velocity v and curvature κ of the ego-vehicle may not always be possible. Instead, control is typically exerted through acceleration a and the curvature rate u_κ. In order to take this into account, we expand the model to include v and κ as state variables and adopt a and u_κ as modified inputs:

$$\begin{aligned} \dot{x} &= v\cos\psi\,, \\ \dot{y} &= v\sin\psi\,, \\ \dot{\psi} &= v\kappa\,, \\ \dot{\kappa} &= u_\kappa\,, \\ \dot{v} &= a\,. \end{aligned} \tag{2.1}$$

It is worth noting that we consider such a simple vehicle model for the following reasons. First, this model has no parameters, thus allowing to focus on the trajectory generation approach. Second, for urban autonomous driving, the kinematic bicycle model has comparable accuracy with a dynamic one, [18], especially for low acceleration values.

Given this consideration, the vehicle model (2.1) is subject to several constraints that ensure both safety and comfort. In particular, the

velocity is bounded by two constants v_{min}, v_{max}, i.e.,

$$v_{min} \leq v \leq v_{max} \,. \tag{2.2}$$

The curvature and its rate are bounded in module as follows,

$$\begin{aligned} |\kappa| &\leq \kappa_{max} \,, \\ |u_\kappa| &\leq u_{\kappa_{max}} \,. \end{aligned} \tag{2.3}$$

Finally, in order to take into account passenger comfort, the longitudinal acceleration a and the lateral acceleration, $v^2\kappa$, are coupled by the ellipse constraint, [23],

$$\left(\frac{2a - (a_{max} + a_{min})}{(a_{max} - a_{min})}\right)^2 + \left(\frac{v^2\kappa}{a_{lat_{max}}}\right)^2 \leq 1 \,. \tag{2.4}$$

2.2.3 Longitudinal and Transverse Coordinates and Virtual Target Vehicle

Given a geometric path for the ego lane, we describe the ego-vehicle kinematics, as outlined in (2.1), with respect to the longitudinal and transverse coordinates (s, w), see Figure 2.2. Following the derivations in Chapter 1.3.1, we can express the ego-vehicle kinematics with these transformed coordinates as follows:

$$\begin{aligned} \dot{s} &= \frac{v\cos\mu}{1 - w\bar{\kappa}_{cl}(s)} \,, \\ \dot{w} &= v\sin\mu \,, \\ \dot{\mu} &= v\kappa - \bar{\kappa}_{cl}(s)\dot{s} \,. \end{aligned}$$

Next, by considering the target lane also to be arc-length parametrized with a smooth center-line, we introduce the use of the VTV that is constrained to move along this center-line, as depicted in Figure 2.2. As

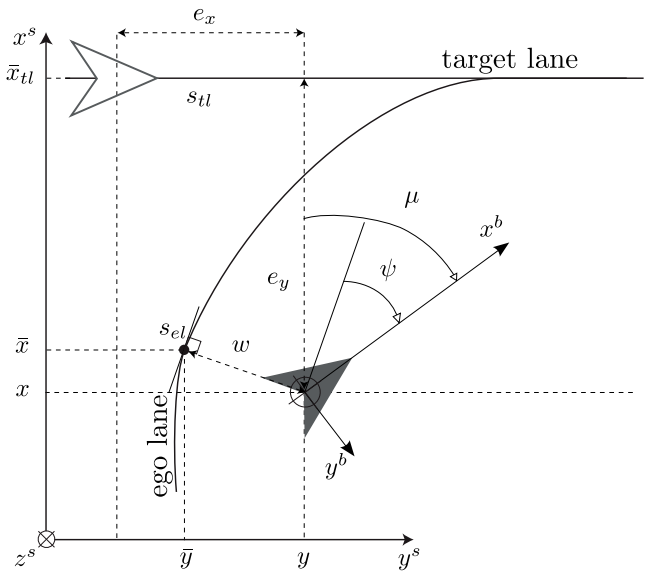


Figure 2.2: Local coordinates around the ego and target path. The bold triangle and the empty triangle indicate the ego-vehicle and the VTV, respectively. The solid lines indicate the center-line of the ego and target lane.

the ego-vehicle approaches the target lane, its objective is to track the VTV. For the sake of presentation, we restrict our attention to the case of a straight target lane, as the one depicted in Figure 2.1. Given this assumption, the position of the ego vehicle relative to the VTV, defined in Chapter (1.3.2), is simplified to:

$$\dot{e}_x = v \cos e_\psi - v_{vtv} \,,$$
$$\dot{e}_y = v \sin e_\psi \,.$$

Finally, bringing these elements together, we re-write the non-linear

system (2.1) in terms of the newly defined coordinate sets:

$$\begin{aligned}
\dot{s} &= \frac{v\cos\mu}{1-w\bar{\kappa}_{cl}(s)}\,,\\
\dot{w} &= v\sin\mu\,,\\
\dot{\mu} &= v\kappa-\bar{\kappa}_{cl}(s)\dot{s}\,,\\
\dot{s}_{tl} &= v_{vtv}\,,\\
\dot{e}_x &= v\cos e_\psi - v_{vtv}\,,\\
\dot{e}_y &= v\sin e_\psi\,,\\
\dot{\kappa} &= u_\kappa\,,\\
\dot{v} &= a\,.
\end{aligned} \tag{2.5}$$

Here, $\mathbf{x}$=$[s, w, \mu, \kappa, v, s_{tl}, e_x, e_y]$ and $\mathbf{u}$=$[u_\kappa, a, v_{vtv}]$ represent the state and control vectors, respectively.

2.3 Optimal Control Problem Formulation

In order to formulate the OCP, in this section we specify additional state-input constraints and define the cost function to be optimized.

We start by defining two additional constraints. First, the ego-vehicle is required to satisfy the road boundaries. In the transformed coordinate system, this constraint simplifies to bounding the lateral displacement w as follows:

$$|w| \leq w_{max}\,. \tag{2.6}$$

Second, to generate a collision-free trajectory, we impose that, at any time t, the ego-vehicle must be at a distance greater than $d_{collision}$ from any obstacles. Such a distance takes into account the safety distance between the ego-vehicle and the obstacles, and an additional distance to model the right-of-way of obstacles (as required in the merging problem,

Section 2.2.1). This constraint can be formulated by defining a circle centered at the obstacle front axes, with radius $d_{collision}$. Specifically, given the front axis position of the i-th obstacle and its future predictions, $(x^i_{obs}(t), y^i_{obs}(t))$, we impose that the constraint[1]

$$(x - x^i_{obs})^2 + (y - y^i_{obs})^2 \geq d^2_{collision}\,, \tag{2.7}$$

is satisfied for all times t. In order to include this constraint in the new proposed formulation, we re-write (2.7) with respect to the new set of coordinates. Given the longitudinal coordinate of the ego-vehicle, $s(t)$, we can express the obstacle position as

$$\begin{bmatrix} x^i_{obs} \\ y^i_{obs} \end{bmatrix} = \begin{bmatrix} \bar{x}_{cl}(s) \\ \bar{y}_{cl}(s) \end{bmatrix} + R_z(\bar{\psi}_{cl}(s)) \begin{bmatrix} s^i_{obs} - s \\ w^i_{obs} \end{bmatrix}, \tag{2.8}$$

where s^i_{obs} and w^i_{obs} are the longitudinal and lateral coordinates of obstacle i, respectively. Next, by subtracting (2.8) from (1.11), we have

$$\begin{bmatrix} x - x^i_{obs} \\ y - y^i_{obs} \end{bmatrix} = R_z(\bar{\psi}_{cl}(s)) \begin{bmatrix} s - s^i_{obs} \\ w - w^i_{obs} \end{bmatrix}. \tag{2.9}$$

Substituting (2.9) in (2.7), we get the collision avoidance in the equivalent form

$$(s - s^i_{obs})^2 + (w - w^i_{obs})^2 \geq d^2_{collision}\,. \tag{2.10}$$

Now we are ready to define the cost function. We start giving an informal idea of the proposed strategy, which is based on the following two observations. First, when the ego-vehicle is far away from the merging zone, we are interested to follow a desired path (i.e., the center-line of the ego lane) with a desired velocity assigned to it (i.e., a space-varying

[1]For the sake of presentation, we consider only circular boundary shapes, although other shapes can be taken into account.

velocity). Such a behavior can be captured by minimizing the following cost,

$$J_{el}(\mathbf{x}) = q_1 w^2 + q_2 \mu^2 + q_3 \kappa^2 + q_4 (v - v^d(s))^2$$

where $q_1, q_2, q_3, q_4 \geq 0$.

Second, when the ego-vehicle is approaching the merging zone, we are interested to track a time parameterized path (i.e., the center-line of the target lane) defined by a desired velocity $v^d(s_{tl})$. Here, we employ a quadratic cost term with respect to the kinematic position error between the ego-vehicle and the VTV, and the velocity of the VTV with respect to the desired one

$$J_{tl}(\mathbf{x}, \mathbf{u}) = q_5 e_x^2 + q_6 e_y^2 + r_1 (v_{vtv} - v_{vtv}^d(s_{tl}))^2$$

where $q_5, q_6 \geq 0$, and $r_1 > 0$.

We define the cost function as a convex combination of the previous function terms and an additional quadratic term in order to take into account the control effort:

$$J(\mathbf{x}, \mathbf{u}) = (1 - \alpha) J_{el}(\mathbf{x}) + \alpha J_{tl}(\mathbf{x}) + r_2 u_k^2 + r_3 a^2 \,, \tag{2.11}$$

where $\alpha \in [0, 1]$ is a switch cost function based on the distance between the ego-vehicle and the VTV. In particular, we use a sigmoid function

$$\alpha(e_x, e_y) = \frac{1}{1 + \exp\left(\sqrt{e_x^2 + e_y^2} - \gamma\right)} \,,$$

where γ is a given parameter that specifies the distance from the merging zone.

We highlight that in [3], the weighting term associated with the VTV's velocity can be used to "morph" between trajectory tracking and maneuver regulation features. In contrast to the previous approach, we

embed such a morphing feature into the optimization process by proposing a suitable cost function.

We are ready to formulate the optimal control problem as follows

$$\begin{aligned} \min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} & \int_0^{t_f} J(\mathbf{x}(\tau),\mathbf{u}(\tau))\, d\tau + m(\mathbf{x}(t_f)) \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t),\mathbf{u}(t))\,,\, \mathbf{x}(0) = \mathbf{x}_0 \\ & h(\mathbf{x}(t),\mathbf{u}(t)) \leq 0 \end{aligned} \tag{2.12}$$

where $t_f > 0$ is fixed, $\dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u})$ describes the nonlinear equations (2.5), $J(\mathbf{x},\mathbf{u})$ is the cost function as in (2.11), $h(\mathbf{x},\mathbf{u})$ are the state/input constraints (2.2),(2.3), (2.4), (2.6), (2.10) and $m(\mathbf{x}(t_f))$ is the Mayer term.

We highlight that the obstacle avoidance collision constraint (2.10) makes the optimization problem nonconvex and computationally challenging. In particular, we solve (2.12) by using the ACADO toolkit, [30]. The multiple-shooting discretization (see Appendix B), is employed with a Runge-Kutta integrator of order 4 and a sampling time of 0.2 s. The underlying Quadratic Programs (QP) are condensed and solved using an online active set strategy implemented in the software qpOASES, [31].

2.4 Numerical Computations

In this section we provide numerical computations showing the effectiveness of the proposed approach. We start with a relatively simple scenario: the ego-vehicle is traveling along the ego lane and is approaching the target lane, where an obstacle is moving. Then, as a more challenging scenario, we increase the number of obstacles. The ego lane is modeled as a 90° right turn with a radius of 20 m. The space-varying desired velocity is equal to 7.2 m/s along the two straight sections and 5.2 m/s along the turn. The desired VTV velocity is constant along the entire

target lane and equal to 7.2 m/s. The constraints parameters are based on [18] and on driving experience, as reported Table 2.1.

Parameter	**Value**	**Units**
Minimum velocity (v_{min})	0	m/s
Maximum velocity (v_{max})	10	m/s
Minimum acceleration (a_{min})	-1.5	m/s^2
Maximum acceleration (a_{max})	1	m/s^2
Maximum lateral acceleration ($a_{lat_{max}}$)	2	m/s^2
Maximum curvature (κ_{max})	0.2	m^{-1}
Maximum width (w_{max})	1.5	m
Collision distance ($d_{collision}$)	10	m

Table 2.1: Set of constraints parameters for merging trajectory generation strategy. The parameters include the minimum and maximum velocities denoted as v_{min} and v_{max}, respectively. Furthermore, the minimum and maximum accelerations, are represented by a_{min} and a_{max}. Lateral motion is addressed through the parameter $a_{lat_{max}}$, denoting the maximum lateral acceleration. Maximum curvature is encapsulated by the parameter κ_{max}. The parameter w_{max} designates the maximum width of the lane. Additionally, the collision distance, denoted as $d_{collision}$, outlines the distance within which collision detection mechanisms must operate to ensure the safety.

We use a planning horizon of 20 s which allows the ego-vehicle to perform a merging maneuver for the entire set of numerical computations. We encourage the reader to refer to the video attachment[2] related to the 2D-plane trajectories of the numerical computations presented below.

[2] `https://youtu.be/jyvSBll2uWA?si=BOIHhASkTG6YxlVK`

2.4.1 Merging with one obstacle

The ego-vehicle initial position is $(x_0, y_0) = (0, 0)$, with heading $\psi_0 = 0$, and velocity $v_0 = 26\ km/h$ (i.e., almost 7.2 m/s). The obstacle is in position $(x^{obs}(0), y^{obs}(0)) = (35, -10)$ and is traveling along the target lane with a constant velocity of $v^{obs} = 10\ km/h$ (2.78 m/s). We solve the optimization problem (2.12), a set of weighting cost terms are selected (obtained after a trial and error process combined with our experience in the nonlinear system (2.5)): $q_1 = 5.0$, $q_2 = 0.1$, $q_3 = 0.5$, $q_4 = 10.0$, $q_5 = 0.01$, $q_6 = 0.01$, $r_1 = 0.01$, $r_2 = 1.0$, and $r_3 = 0.1$. The resulting optimal trajectory is shown in Figure 2.3. Next, we analyze some interesting features of the generated trajectory. At first glance, we can identify a "pass after" behavior. Basically, the ego-vehicle decelerates, thus giving the way to the obstacle (see Fig.2.3f).

In the generated optimal trajectory, we can identify two phases. First, at the beginning, the ego-vehicle is far away from the merging area and the cost J_{el} is minimized: the ego-vehicle is following the center-line (the lateral displacement is almost zero), and decreases its velocity to face the right turn. Second, at about $t = 5\ s$, the ego-vehicle is close to the VTV and the cost J_{tl} is minimized: the ego-vehicle applies a positive curvature and moves toward the outside edge of the right-turn to minimize the kinematic error with respect to the VTV. Moreover, a stronger deceleration is applied (satisfying the ellipse constraint, see Figure 2.3e), thus giving the way to the obstacle. Finally, we analyze the (local) optimal VTV velocity profile, see Figure 2.3d. In the beginning, the VTV has zero velocity, which means that the VTV is "waiting" the ego-vehicle while is traveling along the ego lane. As the ego-vehicle approaches the target lane, the VTV accelerates and its velocity reaches the value of 2.78 m/s, which is exactly the velocity of the obstacle. Consequently, the ego-vehicle "tracks" the VTV position, because the J_{tl} is minimized.

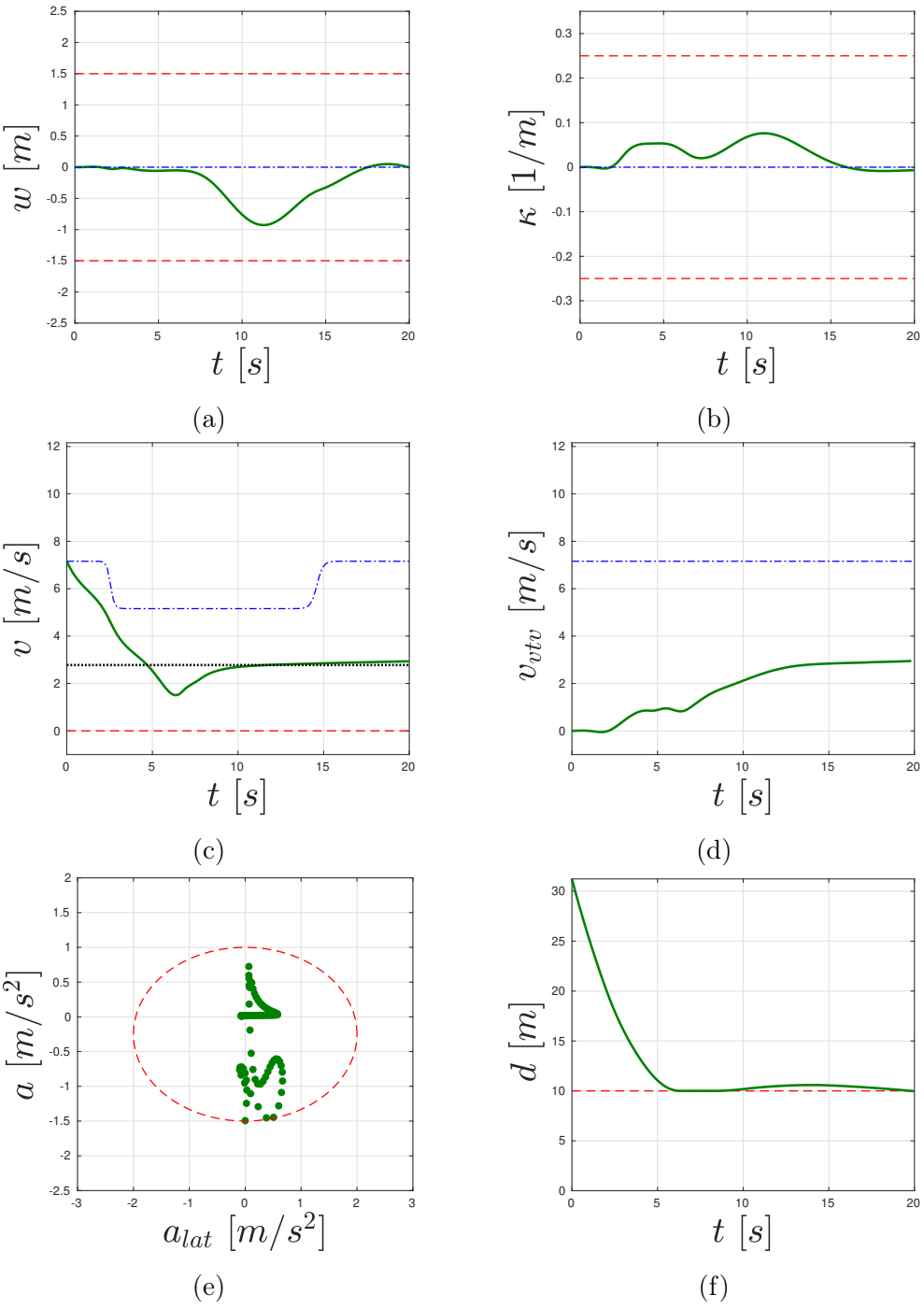


Figure 2.3: Merging with one obstacle: pass after behavior. The optimal (green solid line), the desired (blue dash-dot line) trajectories, and constraints (red dash lines) are shown. Obstacle initial position $(x^{obs}(0), y^{obs}(0)) = (35, -10)$.

It is interesting to investigate how the initial obstacle position affects the generated trajectory. We set $(x^{obs}(0), y^{obs}(0)) = (35, -15)$ and solve the optimization problem. The optimal trajectory is shown in Figure 2.4. In the initial stage, the ego-vehicle follows the path of the ego lane, maintaining the desired velocity. As the ego-vehicle approaches the target lane, the optimization algorithm finds a spatial and temporal gap, affording the opportunity for performing a “pass before" relative to the obstacle. This behavior entails that the VTV increases its velocity, achieved through a sharp acceleration. In particular, the ego-vehicle starts to track the VTV. Two distinct phases can be identified. Initially, the ego-vehicle moves toward the inner edge of the upcoming turn. This behavior is motivated by the intention to minimize the curvature, thereby complying with the imposed acceleration constraint. This phase culminates with the ego-vehicle nearing the the so-called apex point of the curve – the point closest to the inside of the corner, also referred to as the clipping point. During the latter phase, the ego-vehicle accelerates in a smooth fashion. This sequential interaction of these phases translates the ego-vehicle's trajectory form one characterized by the ego lane following to a trajectory that follows the path of the VTV enabling the ego-vehicle to perform a safe merging manuever.

2.4.2 Merging into traffic

We introduce a higher level of complexity by considering the presence of four obstacles traveling along the target lane, simulating a more challenging scenario. The coordinates $(x_1^{obs}(0), y_1^{obs}(0))$=$(35, 5)$, $(x_2^{obs}(0), y_2^{obs}(0)) = (35, -13)$, $(x_3^{obs}(0), y_3^{obs}(0)) = (35, -29)$ and $(x_4^{obs}(0), y_4^{obs}(0)) = (35, -51)$ define the position of these obstacle, each moving with a constant velocity of 3.3 m/s (12 km/h) along the target lane. The initial position,

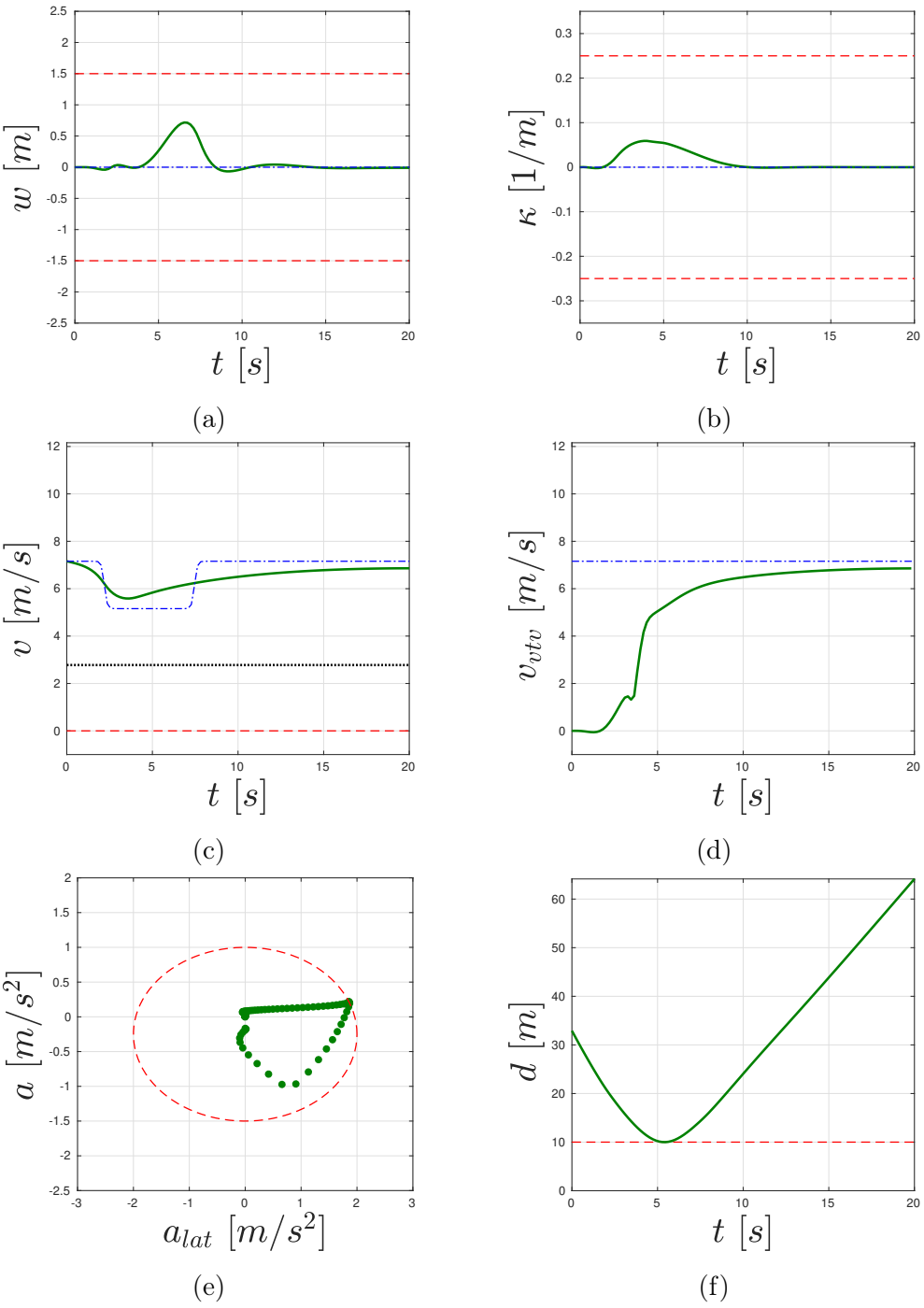


Figure 2.4: Merging with one obstacle: pass before behavior. The optimal (green solid line), the desired (blue dash-dot line) trajectories, and constraints (red dash line) are shown. Obstacle initial position $(x^{obs}(0), y^{obs}(0))$=$(35, -15)$.

orientation and velocity of the ego-vehicle are $(x, y) = (0, 0)$, $\psi = 0$ and $v_0 = 26\ km/h(7.2\ m/s)$, respectively. The optimal trajectory is shown in Figure 2.5. The ego-vehicle starts the maneuver by following the ego lane reference velocity, see Figure 2.5c, for $t \in [0, 2]\ s$. As time advances to approximately 3 s, the ego-vehicle approaches the target lane. Here, the optimization problem minimizes the J_{tl} objective function.

Indeed, we highlight that:

(i) the ego-vehicle decelerates to achieve a complete halt before entering the intersection, ensuring that the collision avoidance is always satisfied,

(ii) slightly moves toward the inner edge of the right turn to gain the best position for the merging,

(iii) the VTV moves toward to minimize the kinematic error (mainly because the projection of the ego-vehicle to the target lane corresponds to a point which is slightly moving forward). Indeed, there is no space-time gap to perform a merging maneuver.

Subsequently, as obstacle1 safely crosses the the intersection zone, the ego-vehicle, having come to a complete stop, accords right-of-the-way to obstacle2 and obstacle3 (see Figure 2.5f). At about $t = 10\ s$, the right space-time gap is found: the VTV accelerates and the starts tracking the VTV position. This behavior culminates in a safely execution of a “pass before" maneuver relative to obstacle4.

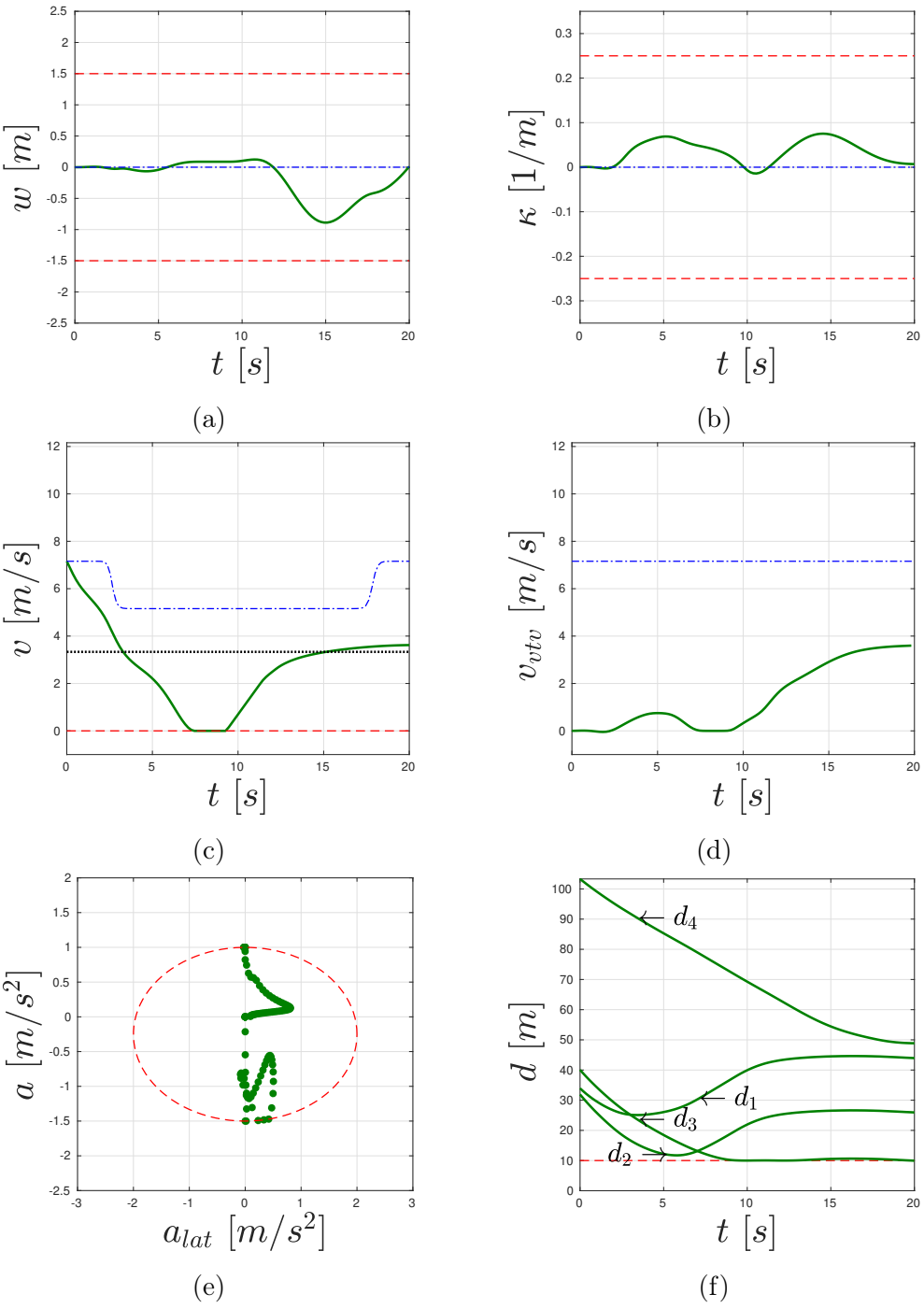


Figure 2.5: Merging with four obstacles: Pass among behavior. The optimal (green solid line), the desired (blue dash-dot line) trajectories, and constraints (red dash line) are shown.

Chapter 3

Real-time Maneuvers Generation Algorithm

In this chapter we develop a novel maneuver generation algorithm, based on nonlinear optimal control technique introduced in [1], to generate collision-free trajectories for Autonomous Vehicles.

3.1 Introduction

Several approaches have been proposed in the literature to solve the collision-free trajectory generation problem, see [32] for a survey. In the last years, optimization-based methods have received increasing attention thanks to the availability of efficient numerical techniques, as, e.g., [1, 2, 33, 34, 35], and the possibility to systematically include vehi-

cle dynamics and suitable constraints in the formulation. The design of optimal collision-free trajectories is challenging mainly because the collision avoidance constraint is non-convex and computationally difficult to handle, see, e.g., [36]. For these reasons, appropriate approximations are introduced. For example, in [10], a linear time-varying constraint on the lateral displacement is proposed. In [37], a non-convex (ellipse-shaped) constraint is taken into account. In [38], an exact reformulation of the collision avoidance constraint is proposed. However, such a reformulation implies the introduction of additional optimization variables and constraints. Once the avoidance constraint has been formulated, it can be embedded into

(i) a constrained optimization problem, as in [9, 4, 39],

(ii) an appropriate unconstrained optimization problem in which the objective function is augmented with an artificial potential field representing the distance to the obstacle, see e.g., [40, 41].

With the collision avoidance formulation in hand, a classical approach used in the literature to design a trajectory is trajectory tracking, aiming to force the vehicle to reach and track a time-parametrized path. However, the requirement of tracking a time-parametrized reference has the following main limitation: poor tracking of the reference path in presence of external disturbances and unmodeled dynamics, see, e.g., [29] and [42]. An alternative approach is maneuver-regulation, [43], namely, converge and follow a desired path with the additional requirement to satisfy a velocity profile along it.

The next part of the chapter is dedicated to the development of the proposed real-time collision-free maneuver generation algorithm, to efficiently solve the collision-free trajectory generation problem. In Section 3.2, we describe the system dynamics with respect to the new set

of coordinates and formulate the collision-free avoidance constraint. In Section 3.3, we describe our real-time strategy to compute collision-free maneuvers. Finally, in Section 3.4 and Section 3.5.2, we provide numerical computations and an experimental test, highlighting the main features of the proposed strategy.

3.2 Problem Formulation

In this section we briefly describe the system dynamics with respect to the longitudinal and transverse coordinates and formulate the collision avoidance constraint.

3.2.1 Ego-vehicle Model and Longitudinal Coordinate Parameterization

In this subsection, we describe the steps necessary for deriving the new formulation for the vehicle dynamics.

First, we describe the ego-vehicle model. Previously we discussed the characterization of the ego-vehicle's 2D motion in typical urban scenarios, where low acceleration values are applied, see Chapter 1. In order to capture this motion accurately, we use a kinematic model, as outlined in equation (1.10). For the proposed formulation, we aim to improve the presented ego-vehicle model by incorporating acceleration as a controllable input. The augmented ego-vehicle model can be expressed as:

$$\begin{aligned} \dot{x} &= v\cos\psi\,, \\ \dot{y} &= v\sin\psi\,, \\ \dot{\psi} &= v\kappa\,, \\ \dot{v} &= a\,, \end{aligned} \tag{3.1}$$

where (x, y) are the longitudinal and lateral coordinates with respect to the inertial frame, ψ is the heading angle, and v is the velocity. The control inputs are the curvature κ and the acceleration a.

Second, given the road geometry, we assume that the path has a reasonably smooth (at least C^2) arc-length parametrized center-line, denoted as $(\bar{x}_{cl}(s), \bar{y}_{cl}(s))$. Following the derivation detailed in Section 1.3.3, we re-write the ego-vehicle dynamics (3.1) using the longitudinal coordinates s as the independent variable:

$$\begin{aligned}
\bar{w}'(s) &= (1 - \bar{\kappa}_{cl}(s)\bar{w}(s)) \tan \bar{\mu}(s)\,, \\
\bar{\mu}'(s) &= \frac{1 - \bar{\kappa}_{cl}(s)\bar{w}(s)}{\cos \bar{\mu}(s)} \bar{\kappa}(s) - \bar{\kappa}_{cl}(s)\,, \\
\bar{v}'(s) &= \frac{1 - \bar{\kappa}_{cl}(s)\bar{w}(s)}{\bar{v}(s) \cos \bar{\mu}(s)} \bar{a}(s)\,, \\
\bar{t}'(s) &= \frac{1 - \bar{\kappa}_{cl}(s)\bar{w}(s)}{\bar{v}(s) \cos \bar{\mu}(s)}\,,
\end{aligned} \tag{3.2}$$

In (3.2) the state and the control vector are $\bar{\mathbf{x}}$=$[\bar{w},\ \bar{\mu},\ \bar{v},\ \bar{t}\,]$ and $\bar{\mathbf{u}}$=$[\bar{\kappa},\ \bar{a}]$ respectively.

Remark 3.1 *We highlight that we include $\bar{t}$ into the dynamics with reference to* (3.2)*. Such additional state variable will be exploited for the formulation of the avoidance constraint, as will be shown in the next subsection.*

3.2.2 Obstacle Avoidance Formulation

A typical formulation for obstacle avoidance involves the imposition of a security distance, denoted as $\tilde{d}$, between the ego-vehicle and the obstacle. This approach is commonly found in the literature, as exemplified in [44]. The security distance accounts for factors such as the size of both the

ego-vehicle and the obstacle, along with an added safety margin. Specifically, given the coordinate center of the obstacle and its future predictions, $(x_{obs}(\cdot), y_{obs}(\cdot))$, we have that the following inequality constraint

$$\left(\frac{x - x_{obs}}{\tilde{d}}\right)^2 + \left(\frac{y - y_{obs}}{\tilde{d}}\right)^2 \geq 1 \tag{3.3}$$

must be satisfied for all times t, see Figure 3.1. In order to include such a constraint in our maneuver regulation problem, we first re-write (3.3) with respect to the (s, w) coordinates. Given the longitudinal coordinate s of the ego-vehicle, we can express the obstacle position as

$$\begin{bmatrix} x_{obs} \\ y_{obs} \end{bmatrix} = \begin{bmatrix} \bar{x}_{cl} \\ \bar{y}_{cl} \end{bmatrix} + R_z(\bar{\psi}_{cl}) \begin{bmatrix} s_{obs} - s \\ w_{obs} \end{bmatrix}, \tag{3.4}$$

where s_{obs} and $w_{obs}(s)$ are the obstacle's longitudinal and lateral coordinates, respectively.

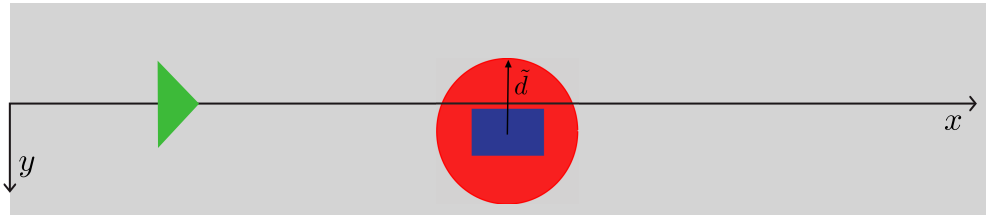


Figure 3.1: Obstacle avoidance constraint representation in Cartesian coordinate system. The lane, the obstacle and the ellipse constraint are depicted in gray, blue, and red, respectively. In order to satisfy the constraint, the ego-vehicle (green bold triangle) must be outside the red boundaries.

Next, by subtracting (3.4) from (1.11), we have

$$\begin{bmatrix} x - x_{obs} \\ y - y_{obs} \end{bmatrix} = R_z(\bar{\psi}_{cl}) \begin{bmatrix} s - s_{obs} \\ w - w_{obs} \end{bmatrix} . \tag{3.5}$$

Substituting (3.5) in (3.3), we get the following equivalent formulation, see Figure 3.2,

$$\left(\frac{s - s_{obs}}{\tilde{d}} \right)^2 + \left(\frac{w - w_{obs}}{\tilde{d}} \right)^2 \geq 1 . \tag{3.6}$$

By transforming (3.6) from a t-dependent to s-dependent description, the constraint can be written as

$$\left(\frac{\bar{w}(s) - \bar{w}_{obs}(s)}{\tilde{d}} \right)^2 \geq 1 . \tag{3.7}$$

The avoidance formulation as in (3.7) captures the case of static obstacles: at a given curvilinear coordinate s, the relative (lateral) distance between the ego-vehicle and the obstacle must be greater than $\tilde{d}$. However, it is very conservative (or, in some case, it fails) when dealing with

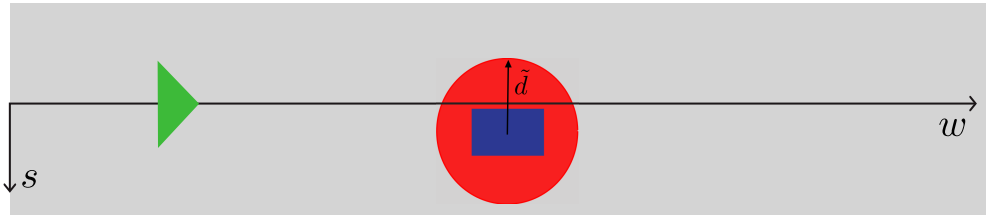


Figure 3.2: Obstacle avoidance constraint representation in longitudinal and transverse coordinate system. The lane, the obstacle and the ellipse constraint are depicted in gray, blue, and red, respectively. In order to satisfy the constraint, the ego-vehicle (greed bold triangle) must be outside the red boundaries.

moving obstacles. For example, let us consider the scenario illustrated in Figure 3.3: the ego-vehicle is moving along a lane (with $w = 0$) and has to avoid a moving obstacle which is on the right-side of the center-line. By using the avoidance formulation (3.7), the ego-vehicle will satisfy (if possible) the security distance $\tilde{d}$ for all s, thus providing a too conservative collision-free maneuver. This is due to the fact that the formulation (3.7) does not take into account the time evolution of the ego-vehicle and the obstacle. In order to overcome this limitation, we propose a novel constraint formulation as follows. We start from the following (somehow) simple idea: in order to avoid collision with a moving obstacle, the ego-vehicle and the obstacle must not be at the same longitudinal coordinate s at the same time instant t. Based on this idea, we introduce the time variable into (3.7), as follows:

$$\left(\frac{\bar{t}(s) - \bar{t}_{obs}(s)}{\tilde{t}}\right)^2 + \left(\frac{\bar{w}(s) - \bar{w}_{obs}(s)}{\tilde{d}}\right)^2 \geq 1\,, \tag{3.8}$$

where $\bar{t}_{obs}(s)$ is the time at which the obstacle is at the longitudinal coordinates s and $\tilde{t}$ is a temporal safety margin parameter. In Figure 3.4

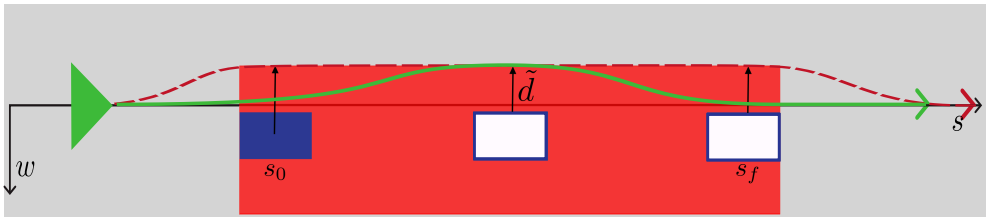


Figure 3.3: Avoidance maneuver scenario. The ego-vehicle (green triangle), the obstacle (solid blue rectangle) and its predictions (empty blue rectangles) are shown. The avoidance maneuvers obtained by using (3.7) and (3.8) are depicted, respectively, in dashed red line and solid green line.

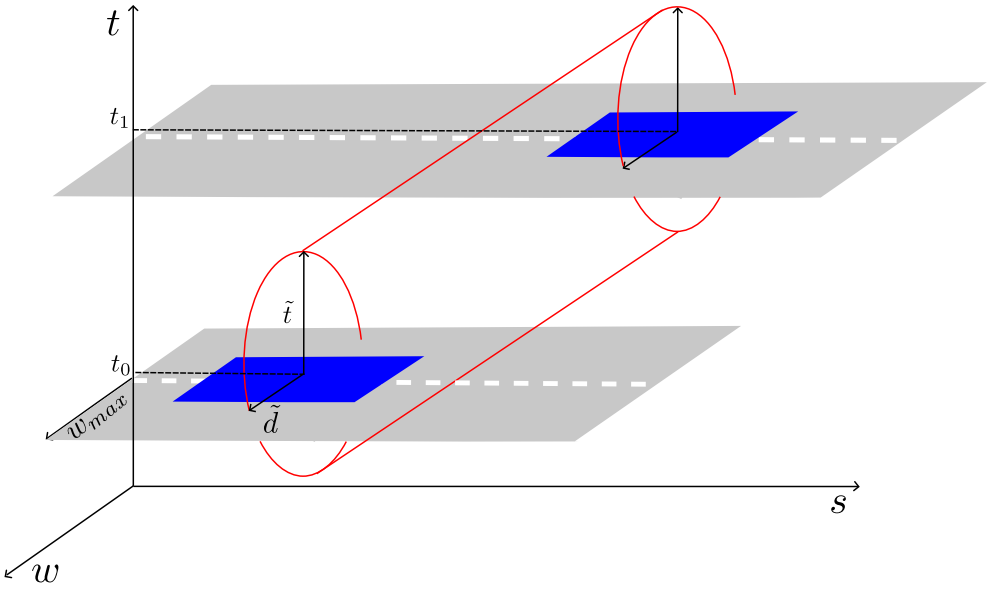


Figure 3.4: Obstacle avoidance constraint representation. The lane, the obstacle and the ellipse constraint are depicted in gray, blue, and red, respectively. In order to satisfy the constraint, the ego-vehicle must be outside the red boundaries.

we provide a 3D representation of the proposed avoidance constraint. We point out that the inequality constraint (3.8) is based on the coordinate center of the obstacle. Such formulation can be easily extended in order to take into account the obstacle's size (i.e., by using the projection of the rear and front axes along the road geometry).

Remark 3.2 *It is worth noting that the inequality constraint* (3.8) *is well-defined if unique values of* $\bar{w}_{obs}(s)$ *and* $\bar{t}_{obs}(s)$ *are determined for a given* s*. This is not the case for a static obstacle, because it has the same longitudinal coordinate* s *for all time values. For this reason, when dealing with static obstacles, we impose a sufficient large value for* $\tilde{t}$*.*

3.3 Maneuver Generation Strategy

In this section, we describe the optimal control-based strategy used to compute real-time collision-free maneuvers.

First, we define additional constraints. Specifically, the ego-vehicle is required to satisfy the road boundaries. This constraint assumes a very simple form with respect to the new set of coordinates, that is,

$$|\bar{w}(s)| \leq w_{max}\,.$$

To have a smooth function defining the constraint, we rewrite it in the equivalent form

$$\left(\frac{\bar{w}(s)}{w_{max}}\right)^2 - 1 \leq 0\,. \tag{3.9}$$

In order to take into account the operational limits of the kinematics model and the comfort of the passenger, we impose state and input constraints on (3.2) as follows. The velocity is bounded by two constants, i.e.,

$$\left(\frac{2\bar{v}(s) - (v_{max} + v_{min})}{v_{max} - v_{min}}\right)^2 - 1 \leq 0\,. \tag{3.10}$$

while the longitudinal acceleration a and the lateral acceleration, $v^2\kappa$, are coupled by the ellipse constraint, [23],

$$\left(\frac{2\bar{a}(s) - (a_{max} + a_{min})}{(a_{max} - a_{min})}\right)^2 + \left(\frac{\bar{v}(s)^2\bar{\kappa}(s)}{a_{lat_{max}}}\right)^2 - 1 \leq 0\,. \tag{3.11}$$

Moreover, in order to take into account the limited wheel steer angle, the curvature is bounded in module as follows,

$$\left(\frac{\bar{\kappa}(s)}{\kappa_{max}}\right)^2 - 1 \leq 0\,. \tag{3.12}$$

Second, we need to define the cost function to be optimized. Specifically, we are interested to follow the center-line of the road with a given velocity

profile, $\bar{v}_{cl}$, assigned on it, and, at the same time, minimize the control effort. Such a behavior can be captured by minimizing the following cost function,

$$J(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = (\bar{\mathbf{x}} - \bar{\mathbf{x}}^d)^T \; Q \; (\bar{\mathbf{x}} - \bar{\mathbf{x}}^d) + (\bar{\mathbf{u}} - \bar{\mathbf{u}}^d)^T \; R \; (\bar{\mathbf{u}} - \bar{\mathbf{u}}^d) \,, \tag{3.13}$$

where $\bar{\mathbf{x}}^d = [0, 0, \bar{v}_{cl}, 0]$, $\bar{\mathbf{u}}^d = [\bar{\kappa}_{cl}, 0]$ is the desired maneuver, $Q = diag(q_1, q_2, q_3, q_4)$ are a positive-semidefinite matrix, while $R = diag(r_1, r_2)$ is positive-definite one. Now we are ready to formulate the optimal control problem:

$$\begin{aligned} \min_{\bar{\mathbf{x}}(\cdot)\,\bar{\mathbf{u}}(\cdot)} & \int_0^{s_f} J(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau + m(\bar{\mathbf{x}}(s_f)) \\ \text{s.t.} \quad & \bar{\mathbf{x}}'(s) = f(\bar{\mathbf{x}}(s), \bar{\mathbf{u}}(s), s)\,, \; \bar{\mathbf{x}}(0) = \mathbf{x}_0 \\ & h(\bar{\mathbf{x}}(s), \bar{\mathbf{u}}(s)) \leq 0\,, \end{aligned} \tag{3.14}$$

where $s_f > 0$ is a fixed horizon, $\bar{\mathbf{x}}' = f(\bar{\mathbf{x}}, \bar{\mathbf{u}}, s)$ describes the nonlinear equations (3.2), $h(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ are the state/input constraints (3.8), (3.9), (3.10), (3.11), (3.12), $J(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is the stage cost as in (3.13) and $m(\bar{\mathbf{x}}(s_f))$ is the terminal cost which minimizes the L2 distance between the ego-vehicle state and the desired end state $\bar{\mathbf{x}}(s_f)$.

We solve (3.14) by using PRONTO, [1], see Appendix A. PRONTO is a direct method for solving continuous-time optimal control problems. It exhibits a second-order convergence rate to a local minimizer satisfying second-order sufficient conditions of optimality. The key point is the design of a projection operator, denoted as $\mathcal{P}$, which maps state-input curves $\boldsymbol{\xi} = (\bar{\boldsymbol{\alpha}}(\cdot), \bar{\boldsymbol{\mu}}(\cdot))$ to trajectories $\boldsymbol{\tau} = (\bar{\boldsymbol{x}}(\cdot), \bar{\boldsymbol{u}}(\cdot))$. This projection operator $\mathcal{P} : \boldsymbol{\xi} \to \boldsymbol{\tau}$ can be formally defined by a space-varying control low, as follows:

$$\begin{aligned} \bar{\mathbf{x}}(s) &= f'(\bar{\mathbf{x}}, \bar{\mathbf{u}}, s), \quad \bar{\mathbf{x}}(0) = \mathbf{x}_0\,, \\ \bar{\mathbf{u}}(s) &= \bar{\boldsymbol{\mu}}(s) + \boldsymbol{K}(s)(\bar{\boldsymbol{\alpha}}(s) - \bar{\mathbf{x}}(s))\,. \end{aligned}$$

It is worth to notice that the definition of $\mathcal{P}$ relies on the solution of a suitable linear quadratic optimal control problem providing the projection operator with a stability like property. Following [1], we handle constraints using a barrier function approach. In particular, the state-input constraints are relaxed by adding them to the cost functional. A barrier function can be defined as

$$b_\delta(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \int_0^{s_f} \sum_j \beta_\delta(-h_j((\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau)))) d\tau \,,$$

where

$$\beta_\delta(z) = \begin{cases} -\log z & \text{for } z > \delta \\ \frac{1}{2}\left(\frac{z-2\delta}{\delta} - 1\right) - \log \delta & \text{otherwise} \end{cases} . \tag{3.15}$$

Using the barrier function β_δ, the problem (3.14) becomes:

$$\begin{aligned} &\min_{\bar{\mathbf{x}}, \bar{\mathbf{u}}} \int_0^{s_f} J(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau)) + \epsilon b_\delta(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau \\ &\text{s.t.} \quad \bar{\mathbf{x}}'(s) = f(\bar{\mathbf{x}}(s), \bar{\mathbf{u}}(s), s)\,,\ \bar{\mathbf{x}}(0) = \mathbf{x}_0\,, \end{aligned} \tag{3.16}$$

for $\epsilon > 0$. The strategy to find an approximated solution to (3.14) can be summarized as follows. Starting with a reasonable large ϵ and δ, Problem (3.16) is iteratively solved by reducing the parameters at each iteration and thus pushing the trajectory toward the constraint boundaries. PRONTO, being a Newton descent method, can only guarantee convergence to a local minimum. We choose the initial guess as follows. Given the reference state-input desired reference, $(\bar{\mathbf{x}}, \bar{\mathbf{u}})^d$, which is not a maneuver (it does not satisfy the dynamics), we use the projection operator [45] to project the state-input desired reference into the feasible maneuvers manifold to obtain a suitable initial guess. Now, with an initial maneuver for the initialization and a desired reference, the algorithm iterates the following step:

Algorithm 1 Real-time Collision-free Maneuver Generation

Input: road geometry $(\bar{x}_{cl}, \bar{y}_{cl}, \bar{\psi}_{cl}, \bar{\kappa}_{cl}, \bar{v}_{cl})$,
bounds $(w_{max}, v_{min}^{max}, k_{max}, a_{min}^{max}, a_{lat_{max}})$,
obstacles data $(x_{obs}(t), y_{obs}(t))$

Initialization:

- compute $\mathbf{x}_0$ (i.e., project $(x, y, \psi)_0$ wrt the road geometry)
- setup dynamics $\bar{\mathbf{x}}' = f(\bar{\mathbf{x}}, \bar{\mathbf{u}}, s)$, $\mathbf{x}(0) = \mathbf{x}_0$
- compute $(\bar{w}_{obs}, \bar{t}_{obs})$ (i.e., project (x_{obs}, y_{obs}) w.r.t the road geometry)
- setup constraints $h(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ with bounds and $\bar{w}_{obs}$, $\bar{t}_{obs}$
- desired maneuver $\bar{\mathbf{x}}^d = [0, 0, \bar{v}_{cl}, 0]$, $\bar{\mathbf{u}}^d = [\bar{\kappa}_{cl}, 0]$

Set: $\epsilon = 1,\ \delta = 1$
for $k = 1, 2, ...$ **do**
compute: $(\bar{\mathbf{x}}, \bar{\mathbf{u}})_k = PRONTO((\bar{\mathbf{x}}, \bar{\mathbf{u}})_{k-1}, \epsilon, \delta)$
update $\epsilon,\ \delta$: $\epsilon \leftarrow \epsilon/6,\ \delta \leftarrow \delta/6$
end for

Output: $(\bar{\mathbf{x}}, \bar{\mathbf{u}})_{opt} = (\bar{\mathbf{x}}, \bar{\mathbf{u}})_k$

(i) compute the optimal collision-free maneuver by using PRONTO,

(ii) use the previous optimal maneuver as initial guess for the next step,

(iii) update the constraints parameters, ϵ and β,

(iv) solve (3.16) with an updated barrier function.

Algorithm 1 gives a pseudocode description of the real-time maneuver generation strategy. We want to stress that PRONTO ensures the recursive dynamics and constraints feasibility of intermediate solutions. As a

result, our algorithm can reliably use sub-optimal (intermediate) solutions in cases of overhead computations.

It is worth noticing that the approximate log barrier function (3.15) handle efficiently constraints (3.9), (3.10), (3.11), and (3.12). For example, (3.9) translates to the strongly convex function $\beta_\delta\left(1-\left(\frac{\bar{w}}{w_{max}}\right)^2\right)$. At $\sigma=1$, this function simplifies to $\left(\frac{\bar{w}}{w_{max}}\right)^2+\left(\frac{\bar{w}}{w_{max}}\right)^4\Big/2$. A crucial aspect here is that the chosen constraint function, is bounded below by -1. This property ensures that $\beta_\delta(-c(\bar{w}))$ is non-negative for all values of $\bar{w}$ and for any δ in the range $(0,1]$. In contrast, constraint (3.8) aimed at avoiding obstacles presents a noteworthy challenge due to its absence of a lower bound. Consequently, it rewards maintaining a significant distance from the obstacle even when the primary goal is simply to avoid it. In order to address this issue, we introduce a saturation step to the argument of $\beta_\delta(\cdot)$ before its application. Specifically, we replace $\beta_\delta(-c(\bar{t},\bar{w}))$ with $\beta_\delta(\sigma(-c(\bar{t},\bar{w})))$. This modification leads us to the concept of the "hockey stick" function [1], as defined below:

$$\sigma(z)=\begin{cases}\tanh(z), & z\geq 0\\ z, & otherwise\end{cases}.$$

This adjustment allows us to effectively disregard obstacles that are sufficiently distant.

3.4 Numerical Computations

In this section, we present numerical computations in order to show the effectiveness of both the proposed avoidance formulation and the proposed algorithm. We start our analysis with the following scenario: the AV is traveling along a straight path and an obstacle is moving with a

low velocity on the right side of the lane. We refer to this scenario as the *lateral dynamic avoidance* case. Then, we consider the *longitudinal dynamic avoidance* case: a moving obstacle cuts-off the ego-vehicle's path. For both the scenarios, the dynamic obstacles are moving with constant velocity and their future positions are known (by simple integrating the velocity). We set the constraints parameters based on [18] and on driving experience, as reported Table 3.1.

Parameter	**Value**	**Units**
Maximum width (w_{max})	1.25	m
Minimum velocity (v_{min})	0.1	m/s
Maximum velocity (v_{max})	19.4	m/s
Minimum acceleration (a_{min})	-1.5	m/s^2
Maximum acceleration (a_{max})	1	m/s^2
Maximum curvature (κ_{max})	0.2	m^{-1}
Maximum lateral acceleration ($a_{lat_{max}}$)	2.0	m/s^2
Safety time (t_{safety})	3	s
Safety distance (d_{safety})	2.5	m

Table 3.1: Set of constraints parameters of the maneuver regulation strategy. The maximum width (w_{max}) parameter specifies the maximum lateral extent of the lane. The velocity constraints are encapsulated by the parameters v_{min} and v_{max}, respectively. Acceleration limits are established through a_{min} and a_{max}, denoting the minimum and maximum accelerations, respectively. The curvature of trajectory paths is addressed by the parameter κ_{max}, quantifying the maximum allowable curvature. Lateral motion is further constrained by the maximum lateral acceleration, $a_{lat_{max}}$. Safety considerations are incorporated through the parameters t_{safety} and d_{safety}, representing the safety time and distance.

3.4.1 Lateral Dynamic Avoidance Maneuver

In the first scenario, the ego-vehicle is traveling with a constant velocity $v_0 = 50\ km/h$ (13.88 m/s) along a straight lane and has to avoid an obstacle moving on the same lane with a constant velocity of 20 km/h (about 5.55 m/s). After a trial and error process, we choose the following cost for the cost function: $q_1 = 0.1$, $q_2 = 0.1$, $q_3 = 1.0$, $q_4 = 0.0$, $r_1 = 100.0$ and, $r_2 = 0.1$. The optimal maneuver is shown in Figures 3.5 and 3.6. Next we highlight some important features of the computed maneuver. Specifically, we identify three phases. First, at the beginning, the ego-vehicle is following the reference path: the lateral displacement $\bar{w}$ is zero (Figure 3.6a), and the velocity $\bar{v}$ is equal to the reference one (Figure 3.6b). Second, the ego-vehicle executes a very smooth avoidance

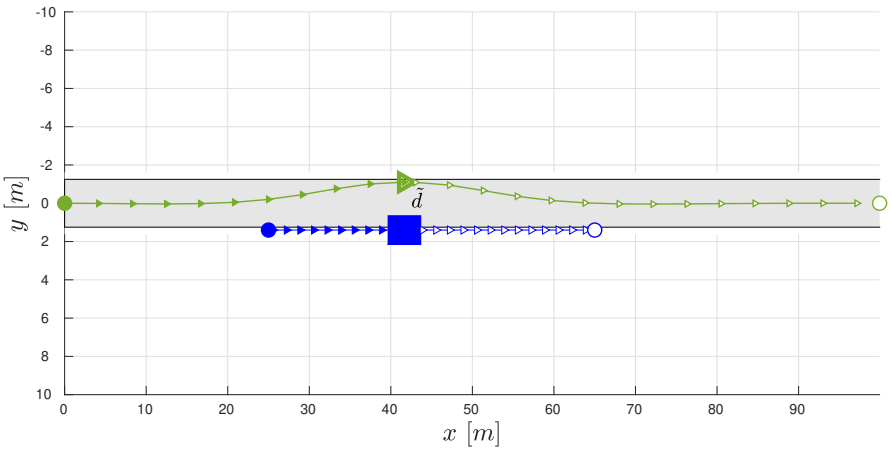


Figure 3.5: Avoidance maneuver. The ego-vehicle (bold green triangle) and the obstacle (blue rectangle) are shown. The ego-vehicle and the obstacle trajectories are indicated with solid triangular green line and solid triangular blue line, respectively.

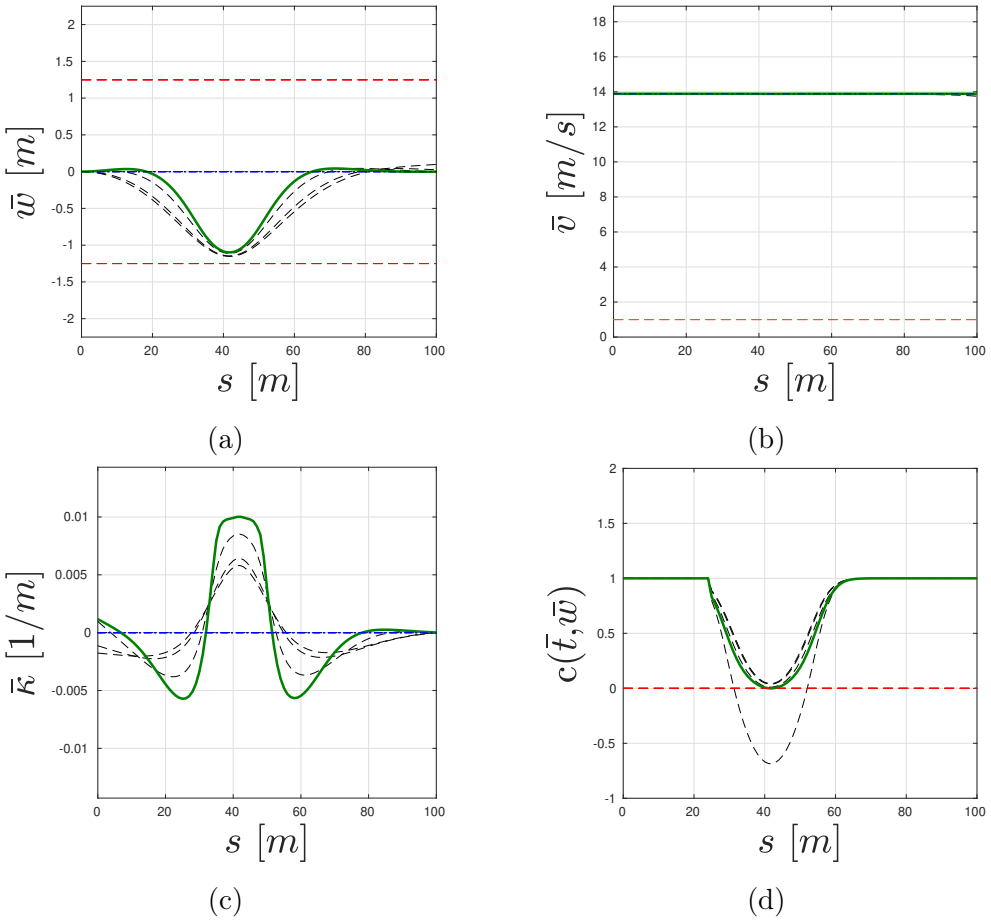


Figure 3.6: Lateral avoidance scenario. The intermediate (dashed black lines) and optimal maneuvers (solid green line) are shown. The desired maneuver is depicted in dash-dotted blue line, while constraints are in dashed red line.

maneuver by applying first a negative curvature, Figure 3.6c, thus moving toward the left boundaries of the lane, Figure 3.6a, to avoid the obstacle. Note that the avoidance constraint, see Figure 3.6d, is active at about $s = 42\ m$ which is the curvilinear coordinate where the ego-vehicle comes

alongside the obstacle, as depicted in Figure 3.5. Third, once the ego-vehicle overtakes the obstacle, it goes back in following the center-line. It is interesting to note that the projection of the initial guess is infeasible for the avoidance constraint. Nevertheless, intermediate trajectories are all feasible, see dashed black lines in Figure 3.6.

3.4.2 Longitudinal Dynamic Avoidance Maneuver

In the second scenario, a moving obstacle cuts off the ego-vehicle's path. Specifically, the ego-vehicle is traveling along a straight path with $v_0 = 13.9\ m/s$ (similarly in Figure 3.5), the moving obstacle pulls out from the right-side of the path, at the longitudinal coordinate of $s = 40\ m$ and at $v_{obs} = 2.8\ m/s$. The cost function weights (3.13) penalize deviation

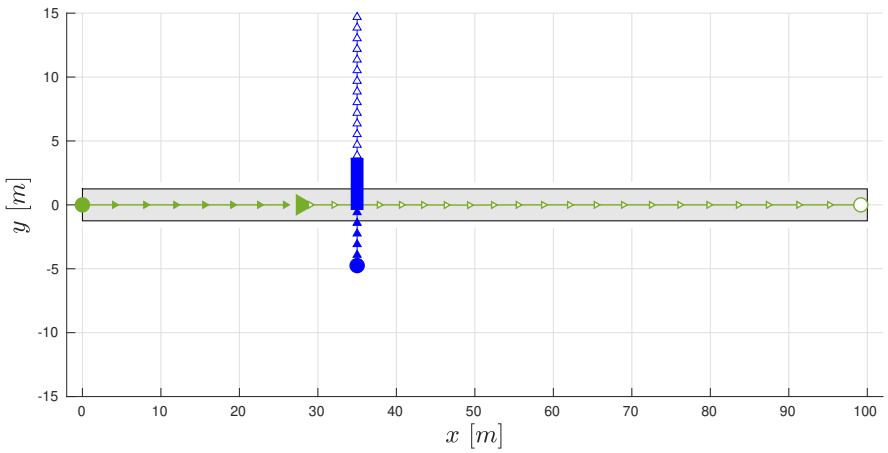


Figure 3.7: Avoidance maneuver. The ego-vehicle (bold green triangle) and the obstacle (blue rectangle) are shown. The ego-vehicle and the obstacle trajectories are indicated with solid triangular green line and solid triangular blue line, respectively.

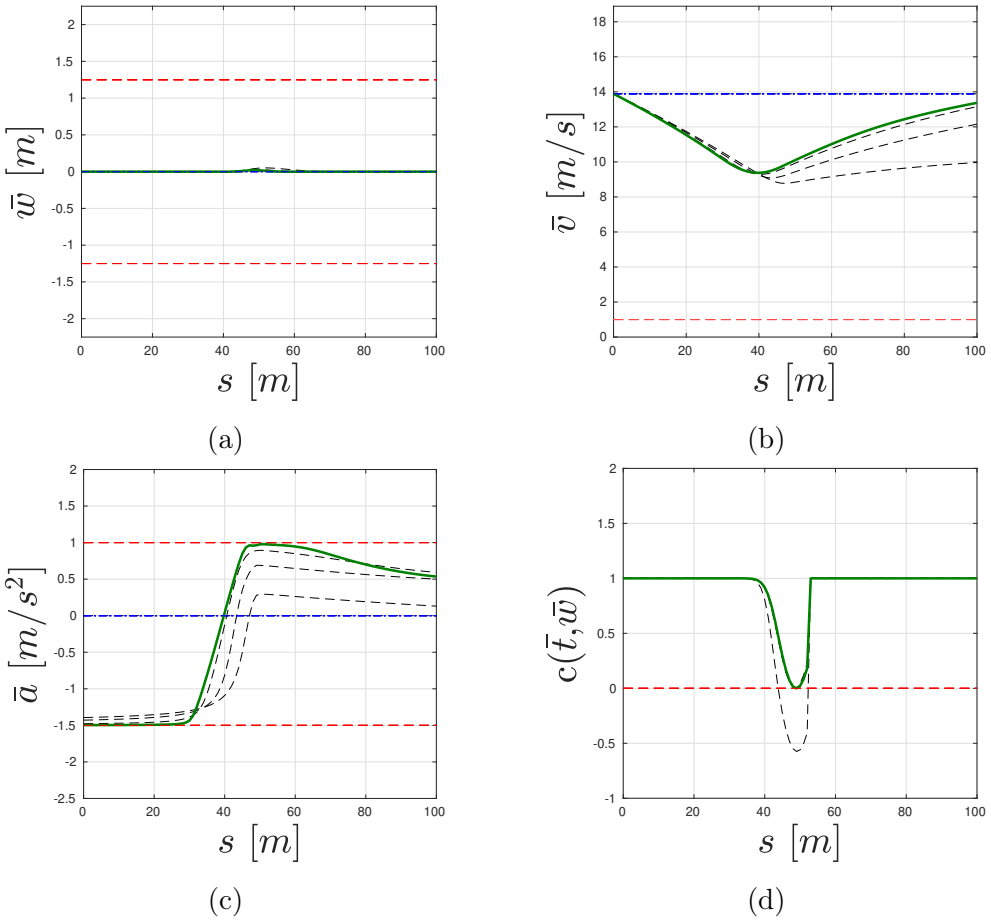


Figure 3.8: Longitudinal avoidance scenario. The intermediate (dashed black lines) and optimal maneuvers (solid green line) are shown. The desired maneuver is depicted in dash-dotted blue line, while constraints are in dashed red line.

from center-line more than deviation to reference velocity: $q_1 = 10.0$, $q_2 = 10.0$, $q_3 = 0.1$, $q_4 = 0.0$, $r_1 = 100.0$ and, $r_2 = 0.1$. The optimal generated maneuver is depicted in Figure 3.7 and 3.8. At beginning of the maneuver, the ego-vehicle applies the minimum longitudinal acceler-

ation, Figure 3.8c, in order to decrease its velocity, Figure 3.8b, and thus to let the obstacle pass. At about $s = 40\ m$, the avoidance constraint becomes active, Figure 3.8d, and the ego-vehicle applies the maximum acceleration to reach the desired velocity. The lateral displacement, Figure 3.8a, is zero, as we expect for the longitudinal avoidance maneuver. Furthermore, we point out that only the initial trajectory is infeasible, see Figure 3.8d. Finally, we highlight that, by using the avoidance constraint (3.7) instead of (3.8), the optimal control problem has no (feasible) solution, thus confirming the importance of the proposed avoidance constraint formulation.

3.5 Validation

In order to validate our collision-free maneuver generation algorithm, we have employed a two-fold approach. First, we integrated it into a high-fidelity autonomous driving simulator. Second, we tested its performance on a real self-driving vehicle. Both the simulator and the self-driving vehicle have been developed by Vislab. Specifically, our real-time maneuver generation algorithm has been embedded into the planning module of the VisLab Autonomous Driving stack. It is worth noting that the accuracy of obstacle predictions is ensured through a motion forecasting module. At each time step, this module provides us with information regarding both the current positions of obstacles and their future predictions. For a more comprehensive understanding of the entire architecture of our autonomous vehicle system, we encourage interested readers to explore the detailed descriptions provided in the references [46] and [47]. The proposed algorithm has been implemented in c++ (in order to integrate the differential equations required by PRONTO, we use integrators based on *boost odeint*, [48]), and it is applied in a receding horizon fashion during

the tests. We use a fixed space horizon of 100 m, a space discretization of 1 m, and updated our planning every 100 ms. The optimal collision-free maneuvers generated by our algorithm are used as reference trajectories for the low-level controller.

3.5.1 Simulation Results

The output of the low-lever controller serves as input to a simulated dynamic vehicle model. This model, elaborated upon in Chapter 1.4, encompasses various subsystems typically found in a vehicle, such as the chassis and both front and rear tires. It is important to note that the vehicle chassis has 6 degrees of freedom (DOF), while each wheel has 1 DOF. Furthermore, the steering and torque systems of the vehicle are related to acceleration and curvature through lookup tables, while a Pacejka magic formula tire model is used to accurately model the tire-ground interaction, see Section 1.4.1.

We conducted the simulated test in a simplified urban scenario featuring a static parked car and a dynamic obstacle, specifically a bicycle, see Figure 3.9. We invite the reader to watch the video[1] attachment corresponding to the discussed simulated test.

In Figure 3.10, we compare the lateral displacement, yaw-rate, and velocity of the simulated dynamic vehicle and the ones generated by our algorithm at the actual longitudinal coordinate s $[m]$. Next, we highlight two maneuvers generated by the proposed algorithm and successfully executed by the simulated vehicle.

First, at the beginning of the simulated test, the ego-vehicle starts with zero velocity and accelerates smoothly in order to reach the de-

[1] https://drive.google.com/file/d/1EK8g-CUaJeo_iapLcvaWJPi-E7EnpXA8/view?usp=sharing

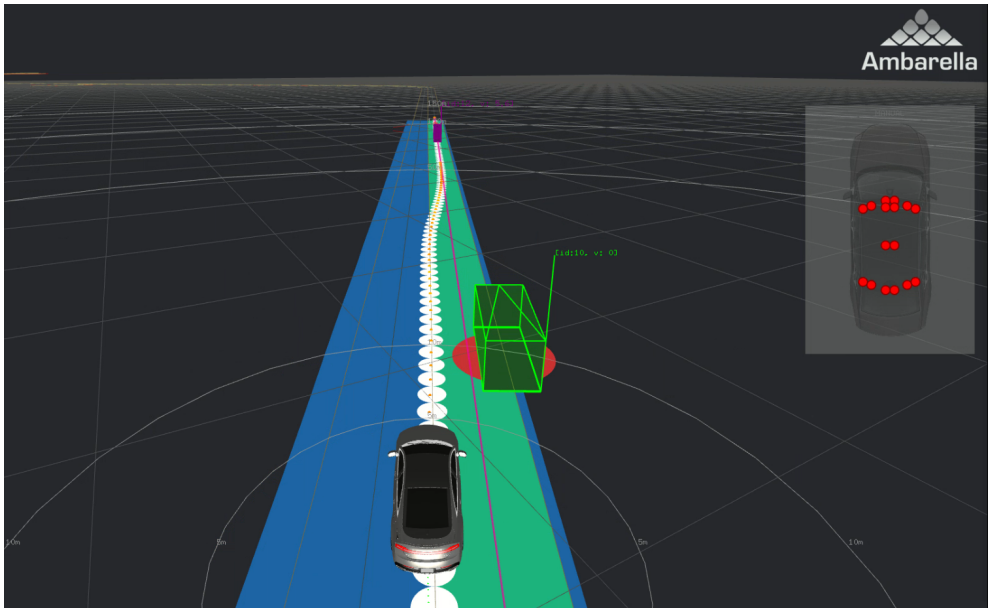
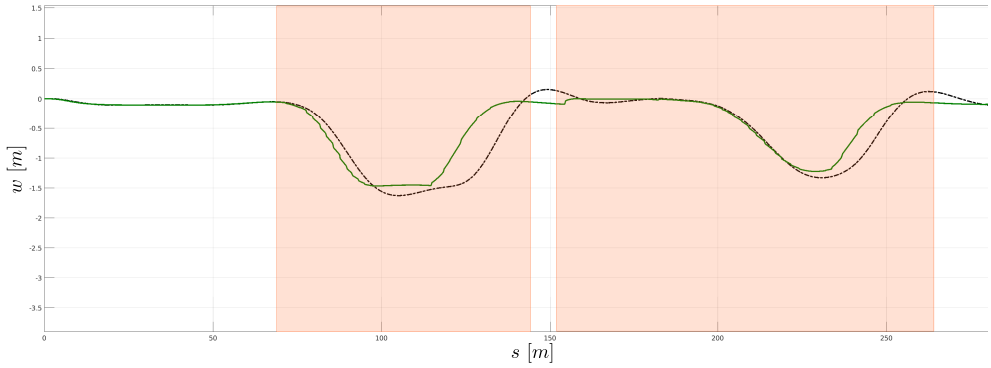
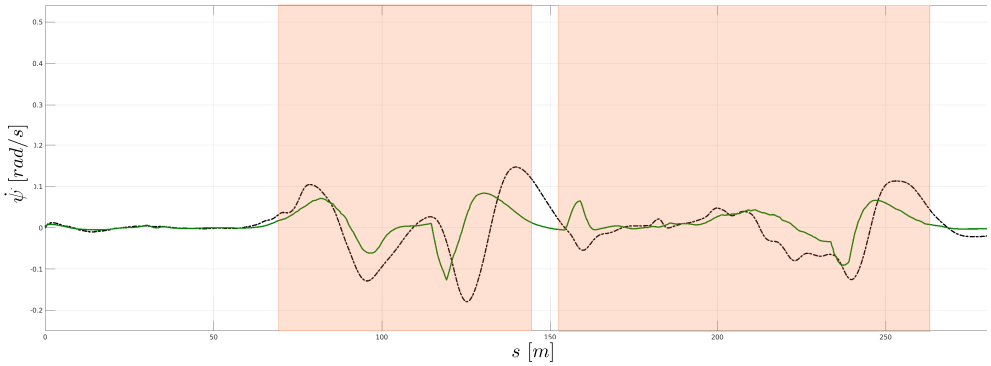


Figure 3.9: Simulation scenario. The ego-vehicle (gray car) shares the road with a static obstacle (green rectangle), and a dynamic obstacle (purple rectangle). The generated trajectory by the maneuver regulation algorithm, is visually indicated by a series of white dots. The road boundaries are indicated by the green corridor.

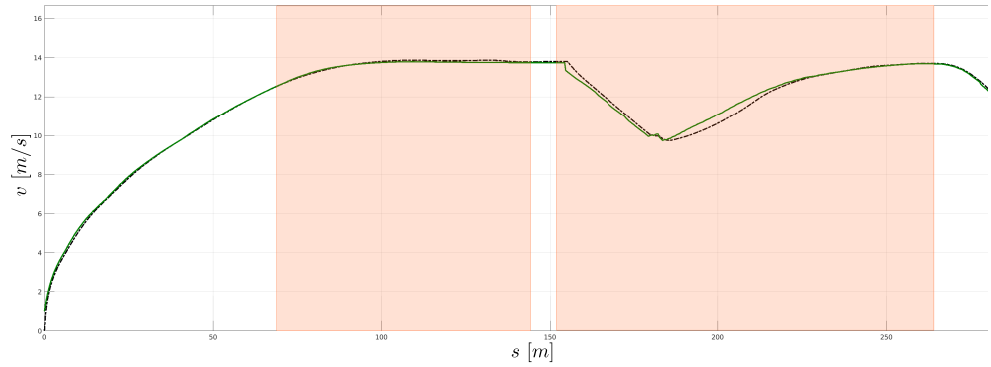
sired velocity of 13.9 m/s. Along the straight lane, a static car parked on the right side of the lane is detected. As expected, the ego-vehicle avoids the static car in a smooth fashion as shown in the first highlighted section of Figure 3.10. It is worth noting that, during this dynamic avoidance maneuver, the constraint (3.8) is always satisfied, and the executed maneuver is very smooth as shown in the first highlighted section of Figure 3.10. Second, after about 160 m, a bicycle moving along the center of the lane, crosses the road and riding near the right-side of the lane. The ego-vehicle first decreases its velocity, then moves on the left side of the



(a)



(b)



(c)

Figure 3.10: Comparison between the reference maneuver (green solid line) generated by the proposed algorithm and actual maneuver of the simulated dynamic vehicle (black dashed line).

lane (thus avoiding the moving bicycle), and finally merge back to its own lane. Again, the safety distance imposed by the avoidance constraints is satisfied and the ego-vehicle's velocity matches the one generate by the algorithm.

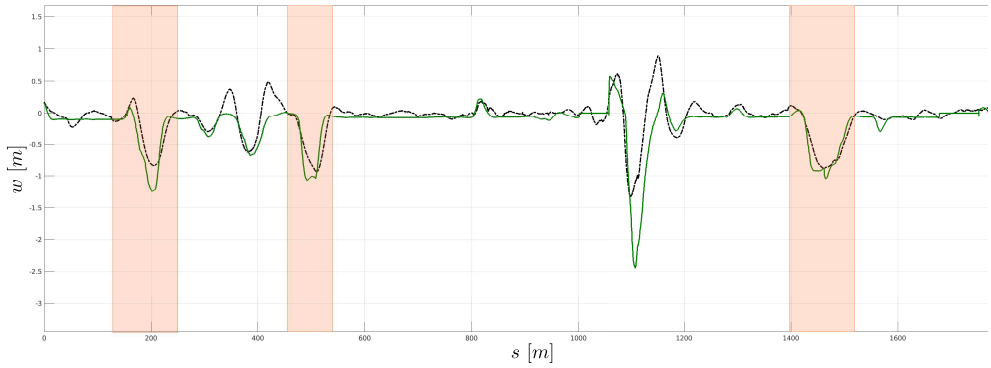
3.5.2 Experimental Results

The experimental test has been carried out in Parma, Italy, on the campus area (urban roads) open to regular traffic. The main objective of the test is to demonstrate the efficacy of the proposed algorithm in generating real-time feasible maneuvers. We invite the reader to watch the video[2] attachment corresponding to the discussed test.

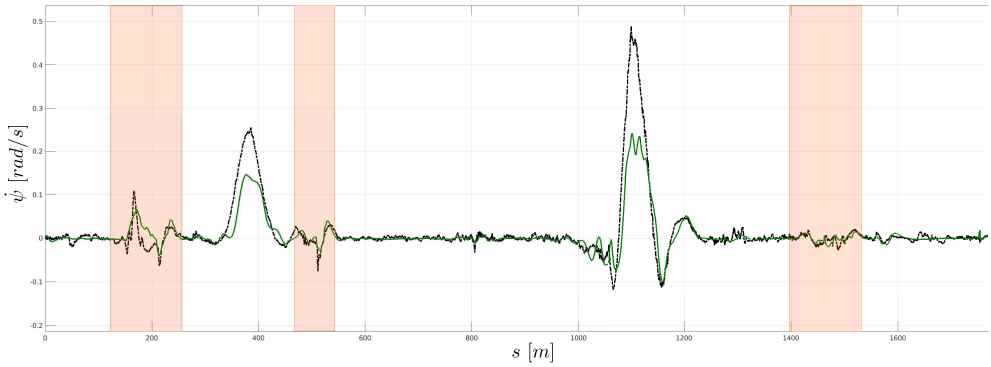
In Figure 3.11, we compare the actual lateral displacement, yaw-rate, and velocity of the actual ego-vehicle and the ones generated by our algorithm at the actual longitudinal coordinate s $[m]$. Next, we highlight three maneuvers generated by the proposed algorithm and successfully executed by the ego-vehicle.

First, at the beginning of the test, the ego-vehicle starts with zero velocity and accelerates smoothly (i.e., the acceleration constraint (3.11) is not active) in order to reach the desired velocity of 13.9 m/s. Along the straight lane, a bicycle (coming from a side road) crosses the road and starts riding on the right-side of the lane. As soon as the bicycle is detected, the ego-vehicle first decreases its velocity, then moves on the left side of the lane (thus avoiding the moving bicycle), and finally merge back to its own lane. It is worth noting that, during this dynamic avoidance maneuver, the constraint (3.8) is always satisfied, and the executed maneuver is very smooth as shown in the first highlighted section of Figure 3.11. Second, after about 300 m, a static car parked along the

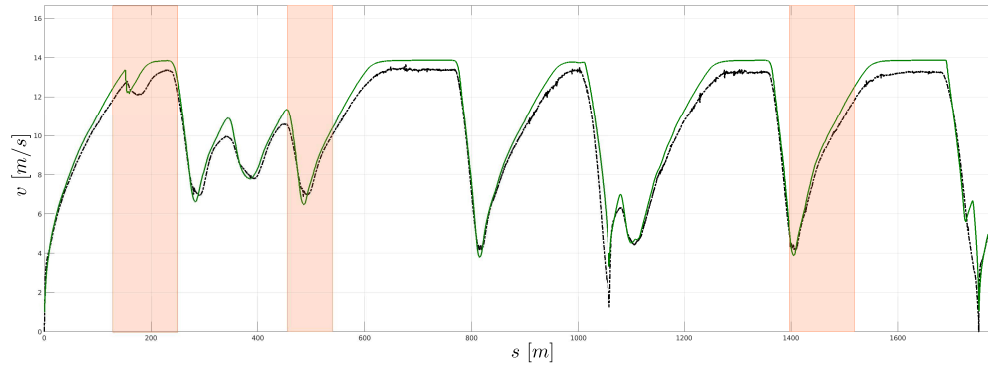
[2] `https://youtu.be/x1glAcRP1TM`



(a)



(b)



(c)

Figure 3.11: Comparison between the reference maneuver (green solid line) generated by the proposed algorithm and actual maneuver of the autonomous vehicle (black dashed line).

right side of the lane is detected. As expected, the ego-vehicle avoids the static car in a smooth fashion as shown in the second highlighted section of Figure 3.11. Again, the safety distance imposed by the avoidance constraints is satisfied and the ego-vehicle's velocity matches the one generated by the algorithm. Third, in the last highlighted section of Figure 3.11, the ego-vehicle needs to avoid six parked cars and a pedestrian. Similarly to the previous case, the ego-vehicle successfully avoid the obstacles by tracking the generated maneuver.

Overall, we observe a good trajectory matching, even though some differences can be noticed. First, when the vehicle is driving in the roundabout (thus making a U-turn, refer to the video attachment), the ego-vehicle's position slightly deviates from the generated path, see Figure 3.11a at about $s = 1100\ m$, and the actual yaw-rate differs from the generate one, see Figure 3.11b. Since the low-level controller (which is out of the scope of this work) is designed in order to penalize maneuvers close to the lane boundaries, the actual lateral displacement becomes lower than the reference one and the actual yaw rate becomes greater than the reference one (notice that the velocity is well-matched in this segment of the road). A similar behavior can be observed when the ego vehicle is executing a 90-degree turn (see Figure 3.11 at about $s = 330\ m$). Second, we observe that the actual velocity does not perfectly match the generated one in steady-state, see Figure 3.11c for $v = 13.9\ m/s$. Based on the generated velocity and acceleration profiles, the low-level controller generates the throttle (or braking) commands. However, it does not take into account aerodynamic and resistance forces and, consequently, the actual velocity becomes slightly lower than the generated one. In both cases, the actual trajectory satisfies the constraints and well-matches the generated one. Finally, we provide the CPU time needed for the execution of the proposed algorithm. Specifically, the average computation time of

the proposed algorithm in the case of no obstacles is 36 ms. For the avoidance scenarios (i.e., for the three highlighted zones in Figure 3.11) the average computation time is 54 ms.

3.6 Discussion

These results confirm that the proposed trajectory generation algorithm allows to compute feasible collision-free maneuvers with a computation times below 100 ms, thus enabling a real-time implementation at 10 Hz. We highlight that the approaches most closely related to our problem setup and constraints formulation are the ones proposed in [37], [9], and [4]. Next, we highlight the main differences. In [37], a maneuver regulation perspective is adopted for generating collision-free trajectories. The independent variable of the optimization problem is the time and a suitable approximation (based on Bezier curves) is introduced in order to describe the road geometry (i.e., centerline of the road, road boundaries, target velocity). In contrast to the previous approach, we tackle the optimization problem using the longitudinal coordinate as the independent variable and thus we do not resort to approximations for the (reformulated) vehicle system. In [9] and [4], the model dynamics is described using the curvilinear coordinate. In order to generate obstacle collision-free trajectory, the road-boundaries (which are functions of the curvilinear coordinate) are re-shaped by taking into account the obstacles' position and size. However, such a constraint formulation becomes too conservative when dealing with moving obstacles, mainly because the time variable is neglected. On the contrary, we embed the time variable into vehicle dynamics and propose the collision avoidance constraint with respect to the new set of variables, thus handling both static and dynamic obstacles.

Chapter 4

Upper-level Policy Search and MPC for Lane Change

In this chapter we propose a hybrid approach that combine reinforcement learning policy search with model predict control framework to generate lane change maneuvers.

4.1 Introduction

Lane change maneuver has gained significant attention within the research community, as indicated by a growing body of literature (see, e.g. [49, 50, 51]). Executing a successful lane change is particularly challenging due to the requirement for generating collision-free trajectories. Moreover, determining the “right" time to perform a lane change to avoid

other moving vehicles adds an additional layer of complexity to the design of decision-making and planning schemes. Typical approaches to address this challenge have often decoupled decision-making and planning tasks. The most common solution employs rule-based lane change models for decision-making and then integrates the decision logic into a trajectory planning module.

In rule-based models, AVs determine lane change decisions based on predefined rules, such as lane preference or the feasibility of the maneuver (see, e.g., [52]). However, a notable limitation of such models is their performance bottleneck. They may lack accuracy and fail to capture all relevant influencing factors when applied only through threshold-based conditions to assess driving intentions. Other approaches address the problem of finding the optimal time to execute a lane change as a classification problem. In this context, common classification algorithms are based on machine learning techniques such as, e.g., Support Vector Machines (SVM), Bayesian classifiers, and decision trees. In [53], a data-driven approach is used to “mimic" human driver behavior during lane changes. Real-world data are collected in typical lane change scenarios and a SVM classifiers is employed in order to predict when a lane change should be initiated based on a specific driver's preferences. However, feature selection and transformation are required to achieve good performance.

After the AV has determined its driving intention, generating a trajectory becomes crucial for ensuring safety before executing it. Among various control techniques available, Model Predictive Control (MPC) has gained popularity for trajectory generation due to its capacity to handle nonlinear dynamics and state-input constraints, see e.g. [1, 6]. For instance, in [54] an MPC-based lane change algorithm is proposed to integrate the path planning and path following layers together with a

utility function, which helps to automatically determine the target lane. However, the MPC closed-loop performance is sensitive to the design choices of this heuristic decision function. As a result, a series of approximations are employed and may produce conservative solutions. On the other hand, Reinforcement Learning (RL), in particular policy search approaches [55], has recently emerged as an innovative method for learning driving policies, as demonstrated in [56, 57]. The key concept behind RL involves training a policy through iterative trial and error, with the aim of maximizing a performance function, namely the “return". This approach directly translates sensor inputs into actuation commands for the AV. However, it is important to note that RL-based methods are characterized by several challenges. First, it exhibits a high degree of data inefficiency, requiring large amounts of training data to achieve satisfactory results. Second, RL models often suffer from poor generalization, struggling to adapt to unseen situations effectively. Furthermore, one critical concern is the limited safety and stability guarantees offered by RL methods, which can be particularly problematic when applied to AVs.

Various approaches have been explored to integrate learning and control in autonomous systems. For instance, in [58], a sampling-based MPC approach was developed for autonomous driving, with a particular focus on the obstacle avoidance task. However, a notable challenge arises from the need to generate a large amount of samples in real-time, often achieved through parallel processing, which is computationally expensive. In [59] an approach was introduced that integrates machine learning and MPC within an imitation learning framework, applied to lane-keeping maneuvers. Nevertheless, this method involves training deep neural network policies through supervised learning, which necessitates the availability of ground-truth labels, potentially posing limitations in certain scenarios.

The next part of the chapter is dedicated to the development of the hybrid MPC strategy to efficiently deciding when to initiate lane change maneuvers and generate collision-free trajectories.

In Section 4.2, we describe the system dynamics and formulate the lane change problem as a parametric MPC problem. The parameter within this framework determines whether to remain in the current lane or transition to an adjacent one. In Section 4.3, we address the challenge of finding the parameter to execute the lane change maneuver by formulating it as a probabilistic policy search problem. Our approach employs a weighted maximum likelihood method for learning the policy parameter, offering a closed-form solution for policy updates. Then, inspired by the work outlined in [60, 61], which focused on maneuvering a quadrotor through the center of a rapidly moving gate, we employ a self-supervised learning approach. This enables to generate online the parameter to generate the maneuver based on observations of the surrounding environment. This strategy is evaluated through numerical computations and illustrated in Section 4.4.

4.2 Problem Formulation

Let us consider the scenario depicted in Figure 4.1. The road is composed of two parallel lanes, called the "*ego lane*" and the "*target lane*". The AV (from now on, referred as the ego-vehicle) is traveling along the ego lane, while a *front vehicle*, indicated as the FV, is moving on the same lane. We assume that the FV is traveling slower than the ego-vehicle, which motivates a lane change maneuver. In such a scenario, we are interested in generating a lane change maneuver by designing an MPC with an upper-level decision variable. This variable selects the "right time" to execute the maneuver while avoiding the "*lateral vehicle*", indicated as

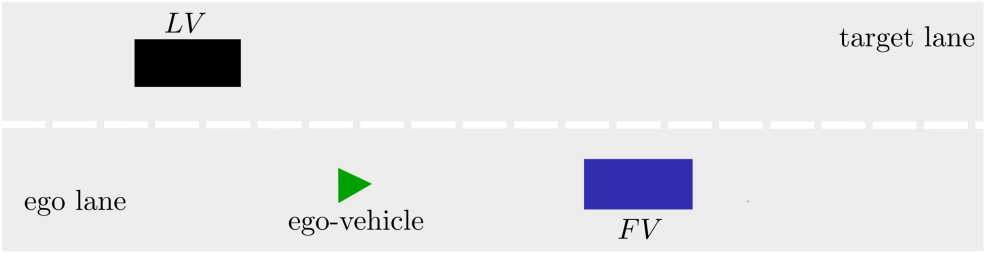


Figure 4.1: 2D representation of lane change scenario.

the LV, moving on the target lane.

4.2.1 Ego-Vehicle Motion

For the given scenario, we describe the 2D motion of a car-like vehicle using the set of equations describe in Chapter 1.3 and recalled below:

$$\begin{aligned} \dot{x} &= v\cos\psi\,, \\ \dot{y} &= v\sin\psi\,, \\ \dot{\psi} &= v\kappa\,, \\ \dot{v} &= a\,, \end{aligned} \tag{4.1}$$

where the variables (x, y) represent the longitudinal and lateral coordinates in the inertial frame, ψ denotes the heading angle, and v represents the velocity. Control over the ego-vehicle's motion is achieved by manipulating the curvature κ and the acceleration a. We recall also that this simplified model is particularly suitable for scenarios with relatively low acceleration inputs and closely approximates the behavior of more complex dynamic models, as detailed in [18].

Next, we re-write (4.1) in terms of the longitudinal and lateral coordinates (s, w), where s represents the position along the center-line of the

ego lane, while w indicates the displacement transverse to this center-line. This coordinate system is particularly useful when considering lane changes, where it is more intuitive to think in terms of lateral distance from the desired lane position, rather than Cartesian coordinates (x, y).

We assume that the ego lane has a reasonably smooth (at least C^2) arc-length parametrized center-line, denoted as $(\bar{x}_{cl}(s), \bar{y}_{cl}(s))$. Following the derivation in Section 1.3.1 we can write (4.1) in (s, w) coordinates system as follows:

$$\begin{aligned} \dot{s} &= \frac{v \cos \mu}{1 - w\bar{\kappa}_{cl}(s)}\,, \\ \dot{w} &= v \sin \mu\,, \\ \dot{\mu} &= v\kappa - \bar{\kappa}_{cl}(s)\dot{s}\,, \\ \dot{v} &= a\,, \end{aligned} \tag{4.2}$$

where $\mu = \psi - \bar{\psi}_{cl}$ represents the local heading error. In equation (4.2) the state and the control vector are denotes as $\boldsymbol{x} = [s, w, \mu, v]$ and $\boldsymbol{u} = [\kappa, a]$, respectively.

4.2.2 Model Predictive Control Formulation

In order to address the lane change problem, we set up a parametric nonlinear MPC problem. In this section, we define state-input constraints and formulate the cost function for optimization.

First, we define constraints. The ego-vehicle is required to satisfy the road boundaries, which, in the new coordinate system, take a simple form:

$$w_{min} \leq w \leq w_{max}\,. \tag{4.3}$$

Additionally, we account for the operational limits of the kinematics model and the passenger comfort by imposing state and input constraints

on (4.2) as follows. The velocity is bounded by two constants, i.e.,

$$v_{min} \leq v \leq v_{max}\,, \tag{4.4}$$

while the longitudinal acceleration is bounded as follows,

$$a_{min} \leq a \leq a_{max}\,. \tag{4.5}$$

Moreover, in order to take into account the limited wheel steer angle, the curvature is bounded in module as follows,

$$|\kappa| \leq \kappa_{max}\,. \tag{4.6}$$

We design the cost function J as follows. The ego-vehicle is supposed to travel on the ego lane. In order to execute the lane change maneuver, it needs to minimize the distance from the target lane. Hence, the ego-vehicle needs to stop following the ego lane and start following the target lane. Thus, we design the cost function as a sum of three components as follows:

$$J(\mathbf{x}, \mathbf{u}) = J_{el}(\mathbf{x}, \mathbf{u}) + J_{tl}(\mathbf{x}, \mathbf{u}) + J_u(\mathbf{u})\,, \tag{4.7}$$

where J_{el} penalizes deviance from the ego lane:

$$J_{el} = (1 - \gamma(\theta))(q_1 s^2 + q_2 w^2 + q_3(v - v^{el})^2)\,;$$

J_{tl} penalizes the deviance from the target lane:

$$J_{tl} = \gamma(\theta)(q_4 s^2 + q_5(w - w^{tl})^2 + q_6(v - v^{tl})^2)\,;$$

and J_u penalizes the control effort:

$$J_u = r_1 \kappa^2 + r_2 a^2\,.$$

The time-varying switch term $\gamma(\theta)$ is defined as:

$$\gamma(\theta) = \frac{1}{1 + \exp(\alpha(t - \theta))}$$

where $\alpha \in \mathbb{R}^+$ controls the temporal spread, and θ determines the time to execute the lane change. For $t \leq \theta$, $\gamma \approx 0$, indicating the ego-vehicle should follow the ego lane, while for $t > \theta$, $\gamma \approx 1$, which implies following the target lane.

We are ready to formulate the $\text{MPC}_\theta(\boldsymbol{x}, \boldsymbol{u})$ problem:

$$\begin{aligned} \min_{\mathbf{x}(\cdot)\,\mathbf{u}(\cdot)} & \int_0^{t_f} J(\mathbf{x}(\tau), \mathbf{u}(\tau), \theta)\, d\tau + m(\mathbf{x}(t_f)) \\ \text{s.t.} \quad & \mathbf{x}'(t) = f(\mathbf{x}(t), \mathbf{u}(t))\,, \, \mathbf{x}(0) = \mathbf{x}_0 \\ & h(\mathbf{x}(t), \mathbf{u}(t)) \leq 0\,, \end{aligned} \tag{4.8}$$

where $t_f > 0$ is the time horizon, $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ represents the nonlinear dynamics (4.2), $h(\mathbf{x}, \mathbf{u})$ denotes state and input constraints (4.3), (4.4), (4.5), (4.6), $J(\mathbf{x}, \mathbf{u}, \theta)$ is the stage cost as in (4.7), and $m(\mathbf{x}(s_f))$ minimizes the L2 distance between the ego-vehicle state and the desired end state $\mathbf{x}(t_f)$.

It is important to highlight that a key requirement to solve the problem is to determine optimal θ in advance. We determine θ through a probabilistic policy search approach, which will be detailed in the following section.

4.3 Upper-level Policy Learning

We formalize the problem of finding parameter θ by introducing the concept of learning an upper-level policy $\pi_{\boldsymbol{\omega}}(\theta)$ responsible for selecting the parameter of the $\text{MPC}_\theta(\boldsymbol{x}, \boldsymbol{u})$ policy. This selection process can be

captured by modeling $\pi_{\boldsymbol{\omega}}(\theta)$ as a Gaussian distribution:

$$\pi_{\boldsymbol{\omega}}(\theta) = \mathcal{N}(\theta|\boldsymbol{\omega})\,,$$

where $\boldsymbol{\omega} = [\chi, \sigma^2]$ represent the mean χ and the variance σ^2 of the distribution. In order to address the upper-level policy search problem, we approach it as a probabilistic inference problem, see Figure 4.2. To this end, we introduce the "return event" as the observable variable. The probability of observing the return event is expressed as $p(R = 1|\theta) = p(R|\theta)$. Our objective is to find the optimal $\boldsymbol{\omega}$ that maximizes the probability of this return event. In other words, we aim to solve the following maximum likelihood problem:

$$\max_{\boldsymbol{\omega}} \ \log p_{\boldsymbol{\omega}}(R) = \log \int_{\theta} p(R|\theta)\pi_{\boldsymbol{\omega}}(\theta)\ d\theta\,. \tag{4.9}$$

We solve (4.9) by using the Monte Carlo Expectation Maximization (MC-EM) algorithm [53, 62], a well-known technique for finding maximum likelihood solutions. Before introducing the algorithm, we need to define the return R. Unlike the cost function used in (4.8), the design of R offers more flexibility. Next, we specify the different contributions that concur to define the task of performing a lane change when necessary.

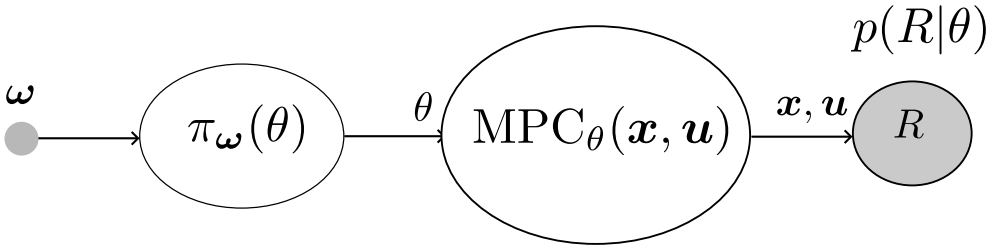


Figure 4.2: Graphical representation of upper-lever policy search as a probabilistic inference problem.

First, the ego-vehicle needs to avoid collision with obstacles. In order to penalize collisions, we employ a trajectory evaluation mechanism in which we assign negative rewards to collision states. This evaluation is expressed as follows:

$$N_c(\boldsymbol{x}) = \begin{cases} -p_1, & \text{if } c(\mathbf{x}, \mathbf{x}^{obs}) > 0,\ obs \in \{FV, LV\} \\ 0, & \text{otherwise} \end{cases}, \tag{4.10}$$

where p_1 is a positive constant that weighs the penalty. By assigning negative values to collision states, the ego-vehicle is effectively discouraged from entering such states and encouraged to prioritize safe trajectories that avoid collisions. On the other hand, non-collision trajectories are assigned zero values, indicating their neutrality in terms of collision risk. In (4.10), the function $c(\mathbf{x}, \mathbf{x}^{obs})$ evaluates whether a collision occurs between the ego-vehicle and the obstacles. This function is defined by considering an ellipse centered at the axes of the obstacle relative to the new set of coordinates (s, w). Specifically, given the positions and future predictions of the obstacles, $\mathbf{x}^{obs} = [s^{obs}, w^{obs}]$, we evaluate the collision avoidance as follows,

$$c(\mathbf{x}, \mathbf{x}_o^{obs}) = -1 + \left(\frac{s - s^{obs}}{\bar{s}}\right)^2 + \left(\frac{w - w^{obs}}{\bar{w}}\right)^2. \tag{4.11}$$

In this formulation, we leverage the properties of the ellipse to determine if the ego-vehicle's current state, represented by (s, w), falls within its boundary. If (s, w) lies inside the ellipse, it indicates a collision between the ego-vehicle and the obstacle, triggering a collision evaluation function with an appropriate outcome.

Second, in order to discourage unnecessary lane changes, we introduce a contribution penalizing trajectories in which a change in the lateral coordinate w occurs. The formulation of this contribution is presented

below:

$$N_{lc}(\boldsymbol{x}) = \begin{cases} -p_2, & \text{if } |w(t) - w(t+1)| > 0 \\ 0, & \text{otherwise} \end{cases}, \tag{4.12}$$

where p_2 is a positive constant. If the absolute difference in lateral deviation is greater than zero, it indicates a lane change and a penalty of p_2 is assigned. This penalty discourages such lane changes that do not contribute to the ego-vehicle's overall objective or safety. By incorporating this term into the return function, we guide the ego-vehicle's decision-making process to reduce unnecessary lane changes.

Third, in order to discourage trajectories that anticipate changing the lane to avoid the FL, we penalize lateral coordinates w that deviate from the ego lane center-line. The lateral displacement penalty is defined as:

$$N_w(\boldsymbol{x}) = \begin{cases} -p_3, & \text{if } w < 0 \\ 0, & \text{otherwise} \end{cases}, \tag{4.13}$$

where p_3 is a positive constant. This contribution encourages trajectories that stay closer to the ego-lane center-line while allowing a smooth increase in penalty as the lateral displacement deviates from the center-line.

Finally, we can formulate our return R as the sum of the previously presented contributions (4.10),(4.12),(4.13), resulting in the following expression:

$$R(\boldsymbol{x}) = \int_0^{t_f} \left(N_c(\boldsymbol{x}) + N_{lc}(\boldsymbol{x}) + N_w(\boldsymbol{x})\right) dt\,. \tag{4.14}$$

Maximizing this return guides the upper-level policy $\pi_{\boldsymbol{\omega}}(\theta)$ to select parameter θ that enable the ego-vehicle to successfully avoid the FV and the LV by performing lane changes when necessary.

We are now ready to describe the MC-EM algorithm. Similar to the standard EM algorithm, we begin by decomposing (4.9) into two terms

with the introduction of a variational distribution $q(\theta)$:

$$\log p_{\boldsymbol{\omega}}(R) = \mathcal{L}_{\boldsymbol{\omega}}(q(\theta)) + KL(q(\theta)||\pi_{\boldsymbol{\omega}}(\theta))\,,$$

where with $KL(\cdot)$ we denote the Kullback-Leibler divergence. Since the KL-divergence is always larger or equal to zero, the term $\mathcal{L}_{\boldsymbol{\omega}}(q(\theta))$ is a lower bound of the log marginal-likelihood $\log p_{\boldsymbol{\omega}}(R)$. The two update steps in EM correspond to maximizing the lower bound $\mathcal{L}$ and minimizing the KL-divergence term. In the MC version of EM algorithm we use a sample-based approximation for the variational distribution $q(\theta)$, i.e., in the E-step, we minimizes the KL-divergence by using samples

$$\theta_i \sim \pi(\theta_i|\boldsymbol{\omega}_k)\,.$$

Then, these samples θ_i are used in the M-step to estimate the expectation of the complete data log-likelihood by maximizing the following objective:

$$\boldsymbol{\omega}_{k+1} = \arg\max_{\boldsymbol{\omega}} \sum_i w_i \log \pi(\theta_i|\boldsymbol{\omega}_k)\,, \tag{4.15}$$

where $w_i = f(R_i)$ are the wights. In order to transform the return R into an improper probability distribution, we employ the exponential transformation [63]:

$$w_i = \exp(\beta_k(R_i - \max \boldsymbol{R}_k))\,,\,.$$

The parameter β serves as the “temperature" of the distribution and can be determined using the following heuristic [64]:

$$\beta_k = \frac{\beta_0}{\max \boldsymbol{R}_k - \min \boldsymbol{R}_k}\,.$$

It is worth noting that higher values of higher values of β lead to more greedy policy updates. The MC-EM algorithm iteratively refines the

upper-level policy, until convergence is achieved. Convergence is typically indicated when parameter estimates stabilize. The MC-EM algorithm has a closed-form solution for a Gaussian policy. Algorithm 2 gives a pseudocode description of the strategy.

Algorithm 2 Upper-level Policy Learning

Input: initial policy parameters $\boldsymbol{\omega}_0$,
initial temperature parameter β_0,
initial traj $(\boldsymbol{x}_0, \boldsymbol{u}_0)$, $k = 0$
repeat
for $i = 1$ to I **do**
Sample θ_i from $\pi(\theta_i|\boldsymbol{\omega}_k)$
Solve $(\boldsymbol{x}, \boldsymbol{u})_i = \text{MPC}_{\theta_i}(\boldsymbol{x}_0, \boldsymbol{u}_0)$
Evaluate $R_i = R(\boldsymbol{x}_i)$
end for
Construct $\boldsymbol{R}_k = [R_1, ..., R_N]$
Calculate weights:
$w_i = \exp(\beta_k(R_i - \max \boldsymbol{R}_k))$, $i = 1$ to I ▷ E-step
Maximize objective:
$\boldsymbol{\omega}_{k+1} \leftarrow \arg\max_{\boldsymbol{\omega}} \sum_i w_i \log \pi(\theta_i|\boldsymbol{\omega}_k)$ ▷ M-step
Update temperature: $\beta_{k+1} \leftarrow \frac{\beta_0}{\max \boldsymbol{R}_k - \min \boldsymbol{R}_k}$
until Convergence criteria met
Output: Optimized policy parameters $\boldsymbol{\omega}$

4.3.1 Deep Upper-Level Policy

In order to address dynamic scenarios where the environment rapidly change, we combine the MC-EM algorithm with a self-supervised learning approach to adaptively select the parameter θ based on the environment's

observation.

First, we characterize the observation vector. At any given time t, the observation vector, $\mathbf{o}_t$, captures relevant information about the ego-vehicle and surrounding obstacles. Specifically, it includes the ego-vehicle's position and velocity, $\mathbf{o}_t^{ego} = [s, e_y, e_\psi, v]_t$, as well as the positions and velocities of obstacles, $\mathbf{o}_t^{obs} = [s, e_y, e_\psi, v]_t^{obs}$. Formally, the observation $\mathbf{o}_t$ is defined as:

$$\mathbf{o}_t = [\mathbf{o}_t^{ego} - \mathbf{o}_t^{FV}, \mathbf{o}_t^{ego} - \mathbf{o}_t^{LV}] .$$

Second, we collect a dataset $\boldsymbol{D}$ by simulating various scenarios as follows. Each scenario begins with random initial states for the ego-vehicle and the two obstacles. This randomization is crucial as it allows us to explore a wide range of possible situations. Then, we record the observation vector $\boldsymbol{o}_t$. In order to identify the optimal time for executing a lane change, we employ the MC-EM algorithm, which finds the optimal parameter θ_t^*. With the optimal parameter θ_t^* in hand, we solve (4.8), yielding the first optimal control input, applied to the ego-vehicle. After executing the first optimal control input, we record the next observation $\boldsymbol{o}_{t+1}$. We repeat this process until either the scenario reaches its maximum simulation steps or no collision-free lane change maneuver is found. This systematic approach constructs our dataset $\boldsymbol{D}$, which includes observations and corresponding optimal parameters ($\boldsymbol{o}_t$, θ_t^*) from different scenarios.

Third, we use dataset $\boldsymbol{D}$ to train a general-purpose neural network, denoted as $f_{\boldsymbol{\phi}}$, with $\boldsymbol{\phi}$ representing the network's parameters. We optimize $\boldsymbol{\phi}$ using the following Mean Squared Error (MSE) loss function:

$$\arg\min_{\boldsymbol{\phi}} |f_{\boldsymbol{\phi}}(\boldsymbol{o}_t) - \theta_t^*|^2$$

which minimizes the difference between the neural network's predictions, $f_{\boldsymbol{\phi}}(\boldsymbol{o}_t)$, and the optimal parameter θ_t^* for a given observation $\boldsymbol{o}_t$.

Finally, once the neural network is trained, it can be employed online in the inference phase to handle unseen situations. Given the current observations, the model predicts the optimal time to execute a lane change. It is important to note that the MPC control policy (4.8), used to construct dataset $\boldsymbol{D}$, do not take into account explicitly avoidance constraints. As a consequence, the neural network is trained to provide an upper-level parameter θ even when generating a collision-free lane change is impossible. In order to ensure collision-free maneuvers in the inference phase, the (4.8) is augmented with avoidance constraints (4.11), providing safety if the lane change maneuver cannot be executed.

4.4 Numerical Computations

In this section, we present numerical computations that demonstrate the effectiveness of the proposed approach.

First, we consider a setup where all vehicles involved start from fixed initial conditions. Given the vehicle's dynamic, we want to plan a trajectory over a fixed time horizon, such that the planned lane-change maneuver is collision-free. To achieve this, we leverage CasADi, an open-source tool for nonlinear optimization and algorithm differentiation, for implementing problem (4.8). We employ a discretization step of $dt = 0.1\ s$ and a planning horizon $t_f = 10\ s$.

Second, we incorporate observation vectors and train an upper-level policy capable of adaptively generating a lane change maneuver when feasible. This scenario mimics real-world driving situations where vehicles must assess the traffic conditions and make dynamic decisions for safe lane changes. We use a simulation time of $t_s = 20\ s$ and apply the same planning settings as in the first setup.

The constraint parameters of problem (4.8) we consider are based on

[18] and practical driving experience, see Table 4.1.

Parameter	**Value**	**Units**
Maximum velocity (v_{max})	13.9	m/s
Minimum velocity (v_{min})	0	m/s
Minimum acceleration (a_{min})	-2.0	m/s^2
Maximum acceleration (a_{max})	1.5	m/s^2
Maximum curvature (κ_{max})	0.02	m^{-1}
Minimum lateral offset (w_{min})	-3.75	m
Maximum lateral offset (w_{max})	1.25	m
Obstacle longitudinal safe distance ($\tilde{s}$)	10	m
Obstacle lateral safe distance ($\tilde{w}$)	0.5	m

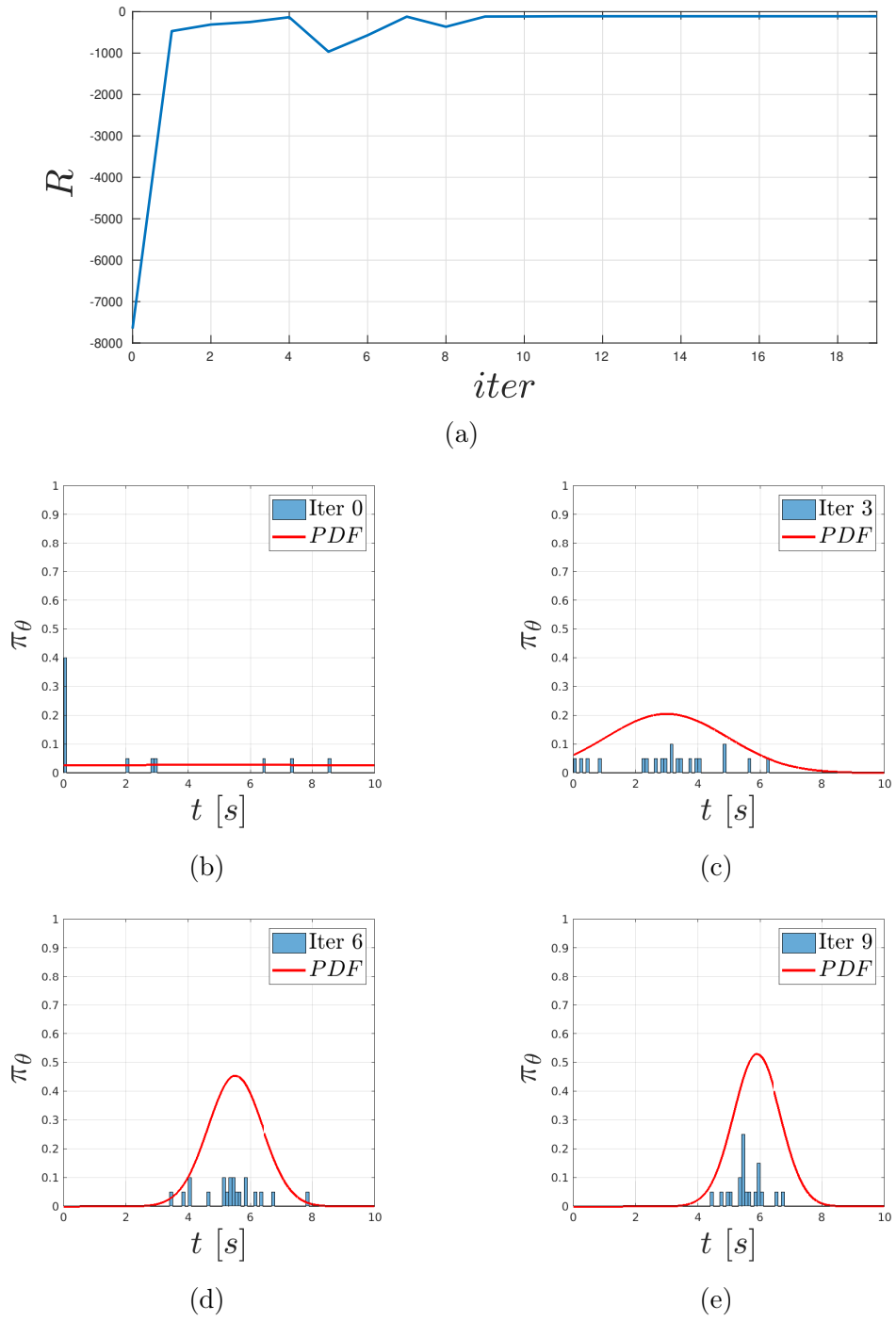
Table 4.1: Set of constraints parameters for the nonlinear MPC to address the lane change problem. The maximum velocity (v_{max}) sets the upper limit for translational motion, while the minimum velocity (v_{min}) establishes a baseline. Acceleration parameters, encompassing minimum acceleration (a_{min}) and maximum acceleration (a_{max}), dictate the system's capability for deceleration or acceleration. Trajectory curvature is bounded by the parameter κ_{max}, guiding the feasibility of navigating through curved paths. Lateral offsets, denoted as w_{min} and w_{max}, represent the minimum and maximum lateral distances from a reference point. Safety considerations are incorporated through obstacle distances. The obstacle longitudinal safe distance ($\tilde{s}$) delineates the required distance in the longitudinal direction from obstacles, while the obstacle lateral safe distance ($\tilde{w}$) establishes the minimum lateral distance for safety.

4.4.1 Upper-level policy for Lane Change Trajectory Generation

The ego-vehicle's initial position is set at $(x_0, y_0) = (80, 0)$, with heading angle $\psi_0 = 0$, and a velocity of $v_0 = 35\ km/h$ (i.e., almost 9.7 m/s). The FV obstacle starts at the position $(x^{FV}(0), y^{FV}(0)) = (130, 0)$ and moves along the ego lane at a constant velocity of $v^{FV} = 11\ km/h$ (3 m/s), while the LV obstacle starts at position $(x^{LV}(0), y^{LV}(0)) = (37, 2.5)$ and travels along the target lane with at a constant velocity of $v^{LV} = 30\ km/h$ (8.3 m/s).

In order to provide an example of finding the parameter θ using the MC-EM algorithm, let us consider the learning progress of the upper-level policy $\pi_{\boldsymbol{\omega}}$ using a fixed value of $\beta = 3$, see Figure 4.3. As discussed before, the algorithm starts by randomly generating a list of I samples of θ_i. In the example, I is set to 20. Each θ_i is drawn from the upper-level policy $\pi_{\boldsymbol{\omega}}(\theta)$, which is modeled as Gaussian distribution with parameters $\boldsymbol{\omega} = (\mu, \sigma^2)$. In the first iteration, see Figure 4.3b, the variance σ^2 is set to a large value. This high variance encourages a wide exploration of the θ domain. Then, a set of predicted trajectories $(\boldsymbol{x}, \boldsymbol{u})$ are obtained by solving I optimization problems (4.8). These trajectories represent different potential lane change maneuvers based on the sampled values of θ_i. The quality of these sampled trajectories is evaluated using the return (4.14). Next, the parameters $\boldsymbol{\omega}$ (mean and variance) are updated solving (4.15), see Fig 4.3c and 4.3d. This update process is repeated until the return no longer converges, see Figure 4.3a.Once convergence is achieved, the policy can be represented by the bell-shaped distribution as the one shown in Figure 4.3e (in this case, convergence is achieved after 9 iterations).

The optimal trajectory is depicted in Figure 4.4. Next, we highlight



(a)

(b)

(c)

(d)

(e)

Figure 4.3: Learning progress of the upper-level policy. The top sub-figure depicts the return curve with a temperature parameter $\beta = 3.0$, while the bottom sub-figures illustrate the policy distribution at various iteration stages $(0, 3, 6, \text{and } 9)$.

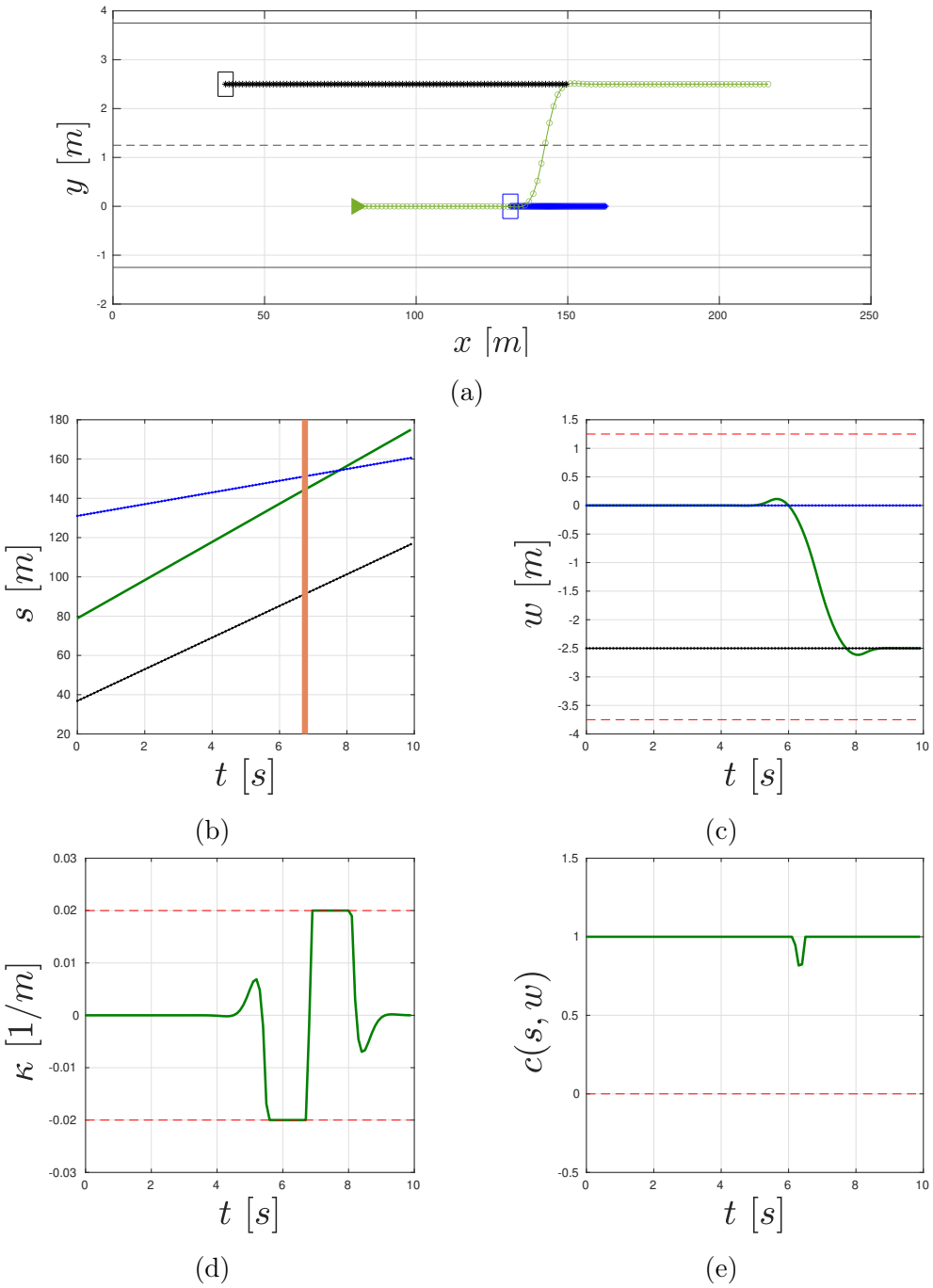


Figure 4.4: Lane change maneuver. The ego-vehicle's optimal maneuver (solid green line) is shown, while the maneuvers of the FV and the LV are depicted in dash-dotted blue and black lines, respectively. The optimal $\theta^* = 6.7\ s$ is highlighted by the vertical orange line.

some interesting features of the generated trajectory. At first glance, we can identify the planning of a lane change maneuver. The ego-vehicle trajectory transitions from its current lane to the target lane, in order to avoid the FV obstacle, which is moving at a slower velocity. At the same time, it avoids a potential collision with the LV, which is travelling on the target lane (it is worth noting that the collision avoidance constraint is always satisfied, see Figure 4.4e). Next, we highlight three distinct phases. First, at the beginning, the ego-vehicle is positioned a considerable distance away from the FV. During this phase, the primary objective is to minimize the cost function J_{el}, which is associated with keeping the ego-vehicle aligned with the center-line of its current lane. The lateral displacement from the center-line is almost zero, see Figure 4.4c. Second, as depicted by the vertical line in Figure 4.4b, at around $t = 6.7\ s$, the upper-level policy triggers a lane-change, thus minimizing the cost function J_{tl}. During this phase, the ego-vehicle steers by applying a negative curvature, see Figure 4.4d, and moves towards the center-line of the target lane, see Figure 4.4a. Finally, the ego-vehicle approaches the center-line of the target lane and proceeds along it, thus confirming the successful execution of the lane change.

4.4.2 Deep Upper-level Policy for Online Scenarios

Next, we want to find an upper-level policy that enables to select online the parameter for executing a lane change based on the environment observations. To do this, we leverage a combination of MC-EM algorithm and self-supervised learning technique as follows.

In order to construct $\boldsymbol{D}$, we collect 100.000 samples of $(\boldsymbol{o}_t, \theta_t^*)$ pairs. Then, we use $\boldsymbol{D}$ to train a MultiLayer Perceptron (MLP) model. We employ TensorFlow [65], a versatile machine learning framework, to im-

plement the MLP. We chose an architecture that consists of 32 hidden layers, each comprising 32 units. Rectified Linear Unit ($ReLU$) nonlinearities are applied to these layers to enhance the model's capacity to learn and generalize from the data. Finally, ADAM optimization is then used to update the weights during the backpropagation phase.

We encourage the reader to refer to the video attachment[1] related to the execution of 20 random maneuvers. Next, we highlight some interesting features related to the first two maneuvers of the attached video.

In the first example, depicted in Figure 4.5a, we can identify two consecutive lane change maneuvers. The scenario begins with both the ego-vehicle and the FV positioned on the same lane, as illustrated in Figure 4.5b. However, there is a significant difference in their velocities: the ego-vehicle and the FV are traveling at 9.7 m/s and at 3.8 m/s, respectively, see Figure 4.5c. As a result, a decision is made by the deep upper-level policy at time $t = 0$ s, leading to a lane change maneuver, as depicted in Figure 4.5a. After such a lane change, the ego-vehicle is traveling along the target lane. However, the LV is also traveling on the same lane (i.e., the target lane) with a velocity of 6.5 m/s. In this situation, the deep upper-level policy once again intervenes by selecting an appropriate θ parameter for the MPC. This decision allows the ego-vehicle to perform another lane change maneuver at about $t = 10$ s, effectively avoiding a collision with the LV. It is important to highlight that during the execution of this maneuver, the avoidance constraint is always satisfied, see Figure 4.5e. In the second example, illustrated in Figure 4.6, we can observe a different behavior compared to the previous case.

[1] https://www.youtube.com/watch?v=oJjOMCAav7I&ab_channel=FrancescoLaneve

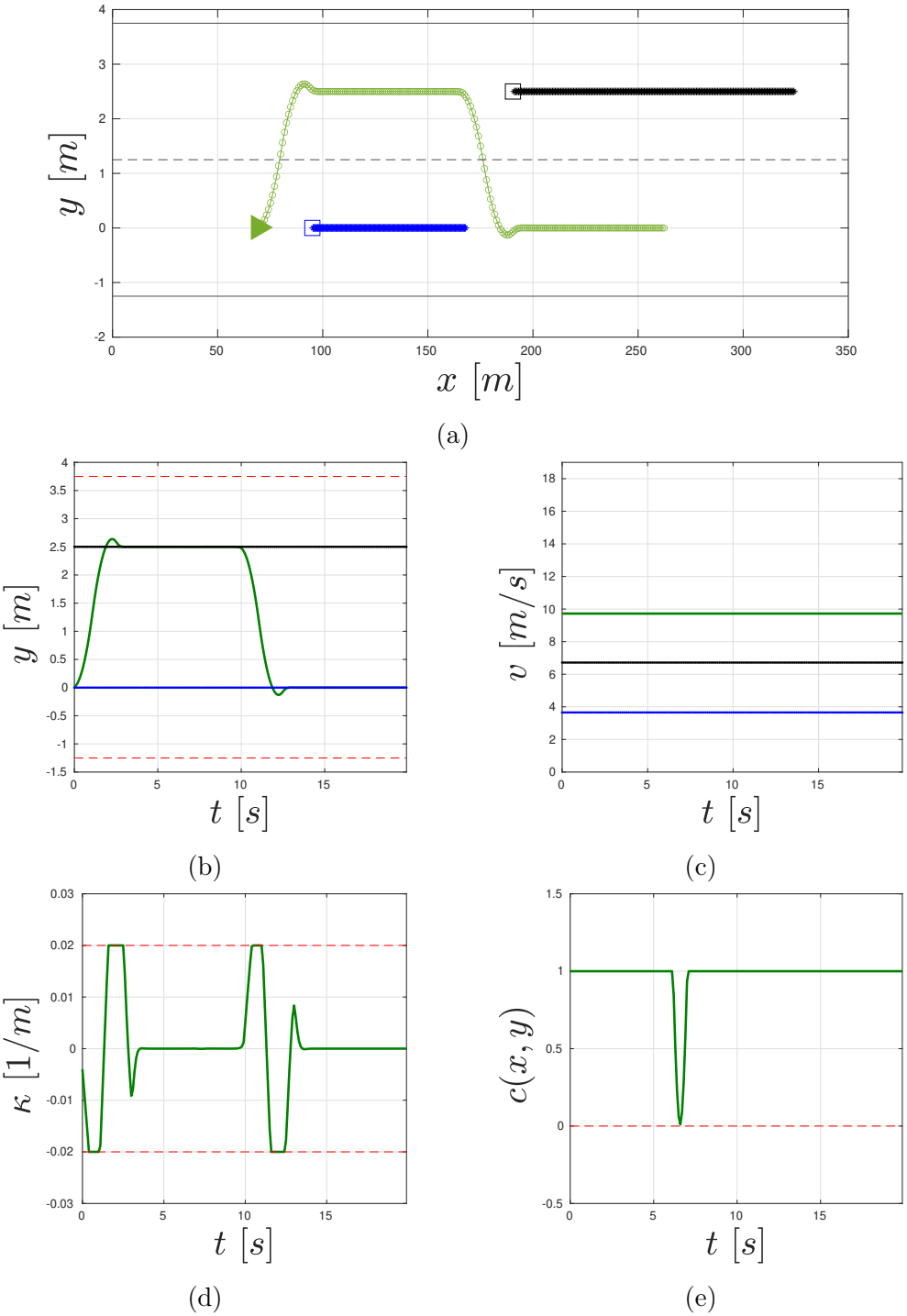


Figure 4.5: Double lane change maneuver. The ego-vehicle's optimal maneuver is shown in solid green line. The maneuvers of the FV and the LV are in dash-dotted blue and black lines, respectively.

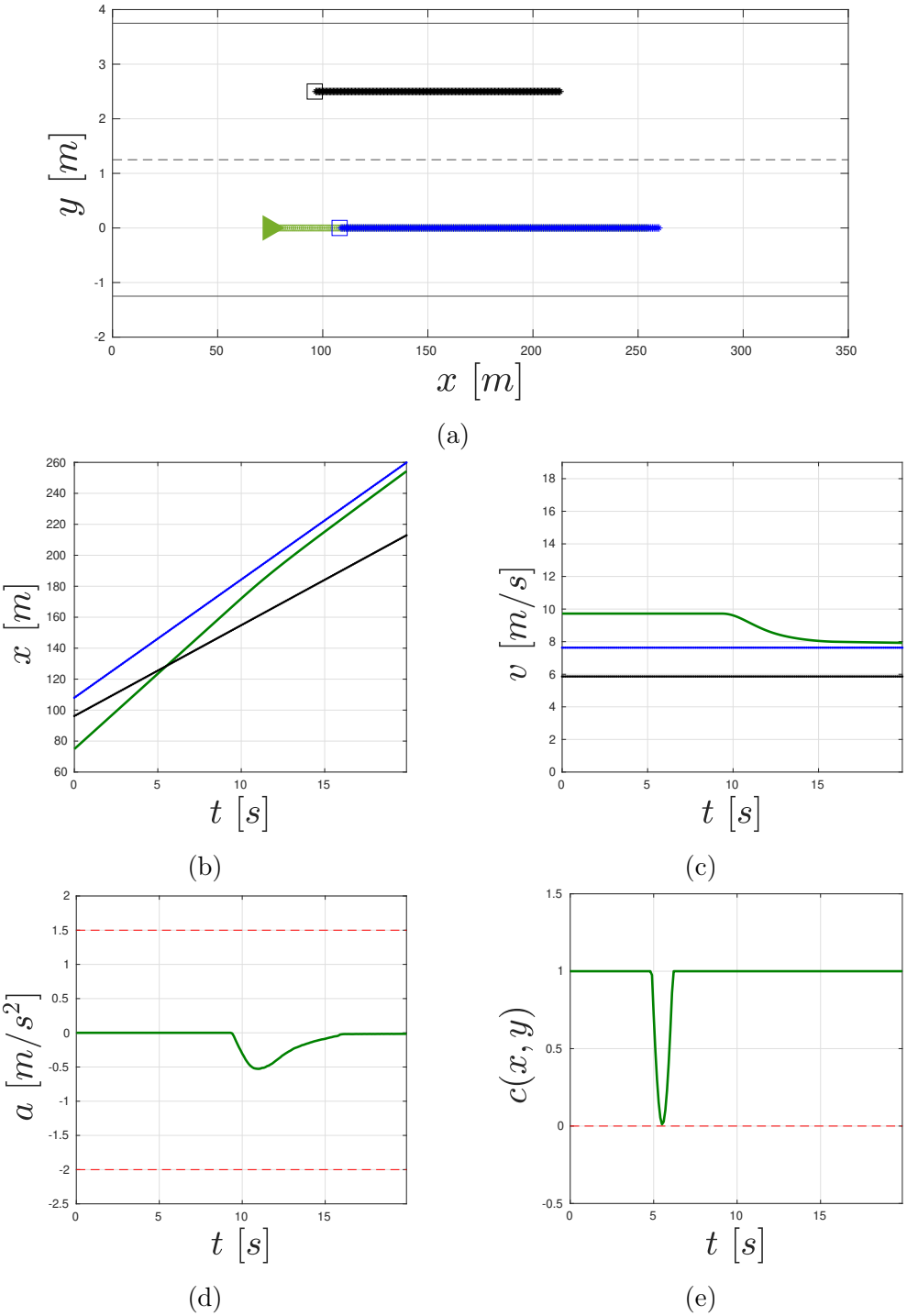


Figure 4.6: Longitudinal avoidance maneuver. The ego-vehicle's optimal maneuver is shown in solid green line. The maneuvers of the FV and the LV are in dash-dotted blue and black lines, respectively.

In this case, the initial positions (see Figure 4.6b) and velocities (see Figure 4.6c) of the ego-vehicle and the two obstacles do not allow the generation of a collision-free lane change trajectory. Indeed, there is no sufficient time-space gap available in order to perform a safe lane change maneuver. In such a case, in order to keep a safety distance from the FV, the ego-vehicle decreases its velocity by applying a negative acceleration, see Figure 4.6d. As expected, also in this case, the safety distance imposed by the avoidance constraint is always satisfied, see Figure 4.6e.

This last scenario highlights the essential requirement to include hard collision avoidance constraints in our proposed formulation. This constraint becomes particularly crucial in situations where the upper-level policy cannot feasibly determine an appropriate θ to ensure collision-free maneuver.

Conclusions

In this dissertation we developed novel methods based on nonlinear optimal control techniques for trajectory generation of autonomous vehicles. These methods have been developed to enhance the capabilities of autonomous vehicles in navigating dynamic environments, with a primary focus on improving safety and passenger comfort.

In the first part of the dissertation, we introduced a family of reduced-order car models that were tailored to facilitate trajectory planning strategies and evaluate these strategies in simulation. We derived the equations of motion for both kinematic and dynamic bicycle models, where the latter included tire modeling for a more realistic representation of vehicle behavior. Furthermore, we redefined the kinematic model in terms of longitudinal and lateral coordinates, simplifying the equations of motion and aligning them with human perception and control of vehicle motion.

In the second part we proposed a strategy for the trajectory generation in a merging scenario with non-cooperative obstacles. The proposed formulation takes advantage of the use of transverse coordinates and the virtual target vehicle approach to capture interesting dynamics features. In particular, an optimization strategy based on optimal control problem has been developed to generate a feasible merging trajectory, and at the same time, guarantees collision avoidance in the presence of obstacles. We evaluated the proposed strategy in a simulated scenario and showed

that the proposed approach can successfully generate a merging trajectory, allowing the autonomous vehicle to perform a safe merging even in presence of multiple obstacles.

As the main contribution of this dissertation, we proposed a real-time strategy to address the problem of generating feasible trajectories for autonomous vehicles in presence of obstacles. In particular, we i) re-write the vehicle dynamics with respect to the transverse coordinates, ii) propose a novel avoidance constraint formulation and, iii) set up a maneuver regulation-based optimal control strategy in the transverse coordinates. We proved the effectiveness of the proposed approach in a simulated scenario and showed that the proposed constraint is able to avoid moving obstacles. Moreover, the optimal maneuvers are used as reference for a low-lever controller of a real vehicle. The integration of the proposed collision-free maneuver generation strategy on a real vehicle shows the feasibility of the computed maneuvers even in presence of unmodeled dynamic effects.

Finally we presented a novel approach to tackle the lane change maneuver challenge by framing it as a parametric model predictive control problem. Unlike conventional methods that decouple decision-making and planning, our approach integrates upper-level policy search with model predictive control-based low-level policy generation to optimize the lane change strategy effectively. We employed a weighted maximum likelihood approach for policy learning and incorporated self-supervised learning techniques to adapt to dynamic, online scenarios, ensuring adaptability to unexpected environmental changes. The numerical results presented demonstrate the efficacy of our approach, underscoring its potential to enhance vehicle maneuvering in dynamic environments.

The results of this dissertation were presented in the following publications [39, 66, 67].

Appendix A

The Projection Operator-based Newton Method

We recall the Projection Operator-based Newton method for trajectory optimization [45] and [68], used to generate optimal maneuvers for the collision-fee algorithm presented in Chapter 3. The problem under consideration is an optimal control problem, which can be formulated as follows:

$$\begin{aligned} \min_{\bar{\boldsymbol{x}},\bar{\boldsymbol{u}}} \quad & \ell(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}}) := \int_0^{s_f} J(\bar{\boldsymbol{x}}(s),\bar{\boldsymbol{u}}(s),s)ds + m(\bar{\boldsymbol{x}}(s_f)) \\ \text{s.t} \quad & \bar{\boldsymbol{x}}' = f(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},s) \quad \bar{\boldsymbol{x}}(0) = x_0\,. \end{aligned} \tag{A.1}$$

The optimal control problem (A.1) can be reformulated as an unconstrained optimization problem by introducing a projection operator $\mathcal{P}$:

$$\mathcal{P} : \boldsymbol{\xi} = (\bar{\boldsymbol{\alpha}}(\cdot),\bar{\boldsymbol{\mu}}(\cdot)) \mapsto \boldsymbol{\tau} = (\bar{\boldsymbol{x}}(\cdot),\bar{\boldsymbol{u}}(\cdot))\,.$$

This operator maps curves $\boldsymbol{\xi} = (\bar{\boldsymbol{\alpha}}(\cdot),\bar{\boldsymbol{\mu}}(\cdot))$, that may not satisfy the vehicle dynamic, to bounded trajectories $\boldsymbol{\tau} = (\bar{\boldsymbol{x}}(\cdot),\bar{\boldsymbol{u}}(\cdot))$ of a given manifold $\mathcal{T}$. This operator allows the original constrained optimization problem

(A.1) to be transformed into an equivalent unconstrained optimization problem, expressed as follows:

$$\min_{\boldsymbol{\xi}\in\mathcal{T}} \ell(\boldsymbol{\xi}) = \min_{\boldsymbol{\xi}} \ell(\mathcal{P}(\boldsymbol{\xi}))\,. \tag{A.2}$$

The projection operator is defined as follows:

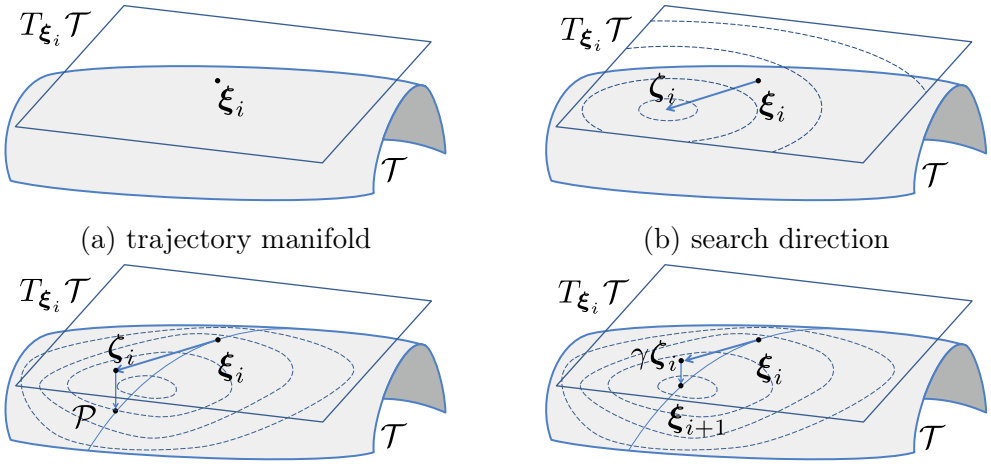
$$\begin{aligned} \bar{\mathbf{x}}(s) &= f(\bar{\mathbf{x}}, \bar{\mathbf{u}}, s), \quad \bar{\mathbf{x}}(0) = \mathbf{x}_0 \\ \bar{\mathbf{u}}(s) &= \bar{\boldsymbol{\mu}}(s) + \boldsymbol{K}(s)(\bar{\boldsymbol{\alpha}}(s) - \bar{\mathbf{x}}(s))\,. \end{aligned} \tag{A.3}$$

The constrained and unconstrained optimization problems are equivalent in the sense that if a trajectory $\boldsymbol{\xi}$ represents an unconstrained local minimum of (A.1), then $\boldsymbol{\xi}^* = P(\boldsymbol{\xi})$ is a constrained local minimum of the modified problem (A.2), see [45]. When applying standard numerical optimization techniques, such as the Newton method or quasi-Newton methods, within finite-dimensional spaces, a descent method for trajectory optimization is employed to minimize the cost functional. In this context, the cost functional $\tilde{\ell}(\boldsymbol{\xi})$ is defined as follows:

$$\tilde{\ell}(\boldsymbol{\xi}) := \ell(P(\boldsymbol{\xi})) \tag{A.4}$$

A geometric representation of the projection operator is shown in Figure A.1. The process of minimizing the trajectory functional involves an iterative approach as presented in Algorithm 3. In this context, $\boldsymbol{\xi}_i$ denotes the current trajectory iterate, $\boldsymbol{\xi}_0$ represents the initial trajectory, and $\boldsymbol{\zeta} \mapsto \nabla\tilde{\ell}(\boldsymbol{\xi}_i)\cdot\boldsymbol{\zeta}$ and $\boldsymbol{\zeta} \mapsto \nabla^2\tilde{\ell}(\boldsymbol{\xi}_i)\cdot(\boldsymbol{\zeta},\boldsymbol{\zeta})$ are respectively the first and second Fréchet differentials of the functional $\tilde{\ell}(\boldsymbol{\xi}) = \ell(P(\boldsymbol{\xi}))$ evaluated at the trajectory $\boldsymbol{\xi}_i$.

The algorithm has the structure of a standard Newton method used for minimizing unconstrained functions. Its core components include the formulation of the projection operator through the design of the matrix



(a) trajectory manifold (b) search direction

(c) line search (d) update

Figure A.1: Geometric representation of the projection operator [1].

$\boldsymbol{K}$ and the computation of derivatives for $\tilde{\ell}(\cdot)$ in order to determine a descent direction. Notably, these crucial steps entail solving linear quadratic optimal control problems, as extensively discussed in [45].

Algorithm 3 Projection Operator Newton Method

Initialize: Choose initial guess $\boldsymbol{\xi}_0 \in \mathcal{T}$

while not converged **do**

Solve the LQR for $\boldsymbol{K}_i$ defining $\mathcal{P}$ about $\boldsymbol{\xi}_i$

Search for descent direction

$$\boldsymbol{\zeta}_i = \arg\min_{\boldsymbol{\zeta}\in T_{\boldsymbol{\xi}_i}\mathcal{T}} \nabla\tilde{\ell}(\boldsymbol{\xi}_i)\cdot\boldsymbol{\zeta} + \frac{1}{2}\nabla^2\tilde{\ell}(\boldsymbol{\xi}_i)\cdot(\boldsymbol{\zeta},\boldsymbol{\zeta})$$

Compute step size via line search $\gamma_i = \arg\min_{\gamma\in(0,1]} \tilde{\ell}(\boldsymbol{\xi}_i + \gamma\boldsymbol{\zeta}_i)$

Project and update $\boldsymbol{\xi}_{i+1} = \boldsymbol{\xi}_i + \mathcal{P}(\boldsymbol{\xi}_i + \gamma\boldsymbol{\zeta}_i)$

end while

Appendix B

Multiple Shooting Method for Optimal Control

We recall the multiple shooting method [2], a powerful numerical technique, used for solving optimal control problems in Chapter 2 and 4. The key to the multiple shooting method lies in dividing the continuous time interval of the optimal control problem into smaller sub-intervals, effectively transforming the problem into a nonlinear programming problem.

Let us begin with a general form of the optimal control problem:

$$\begin{aligned} &\min_{\mathbf{x}(\cdot)\,\mathbf{u}(\cdot)} \int_0^{t_f} J(\mathbf{x}(\tau), \mathbf{u}(\tau))\, d\tau + m(\mathbf{x}(t_f)) \\ &\text{s.t.} \quad \mathbf{x}'(t) = f(\mathbf{x}(t), \mathbf{u}(t))\,, \, \mathbf{x}(0) = \mathbf{x}_0\,. \end{aligned} \tag{B.1}$$

The Multiple Shooting Method proceeds as follows, see Algorithm 4.

The first step is to divide the continuous-time interval, usually denoted as $[0,\ t_f]$, into a finite number of subintervals, typically denoted as N. This results in a sequence of time points, such as $t_0, t_1, \ldots, t_N$. These time points serve as the key nodes where the trajectory of the system will be approximated.

Next, the original optimal control problem (B.1), is reformulated as an nonlinear programming problem over the sub-intervals. For each sub-interval i (ranging from 0 to $N-1$), a local trajectory $(\boldsymbol{x}_i(\cdot),\ \boldsymbol{u}_i(\cdot))$ is defined. These local trajectory aims to approximate the system's behavior within the corresponding sub-interval.

To ensure the continuity of the trajectory, shooting constraints are introduced. These constraints link the final state of one sub-interval to the initial state of the next sub-interval. Mathematically, for each i from 0 to $N-1$, constraints are imposed as:

$$\boldsymbol{x}_i(t_{i+1}) = \boldsymbol{x}_{i+1}(t_{i+1})\,.$$

An objective function for the nonlinear programming problem is constructed based on the cost function J from the original optimal control problem. This objective function is typically the sum of costs over all sub-intervals, aiming to minimize the overall cost.

In order to initiate the optimization process, an initial guess for the initial trajectory is required. This initial guess is critical, as the optimization process will iteratively refine it to approach the true optimal solution.

Then, the nonlinear programming problem, with shooting constraints and the constructed objective function, is solved for each sub-interval. The goal is to find trajectories $(\boldsymbol{x}_i, \boldsymbol{u}_i)$ that minimize the cost while satisfying the continuity constraints.

After solving the problem for each sub-interval, the obtained solution is used to update the overall trajectory $(\boldsymbol{x}, \boldsymbol{u})$. This iterative update refines the approximation of the optimal trajectory.

Finally, convergence criteria are established to determine when to terminate the iterations. Common criteria include monitoring the change in the objective function or ensuring that the shooting constraints are

satisfied within a predefined tolerance. Once these criteria are met, the optimization process is considered converged, and the obtained solution represents an approximation of the optimal trajectory.

Algorithm 4 Multiple Shooting Method for Optimal Control

Divide the time interval $[0, T]$ into N sub-intervals.
Initialize an approximation for the trajectory: $(\boldsymbol{x}_0, \boldsymbol{u}_0)$.
Formulate the optimal control problem as a nonlinear programming problem over sub-intervals.
for $i = 0$ to $N - 1$ **do**
    Define local state and control trajectories: $(\boldsymbol{x}_i, \boldsymbol{u}_i)$.
    Introduce shooting constraints to ensure continuity:

$$\boldsymbol{x}_i(t_{i+1}) = \boldsymbol{x}_{i+1}(t_{i+1})$$

end for
Construct the objective function based on the cost function.
Solve the nonlinear programming problem for each sub-interval to find $(\boldsymbol{x}_i, \boldsymbol{u}_i)$.
repeat
    Update the trajectory $(\boldsymbol{x}, \boldsymbol{u})$ using the nonlinear programming problem solutions.
    Solve the nonlinear programming problem problem again to refine the trajectories.
until Convergence criteria are met.

Bibliography

[1] A Pedro Aguiar, Florian A Bayer, John Hauser, Andreas J Häusler, Giuseppe Notarstefano, Antonio M Pascoal, Alessandro Rucco, and Alessandro Saccon. Constrained optimal motion planning for autonomous vehicles using PRONTO. In *Sensing and control for autonomous vehicles*, pages 207–226. Springer, 2017.

[2] Moritz Diehl, Hans Georg Bock, Holger Diedam, and P-B Wieber. Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer, 2006.

[3] Alessandro Rucco, A. Pedro Aguiar, and John Hauser. Trajectory optimization for constrained UAVs: A virtual target vehicle approach. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 236–245. IEEE, 2015.

[4] Sara Spedicato and Giuseppe Notarstefano. Minimum-time trajectory generation for quadrotors in constrained environments. *IEEE Transactions on Control Systems Technology*, 26(4):1335–1344, 2017.

[5] Alessandro Rucco, Giuseppe Notarstefano, and John Hauser. An efficient minimum-time trajectory generation strategy for two-track

car vehicles. *IEEE Transactions on Control Systems Technology*, 23(4):1505–1519, 2015.

[6] Ivo Batkovic, Mario Zanon, Mohammad Ali, and Paolo Falcone. Real-time constrained trajectory planning and vehicle control for proactive autonomous driving with road users. In *2019 18th European Control Conference (ECC)*, pages 256–262. IEEE, 2019.

[7] Wenjing Cao, Masakazu Mukai, and Taketoshi Kawabe. Merging trajectory generation method using real-time optimization with enhanced robustness against sensor noise. *Artificial Life and Robotics*, 24(4):527–533, 2019.

[8] Francesco Borrelli, Paolo Falcone, Tamas Keviczky, Jahan Asgari, and Davor Hrovat. Mpc-based approach to active steering for autonomous vehicle systems. *International journal of vehicle autonomous systems*, 3(2-4):265–291, 2005.

[9] Mario Zanon, Janick V Frasch, Milan Vukov, Sebastian Sager, and Moritz Diehl. Model predictive control of autonomous vehicles. In *Optimization and optimal control in automotive systems*, pages 41–57. Springer, 2014.

[10] Valerio Turri, Ashwin Carvalho, Hongtei Eric Tseng, Karl Henrik Johansson, and Francesco Borrelli. Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. In *16th international IEEE conference on intelligent transportation systems*, pages 378–383. IEEE, 2013.

[11] Alessandro Beghi, Mattia Bruschetta, and Fabio Maran. A real time implementation of mpc based motion cueing strategy for driving

simulators. In *2012 IEEE 51st IEEE conference on decision and control (CDC)*, pages 6340–6345. IEEE, 2012.

[12] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[13] Thomas Gillespie. *Fundamentals of vehicle dynamics*. SAE international, 2021.

[14] William F Milliken, Douglas L Milliken, and L Daniel Metz. *Race car vehicle dynamics*, volume 400. SAE international Warrendale, 1995.

[15] Florian Bayer and John Hauser. Trajectory optimization for vehicles in a constrained environment. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 5625–5630. IEEE, 2012.

[16] J-A Serret. Sur quelques formules relatives à la théorie des courbes à double courbure. *Journal de mathématiques pures et appliquées*, 16:193–207, 1851.

[17] F Frenet. Sur les courbes à double courbure. *Journal de mathématiques pures et appliquées*, 17:437–447, 1852.

[18] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 1094–1099. IEEE, 2015.

[19] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE intelligent vehicles symposium (IV)*, pages 812–818. IEEE, 2017.

[20] Egbert Bakker, Lars Nyborg, and Hans B Pacejka. Tyre modelling for use in vehicle dynamics studies. *SAE Transactions*, pages 190–204, 1987.

[21] Hans Pacejka. *Tire and vehicle dynamics*. Elsevier, 2005.

[22] Carlos Hidalgo, Ray Lattarulo, Carlos Flores, and Joshué Pérez Rastelli. Platoon merging approach based on hybrid trajectory planning and cacc strategies. *Sensors*, 21(8):2626, 2021.

[23] Robert Hult, Mario Zanon, Sébastien Gros, and Paolo Falcone. Optimal coordination of automated vehicles at intersections with turns. In *2019 18th European Control Conference (ECC)*, pages 225–230. IEEE, 2019.

[24] Ezequiel Gonzale Debada and Denis Gillet. Virtual vehicle-based cooperative maneuver planning for connected automated vehicles at single-lane roundabouts. *IEEE Intelligent Transportation Systems Magazine*, 10(4):35–46, 2018.

[25] Robert Hult, Mario Zanon, Sébastien Gros, and Paolo Falcone. A semidistributed interior point algorithm for optimal coordination of automated vehicles at intersections. *IEEE Transactions on Control Systems Technology*, 2021.

[26] Ivo Batkovic, Mohammad Ali, Paolo Falcone, and Mario Zanon. Safe trajectory tracking in uncertain environments. *arXiv preprint arXiv:2001.11602*, 2020.

[27] A. Pedro Aguiar and Joao P Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE transactions on automatic control*, 52(8):1362–1379, 2007.

[28] A. Pedro Aguiar, Joao P Hespanha, and Petar V Kokotović. Performance limitations in reference tracking and path following for nonlinear systems. *Automatica*, 44(3):598–610, 2008.

[29] Alessandro Saccon, John Hauser, and Alessandro Beghi. A virtual rider for motorcycles: Maneuver regulation of a multi-body vehicle model. *IEEE Transactions on Control Systems Technology*, 21(2):332–346, 2012.

[30] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[31] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[32] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on intelligent transportation systems*, 17(4):1135–1145, 2015.

[33] Steven Diamond and Stephen Boyd. CVXPY: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

[34] Andrea Zanelli, Alexander Domahidi, Juan Jerez, and Manfred Morari. FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1):13–29, 2020.

[35] Michael A Patterson and Anil V Rao. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software*, 41(1):1–37, 2014.

[36] John Canny. *The complexity of robot motion planning*. MIT press, 1988.

[37] Ugo Rosolia, Stijn De Bruyne, and Andrew G Alleyne. Autonomous vehicle control: A nonconvex approach for obstacle avoidance. *IEEE Transactions on Control Systems Technology*, 25(2):469–484, 2016.

[38] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 29(3):972–983, 2020.

[39] Francesco Laneve, Alessandro Rucco, and Massimo Bertozzi. A trajectory optimization strategy for merging maneuvers of autonomous vehicles. In *APCA International Conference on Automatic Control and Soft Computing*, pages 3–14. Springer, 2022.

[40] Michael T Wolf and Joel W Burdick. Artificial potential functions for highway driving with collision avoidance. In *IEEE International Conference on Robotics and Automation*, pages 3731–3736, 2008.

[41] Yong Koo Hwang, Narendra Ahuja, et al. A potential field approach to path planning. *IEEE transactions on robotics and automation*, 8(1):23–32, 1992.

[42] Sara Spedicato, Antonio Franchi, and Giuseppe Notarstefano. From tracking to robust maneuver regulation: An easy-to-design approach

for VTOL aerial robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2965–2970. IEEE, 2016.

[43] John Hauser and Rick Hindman. Maneuver regulation from trajectory tracking: Feedback linearizable systems. *IFAC Proceedings Volumes*, 28(14):595–600, 1995.

[44] Andreas J Häusler, Alessandro Saccon, António Pedro Aguiar, John Hauser, and António M Pascoal. Energy-optimal motion planning for multiple robotic vehicles with collision avoidance. *IEEE Transactions on Control Systems Technology*, 24(3):867–883, 2015.

[45] John Hauser. A projection operator approach to the optimization of trajectory functionals. *IFAC Proceedings Volumes*, 35(1):377–382, 2002.

[46] Alberto Broggi, Stefano Debattisti, Paolo Grisleri, and Matteo Panciroli. The deeva autonomous vehicle platform. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 692–699. IEEE, 2015.

[47] Alberto Broggi, Paolo Medici, Paolo Zani, Alessandro Coati, and Matteo Panciroli. Autonomous vehicles control in the vislab intercontinental autonomous challenge. *Annual Reviews in Control*, 36(1):161–171, 2012.

[48] Karsten Ahnert and Mario Mulansky. Odeint–solving ordinary differential equations in c++. In *AIP Conference Proceedings*, volume 1389, pages 1586–1589. American Institute of Physics, 2011.

[49] Wen Hu, Zejian Deng, Dongpu Cao, Bangji Zhang, Amir Khajepour, Lei Zeng, and Yang Wu. Probabilistic lane-change decision-making and planning for autonomous heavy vehicles. *IEEE/CAA Journal of Automatica Sinica*, 9(12):2161–2173, 2022.

[50] Xuting Duan, Chen Sun, Daxin Tian, Jianshan Zhou, and Dongpu Cao. Cooperative lane-change motion planning for connected and automated vehicle platoons in multi-lane scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[51] Juqi Hu, Youmin Zhang, and Subhash Rakheja. Adaptive lane change trajectory planning scheme for autonomous vehicles under various road frictions and vehicle speeds. *IEEE Transactions on Intelligent Vehicles*, 8(2):1252–1265, 2022.

[52] Suiyi He, Jun Zeng, Bike Zhang, and Koushil Sreenath. Rule-based safety-critical control design using control barrier functions with application to autonomous lane change. In *2021 American Control Conference (ACC)*, pages 178–185. IEEE, 2021.

[53] Charlott Vallon, Ziya Ercan, Ashwin Carvalho, and Francesco Borrelli. A machine learning approach for personalized autonomous lane change initiation and control. In *2017 IEEE Intelligent vehicles symposium (IV)*, pages 1590–1595. IEEE, 2017.

[54] Zhaolun Li, Jingjing Jiang, and Wen-Hua Chen. Automatic lane change maneuver in dynamic environment using model predictive control method. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2384–2389. IEEE, 2020.

[55] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[56] Shuojie Mo, Xiaofei Pei, and Chaoxian Wu. Safe reinforcement learning for autonomous vehicle using monte carlo tree search. *IEEE*

Transactions on Intelligent Transportation Systems, 23(7):6766–6773, 2021.

[57] Jun Chen, Xiangyu Meng, and Zhaojian Li. Reinforcement learning-based event-triggered model predictive control for autonomous vehicle path following. In *2022 American Control Conference (ACC)*, pages 3342–3347. IEEE, 2022.

[58] Arun Muraleedharan, Hiroyuki Okuda, and Tatsuya Suzuki. Real-time implementation of randomized model predictive control for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(1):11–20, 2021.

[59] Flavia Sofia Acerbo, Herman Van der Auweraer, and Tong Duy Son. Safe and computational efficient imitation learning for autonomous vehicle driving. In *2020 American Control Conference (ACC)*, pages 647–652. IEEE, 2020.

[60] Yunlong Song and Davide Scaramuzza. Learning high-level policies for model predictive control. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7629–7636. IEEE, 2020.

[61] Yunlong Song and Davide Scaramuzza. Policy search for model predictive control with application to agile drone flight. *IEEE Transactions on Robotics*, 38(4):2114–2130, 2022.

[62] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Advances in neural information processing systems*, 21, 2008.

[63] Jan Peters and Stefan Schaal. Applying the episodic natural actor-critic architecture to motor primitive learning. In *ESANN*, pages 295–300, 2007.

[64] Gerhard Neumann and Jan Peters. Fitted q-iteration by advantage weighted regression. *Advances in neural information processing systems*, 21, 2008.

[65] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[66] Francesco Laneve, Alessandro Rucco, and Massimo Bertozzi. A real-time collision-free maneuver generation algorithm for autonomous driving. *European Journal of Control*, page 100865, 2023.

[67] Francesco Laneve, Alessandro Rucco, and Massimo Bertozzi. A hybrid approach combining upper-level policy search and mpc for lane change of autonomous vehicles. In *2024 European Control Conference (ECC) (under review)*, 2024.

[68] John Hauser and David G Meyer. The trajectory manifold of a nonlinear control system. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*, volume 1, pages 1034–1039. IEEE, 1998.

Acknowledgements

I would like to express my deepest gratitude to who have played instrumental roles in my journey towards completing my Ph.D.

First and foremost, I extend my heartfelt thanks to my Ph.D advisor, Dr. Alessandro Rucco, for his unwavering support, guidance, and mentorship throughout my doctoral research. Your expertise, patience, and dedication have been invaluable, and I am profoundly grateful for the opportunities you provided to me.

I have also to express my gratitude to my tutor, Prof. Massimo Bertozzi, for his support throughout my academic journey.

I would like to acknowledge VisLab, the company where I conducted my research, for their invaluable resources and collaborative environment. The opportunities to work on real-world pioneer research and engage with industry professionals have enriched my academic experience and provided a broader perspective on my research.

To my parents, Giuseppe and Maria Letizia, my siblings, Eleonora, Giovanni, and Federica, and my sister form another mother, Miriana, I extend my sincere appreciation for always believing in me and providing the encouragement and stability I needed. You have been my constant source of love and support throughout this challenging journey.

To Erika, I am deeply grateful for your unwavering love, patience, and understanding. Your emotional support and belief in me have been a

driving force during the most trying moments of my Ph.D. I look forward to our shared future and the adventures we'll embark on together.

Lastly, I would like to thank all the friends and colleagues who have been a part of my academic and personal life, offering support, camaraderie, and shared experiences that have enriched my journey.

This journey would not have been possible without your collective support. I am sincerely grateful for your contributions to my academic and personal growth.