Università degli Studi di Napoli Federico II
Ph.D. Program in
Information Technology and Electrical Engineering
XXXV Cycle

Thesis for the Degree of Doctor of Philosophy

Control Architectures for Advanced Driver Assistance Systems

by
Nicola Albarella

Advisor: Prof. Stefania Santini

Co-advisor: –

Scuola Politecnica e delle Scienze di Base
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

To my family

Control Architectures for Advanced Driver Assistance Systems

Ph.D. Thesis presented
for the fulfillment of the Degree of Doctor of Philosophy
in Information Technology and Electrical Engineering
by

Nicola Albarella

October 2022

Approved as to style and content by

_____

Prof. Stefania Santini, Advisor

_____

Università degli Studi di Napoli Federico II
Ph.D. Program in Information Technology and Electrical Engineering
XXXV cycle - Chairman: Prof. Stefano Russo

http://itee.dieti.unina.it

Candidate's declaration

I hereby declare that this thesis submitted to obtain the academic degree of Philosophiæ Doctor (Ph.D.) in Information Technology and Electrical Engineering is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
Parts of this dissertation have been published in international journals and/or conference articles (see list of the author's publications at the end of the thesis).

Napoli, December 21, 2022

_____

Nicola Albarella

Abstract

The rapid economic growth has led to an increasing number of vehicles on the road, thus increasing the number of road accidents as well. This issue is a serious dilemma, laying economic burdens on governments, as well as, safety problems on people. Advanced Driving Assistance Systems (ADAS) are software modules assisting the driver, as the name suggests, in monitoring the environment and controlling the vehicle itself. These modules have been demonstrated to be effective in reducing the rate of collisions, and are the main focus of this thesis.
Increasing the level of safety, thus bridging the gap between driving assistance and autonomous driving is a challenging task. While in the former, a safety driver is always there, and ready to intervene if needed, in the latter the driver could even not be on board at all. Therefore, the vehicle must be capable of driving itself, in any scenario, despite the adversity of the environment (e.g. road asphalt condition, weather, etc.), the uncertainty in sensor measurements and the complex interactions with other road users. This thesis tackles this problem from multiple points of view. First, it is shown how, by properly design state of the art ADAS, e.g. by endowing these with additional environment information, it is possible to enhance the overall safety. Moreover, a novel motion planning control architecture is presented. By properly combining the latest advancements in Machine Learning and Optimal Control, safe, effective and scalable driving policies can be learned from data. It will be shown how, by making safety formally explicit, constraints can be put on Machine Learning techniques, thus increasing both performances and safety.

Keywords: Autonomous Driving, Advanced Driving Assistance Systems, Planning, Control

Sintesi in lingua italiana

La rapida crescita economica ha portato all'aumento dei veicoli presenti sulle strade, e conseguentemente, all'aumento degli incidenti stradali. I sistemi elettronici di assistenza alla guida sono moduli software che, come il nome suggerisce, assistono il conducente nel monitoraggio dell'ambiente stradale e nel controllo del veicolo stesso. La loro efficacia nel ridurre il tasso di incidenti stradali, o la loro severità, è stata largamente dimostrata, e dunque sono il focus di questa tesi.
Il fine ultimo è l'incremento della sicurezza stradale e l'innalzamento delle performance dei sistemi di guida assistita in prospettiva di guida autonoma.
Per quanto riguarda la guida assistita, nonostante il conducente sia agevolato nella guida, deve essere sempre pronto ad intervenire in caso di necessità. Al contrario, nel caso di guida autonoma, il conducente potrebbe anche non essere a bordo. Questo implica che un veicolo a guida autonoma deve essere in grado di arrivare a destinazione in qualsiasi scenario stradale, nonostante le incertezze di misura e le complesse interazioni con gli altri utenti della strada. In questa tesi è stato affrontato il problema sopra menzionato, da più punti di vista. Prima di tutto, la tesi mostra come, integrando nei sistemi ADAS (Adaptive Cruise Control, Frenata di Emergenza), informazioni sullo stato del manto stradale, è possibile aumentare il livello di sicurezza rispetto a quanto ad oggi fatto allo stato dell'arte. Nella seconda parte, la tesi affronta il tema più complesso della pianificazione del moto per la guida autonoma. In particolare, combinando tecniche di Machine Learning e Controllo Ottimo, è stato progettato un sistema di pianificazione in grado di generare policy di guida sicure e scalabili.

Parole chiave: Guida Autonoma, Guida Assistita, Pianificazione, Controllo

Contents

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Prof. Stefania Santini, for her guidance and support during my PhD journey. The work presented in this thesis has been carried out in Daisy Lab, led by Prof. Stefania Santini.
Moreover, the research introduced in this dissertation has been supported by a PhD scholarship funded by Kineton S.r.l., and for this reason I would like to thank Prof. Giovanni Fiengo and Dr. Manuela Tufo, for giving me the chance of pursuing a PhD in the first place.
Finally, I would like to thank Tambet Matiisen and Prof. Arun Kumar Singh, for giving me the chance to work in their Autonomous Driving Lab, at the University of Tartu, Estonia. It has been a great learning experience, both professionally and personally.

List of Acronyms

The following acronyms are used throughout the thesis.

AV	Autonomous Vehicle
AD	Autonomous Driving
SAE	Society of Automotive Engineers
ADAS	Advanced Driver Assistance System
ECU	Electronic Control Unit
CAN	Control Area Network
FCW	Forward Collision Warning
ACC	Adaptive Cruise Control
AEB	Autonomous Emergency Brake
ABS	Anti-Lock Braking System
TTC	Time To Collision
CoG	Center of Gravity
MIL	Model In the Loop

VIL	Vehicle In the Loop
MPC	Model Predictive Controller
NMPC	Non-linear Model Predictive Controller
ML	Machine Learning
DL	Deep Learning
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
MLP	Multi-Layer Perceptron
DNN	Deep Neural Network
CNN	Convolutional Neural Network
YOLO	You Only Look Once
MTT	Multi Target Tracking
GNN	Global Nearest Neighbour
IOU	Intersection Over Union
KF	Kalman Filter
CCRS	Car to Car Rear Stationary
CCRM	Car to Car Rear Moving
CCRB	Car to Car Rear Braking
MDP	Markov Decision Process
DQN	Deep-Q Learning

PPO Proximal Policy Optimization

RSS Responsibility Sensitive Safety

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Scope

The rapid economic growth has lead to a considerable expansion of the number of circulating vehicles, especially in big cities, therefore leading to increased traffic congestion and risk of accidents [1]. In the effort of making the driving experience safer, the first vehicle assistance systems where developed around the seventies, e.g. the Anti-Lock Braking System and the Traction Control. Despite their effectiveness has been widely acknowledged, the first systems provide no support if the driver is distracted or in altered state. On the other hand, the continuous improvement on embedded hardware platform enabled the development of *active* assistance electronics in commercial vehicles, in order to manage both safety and performances. Advanced Driver Assistance System (ADAS) are considered a valid solution in alleviating transportation problems, effectively supporting the driver by giving warnings or actively taking over in some driving tasks.
Within this technological paradigm, by properly combining or designing such assistance systems, the vehicles are going to become more and more automated, towards the fully Autonomous Vehicle (AV), in which the human driver is taken out of the loop.
However, modern Autonomous Driving (AD) solutions (e.g. Waymo, Cruise) are built on expensive sensor architectures and are typically developed and deployed on geo-fenced pre-mapped areas, such as cities or even single

routes. Instead, ADAS are nowadays installed on many high-end vehicles and, as production prices decrease, they are promised to low-end vehicles as well. These systems are built on simpler and cheaper sensors, thus they are compatible with a wide variety of target cars and are not limited to specific regions.
In order to bridge the gap between assisted and autonomous driving, it is required to come up with safe, effective and scalable software solutions, so that they can be deployed world-wide, in any road scenario and on any vehicle. Current solutions are tailored to specific driving scenarios and situations, moreover, they require a great deal of fine tuning and testing in controlled environments. Nonetheless, by properly designing the control logic it is possible to increase the overall safety level.
In this work the problem of increasing the vehicles autonomy level is addressed. It is shown that by embedding road state information into state of the art longitudinal ADAS, safety can be guaranteed even in adverse driving scenarios. Moreover, it is shown how co-simulation platforms can greatly ease the software development and testing, thus reducing the overall system cost.
Finally, a novel hierarchical planning architecture is introduced which leverages recent advancements in Machine Learning (ML) and classical control theory. The employment of data-based techniques allows for greater scalability and effectiveness with respect to the state of the art, whereas the classical control can be leveraged to ensure safety at all times.

1.2 Contributions

The main contributions of this thesis are:

- a camera-based Forward Collision Warning (FCW) system is proposed along with a purposely designed co-simulation platform for model based testing. By leveraging a camera, the total system cost is reduced (e.g. with respect to RADAR or LIDAR based systems), moreover the system is designed in such a way that no accurate camera calibration is needed,

- a set of road-grip aware longitudinal ADAS modules is proposed along with model-based techniques to estimate the grip online. The

proposed architecture achieves enhanced safety performances with respect to the state of the art in adverse road adhesion scenarios, moreover, by leveraging, model based estimation, no expensive sensor is required,

- a hierarchical planning architecture is introduced. By formally modelling safety, and appropriately designing the driving policy, enhanced safety performances are achieved with respect to the Deep Reinforcement Learning (DRL) state of the art, meanwhile achieving scalability by leveraging data-driven solutions.

1.3 Thesis Outline

The remainder of this thesis is structured as follows:

- Chapter 2 presents the technical background and notions related to the work in this thesis. A brief introduction on the concept of assisted and autonomous driving is given, along with the classification of autonomous vehicles and the description of the classical software stack.
  Basic concepts related to control theory are given. Specifically, three models are described in details, since they are used throughout the rest of thesis. A short introduction on the Model Predictive Controller (MPC) follows the models description and finally basics of the Responsibility Sensitive Safety (RSS) are sketched.
  In the last section of the chapter, basics of Reinforcement Learning (RL) are given, after the description of the Markov Decision Process (MDP) model. The description of Deep-Q Learning (DQN) and Proximal Policy Optimization (PPO) ends the chapter.

- Chapter 3 describes the design and validation of a FCW system for a class L7E electric vehicle. After a brief review of the related work, the architectural design is described by highlighting the design choices that allows for the targeting of low-end vehicles. Finally the system is validated on both simulated and real experiments.

- Chapter 4 describes the design and validation of a set of longitudinal road-grip aware ADAS modules, specifically, Adaptive Cruise Con-

trol (ACC), Autonomous Emergency Brake (AEB) and Anti-Lock Braking System (ABS). After motivating the research topic and reviewing the related work, the road grip estimation technique is introduced, along with the ADAS designs. Finally the overall architecture is extensively validated in simulation.

- Chapter 5 describes the design and validation of a hierarchical highway planner. Similarly to the previous chapter, the motivation and related works are discussed first. The overall design is later introduced and validated on simulation.

- Chapter 6 summarizes the work presented in this thesis, by highlighting the main research issue and how it has been tackled in this work.

Chapter 2

Technical Background

This is an epigraph, cool

Nicola Albarella

2.1 Autonomous Driving

An Autonomous Vehicle (AV) is a one that can drive itself, from a starting point to a destination, without inputs from human drivers. Various level of autonomy have been defined by the SAE J3016 [2], specifically six levels of increasing autonomy. By referring to Fig. 2.1, from the first to third level only partial automation is achieved because the human driver is still in control, even if the features are engaged. For the aforementioned reason, SAE refers to these features as ADAS, i.e. systems which aim is to support the driver in the driving functions, thus increasing safety and the overall driving experience.
At Level 3, the biggest leap can be found, where the driver is not driving when the features are engaged. Nonetheless, if the AV requests it, the human driver must take the control back. This property is fundamental to understand the difference between levels 2 and 3. Specifically, at Level 3, the vehicle must be capable of understanding its own flaws, mistakes and limitations so that it can ask the human driver to intervene.
At the last two levels, Level 4 and Level 5, the human intervention is not needed anymore, meaning that the AV can classify its own errors and fix

Figure 2.1. SAE J3016 levels of driving autonomy

them accordingly.

Nowadays, commercially available vehicles can reach up to Level 2. Despite the fact that controlling the vehicle dynamics is considered a solved problem, the AVs are still not *smart* enough to reliably negotiate the road with other human drivers, especially in cluttered urban environments.

Nonetheless, Level 4 AVs are available for use, but they are limited to drive in geo-fenced areas, such as cities or even single routes, moreover they are built on expensive sensor suites, thus making the employed architecture hard to scale worldwide. The key technologies enabling AD are combinations of sensors, such as cameras, RADARs, LIDARs, and GNSS, in combination to complex software stacks, which usually involve techniques related to artificial intelligence, sensor fusion and control theory.

Historically, the software stack has been split in three components, namely *Sensing*, *Planning* and *Control*, see Fig. 2.2 for reference. The sensing component is responsible for localizing the AV in a reference frame, along with detecting static and dynamic objects in the environment. The planning component is responsible to map the sensing state to a reference trajec-

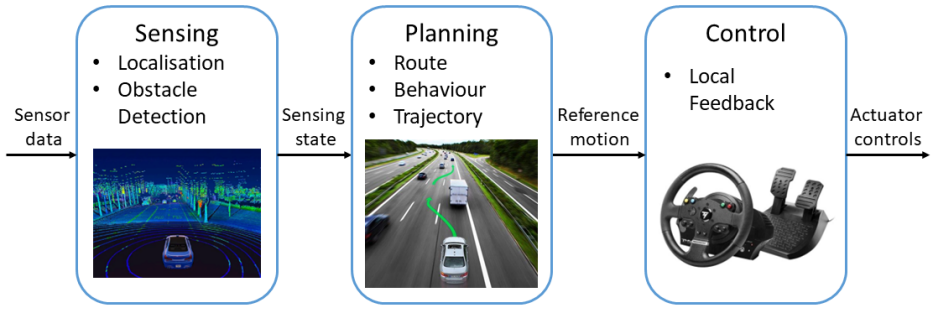


Figure 2.2. Classical architecture of the autonomous driving software stack.

tory in continuous time, whereas the control component is responsible for actuating it on the pedals and steering wheel.
A great deal of effort has been given to the Sensing module, which can give super-human results thanks to recent advancements in Deep Learning (DL). On the other side of the stack, the Control module is typically solved through classical control theory which also gives exceptional results. Despite the fact that, in the recent years many works have been focusing on the Planning module, it is still unclear how one can build a safe, reliable and efficient planners.
It is worth to point out that an alternative to the classical architecture is the end-to-end architecture [3], i.e. a single neural network trained from the raw sensor data to output control commands directly. Nonetheless, such solution is out of the scope of this work.

2.2 Control Theory

In this section a set of useful mathematical models is introduced. Several models have been used in the research literature to describe the state dynamics of a four-wheeled vehicle. Despite having various level of details, they all find use in either planning, control or estimation applications. The choice of a specific model is usually a trade-off between model accuracy and computational efficiency. Additionally, a highly detailed model usually need more parameters, which could be hard to measure or estimate in general.

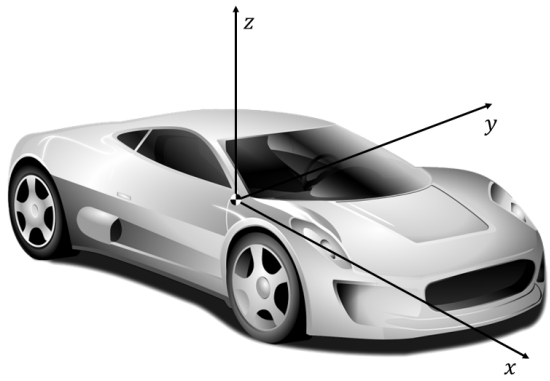


Figure 2.3. Vehicle body frame of reference O_V.

In the following four reference frames will be used, namely the vehicle frame attached to its Center of Gravity (CoG) $O_V = (\vec{x}, \vec{y}, \vec{x})$ (see Fig. 2.3), the four frames $O_W = (\vec{l}, \vec{c}, \vec{n})$ attached to each wheel and the fixed inertial frame. With reference to the inertial frame, multiple choices are available, namely either Cartesian frame $O_C = (\vec{X}, \vec{Y}, \vec{Z})$ and the curvilinear Frenet frame $O_F = (\vec{\sigma}, \vec{e}_y)$ (see Fig. 2.4).
Assuming a two dimensional motion, i.e. ignoring $\vec{Z}$ coordinate,the Cartesian vehicle pose is described by its position (X, Y) and yaw angle ψ, which is defined positive counterclockwise and null when the vehicle is aligned to the X-axis, i.e.:

$$\begin{aligned} \dot{X} &= v_x \cos\psi - v_y \sin\psi, \\ \dot{Y} &= v_x \sin\psi + v_y \cos\psi, \\ \dot{\psi} &= r, \end{aligned} \tag{2.1}$$

where v_x and v_y are the vehicle velocity components in the body reference frame O_V and r is the yaw rate. On the other hand the Frenet frame uses a curvilinear coordinate system to describe the vehicle pose with respect

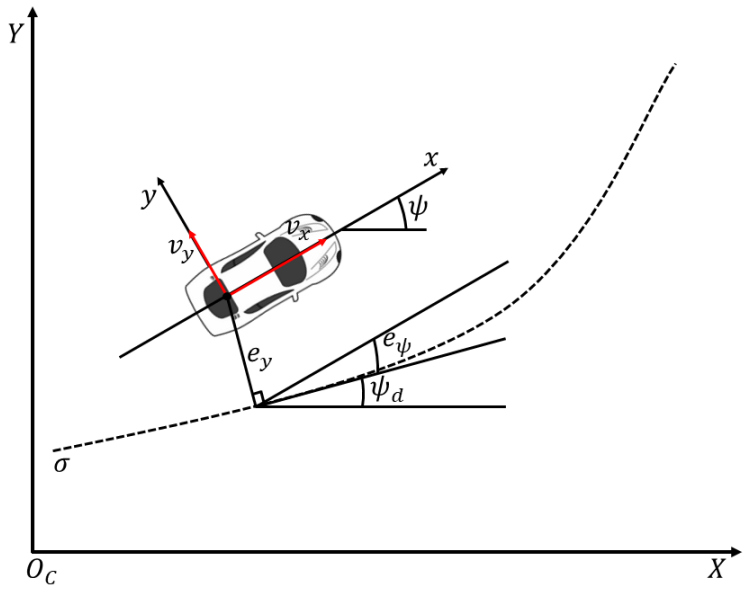


Figure 2.4. Inertial reference systems with Cartesian O_C and Frenet O_F coordinates.

to a known curve. Thus the vehicle pose is given by the couple (σ, e_y) and the angle e_ψ, where σ is the longitudinal position along the curve, or arc length, and e_y is the lateral displacement to the curve and e_ψ is the angle between the vehicle heading and the tangent to the curve. In this reference the 2-D motion is described as:

$$\begin{aligned} \dot{\sigma} &= \frac{1}{1-\rho e_y}\big(v_x \cos e_\psi - v_y \sin e_\psi\big), \\ \dot{e}_y &= v_x \sin e_\psi + v_y \cos e_\psi, \\ \dot{e}_\psi &= r - \dot{\psi}_d, \end{aligned} \tag{2.2}$$

where ρ and ψ_d are the reference line curvature and heading, respectively. The model in Eq. (2.2) is particularly convenient when a local reference line is known and parameterized in terms of its curvature and heading. This is usually true in highly structured environments, such as highway or urban scenarios, where the lane center-lines are taken as references.

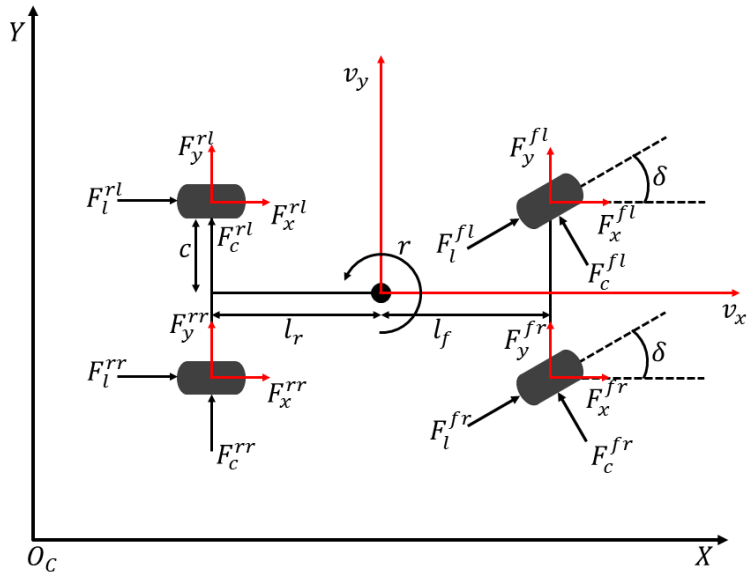


Figure 2.5. Nonlinear four wheel dynamical model showing forces in both reference frames O_V and O_W.

2.2.1 Four Wheels Dynamical Model

The four-wheeled dynamical model is a highly detailed ten dimensional model in which both chassis and tire dynamics are described. It is computationally expensive to be used in a control application, still it is suitable for simulation, thus it allows to build realistic simulation platforms to be leveraged for software testing.
The nonlinear model, depicted in Fig. 2.5, can be summarized as follow:

$$\dot{\xi} = f^{4w}(\xi, u), \tag{2.3}$$

where $\xi \in \mathbb{R}^N$ is the state with $N = 10$, whereas $u \in \mathbb{R}^m$ with $m = 5$ is the control action. Specifically, the state is the combination of longitudinal and lateral velocity in the body frame, v_x and v_y, the yaw ψ and the yaw rate r, the longitudinal and lateral coordinates in the inertial frame, X and Y, along with the speed of the four wheels ω^{fl}, ω^{fr}, ω^{rl}, ω^{rr}, where the first superscript differentiate between front and rear wheels,

whereas the second differentiate between left and right wheel. Moreover, the input is the combination of the steering angle δ and the four wheel torques T^{fl}, T^{fr}, T^{rl}, T^{rr}. A positive torque denotes acceleration and might be available only for driving wheels, whereas negative torque, i.e. braking torque, is available for all the wheels, in general. In the following the general superscript $\{i, j\}$, with $i \in \{f, r\}$ and with $j \in \{l, r\}$, can be used to generalize wheel's notation.
The chassis dynamics in Eq. (2.3) can be obtained by considering the forces about the vehicle CoG, thus obtaining:

$$\begin{aligned} m\dot{v}_x &= mv_y r + F_x^{fr} + F_x^{fl} + F_x^{rr} + F_x^{rl}, \\ m\dot{v}_y &= -mv_x r + F_y^{fr} + F_y^{fl} + F_y^{rr} + F_y^{rl}, \\ I_z\dot{r} &= l_f F_y^{fl} - l_r(F_y^{rl} + F_y^{rr}) + c(F_x^{fr} - F_x^{fl} + F_x^{rr} - F_x^{rl}), \end{aligned} \tag{2.4}$$

where m is the vehicle mass, I_z is the vehicle inertia about the z-axis, l_f and l_r are the distances from the CoG to the front/rear wheel, whereas c is the distance from the CoG to the left/right wheels, and finally (F_x^{ij}, F_y^{ij}) are the tire forces components, in the body frame. Tire forces can be written as:

$$\begin{aligned} F_x^{ij} &= F_l^{ij}\cos\delta_{ij} - F_c^{ij}\sin\delta_{ij}, \\ F_y^{ij} &= F_l^{ij}\sin\delta_{ij} + F_c^{ij}\cos\delta_{ij}, \end{aligned} \tag{2.5}$$

where F_l^{ij} and F_c^{ij} are the wheel forces components in the $\{ij\}$ tire frame. The wheel dynamics are:

$$\begin{aligned} I_w\dot{\omega}^{fl} &= -F_l^{fl}R_w + T^{fl}, \\ I_w\dot{\omega}^{fr} &= -F_l^{fr}R_w + T^{fr}, \\ I_w\dot{\omega}^{rl} &= -F_l^{rl}R_w + T^{rl}, \\ I_w\dot{\omega}^{rr} &= -F_l^{rr}R_w + T^{rr}, \end{aligned} \tag{2.6}$$

where I_w is the wheel inertia and R_w is the wheel radius. The tire forces F_l^{ij} and F_c^{ij} in Eq. (2.5) are given by the non-linear function:

$$\begin{aligned} F_l^{ij} &= f_l(\lambda^{ij}, \mu, F_n^{ij}), \\ F_c^{ij} &= f_c(\alpha^{ij}, \mu, F_n^{ij}), \end{aligned} \tag{2.7}$$

where λ^{ij} and α^{ij} are the longitudinal and lateral slip respectively, μ is the road grip coefficient and F_n^{ij} is the normal force acting on the tire.
The nonlinear function in Eq. (2.7) are modeled by using the well-known Pacejka magic formula [4]. It is a complicated semi-empirical model able to describe tire behaviours over its complete state space, both in the linear and non-linear region. First define the tires' slips as:

$$\begin{aligned} \lambda^{ij} &= \frac{\omega^{ij} R_w - v_l^{ij}}{\max(\omega^{ij} R_w, v_l^{ij})}, \\ \alpha^{ij} &= \arctan\left(\frac{v_c^{ij}}{v_l^{ij}}\right), \end{aligned} \tag{2.8}$$

where v_l^{ij} and v_c^{ij} are the tire speeds components in the $\{ij\}$ tire frame, as:

$$\begin{aligned} v_l^{ij} &= v_y^{ij} \sin\delta_{ij} + v_x^{ij} \cos\delta_{ij}, \\ v_c^{ij} &= v_y^{ij} \cos\delta_{ij} - v_x^{ij} \sin\delta_{ij}, \end{aligned} \tag{2.9}$$

where v_x^{ij} and v_y^{ij} are the tire speeds coordinates in the body frame, evaluated like in the following:

$$\begin{aligned} v_y^{fj} &= v_y + l_f r, \\ v_y^{rj} &= v_y - l_r r, \\ v_x^{il} &= v_x - cr, \\ v_x^{ir} &= v_x + cr. \end{aligned} \tag{2.10}$$

Finally the normal forces acting on the wheels are:

$$\begin{aligned} F_n^{fj} &= \frac{l_r mg}{2(l_f + l_r)}, \\ F_n^{rj} &= \frac{l_f mg}{2(l_f + l_r)}, \end{aligned} \tag{2.11}$$

where $g = 9.81 m/s^2$ is the acceleration of gravity.

Remark 1 *The tire dynamics in Eq. (2.6) are of order of magnitudes*

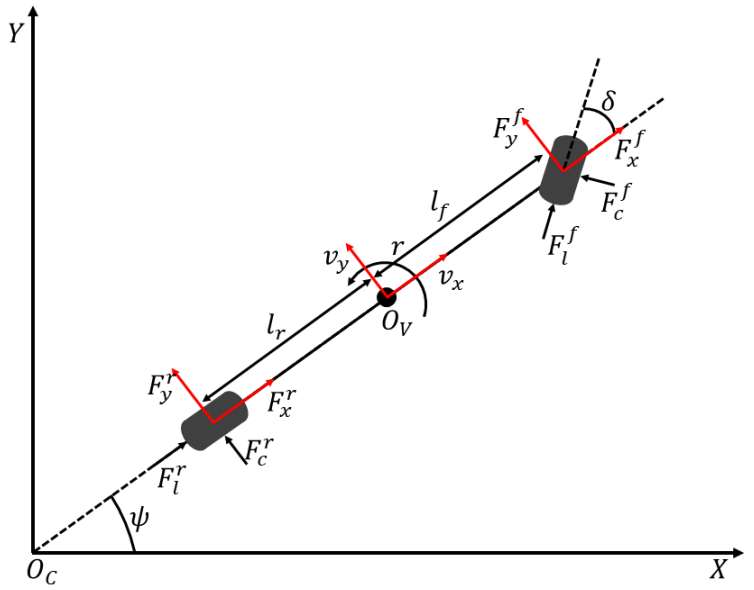


Figure 2.6. Nonlinear dynamical bicycle model showing forces in both reference frames O_V and O_W.

faster than the chassis dynamics in Eq. (2.4), due to the scales of mass and inertia involved in the models. For the aforementioned reason, the tire dynamics can be omitted for chassis control purposes, thus obtaining a simpler model.

In order to complete the set of $N = 10$ equations the inertial motion model in Eq. (2.1) must be added to (2.4) and (2.6), or alternatively the model in Eq. (2.2) can be used with a curvilinear reference system, without affecting the dynamics equations.
The four wheels dynamical model will be used for simulation purposes in Ch. 4.

2.2.2 Two Wheels Dynamical Model

In this section a simplification of the four-wheeled vehicle in Sec. 2.2.1 in introduced. This simplification is obtained from the following

Assumption 1 *Assume a lumped wheel model, i.e. the front, left and right, and rear, left and right, wheels are combined in two wheels centered in the front and rear axles, respectively.*

The nonlinear model, depicted in Fig. 2.6, can be summarized as follow:

$$\dot{\xi} = f^{2w}(\xi, u), \tag{2.12}$$

where $\xi \in \mathbb{R}^N$ is the state with $N = 8$, whereas $u \in \mathbb{R}^m$ with $m = 3$ is the control action. Specifically, the state is the combination of longitudinal and lateral velocity in the body frame, v_x and v_y, the yaw ψ and the yaw rate r, the longitudinal and lateral coordinates in the inertial frame, X and Y, along with the speed of the two lumped wheels ω^f, ω^r, where the superscript differentiate between front and rear wheels. Moreover, the input is the combination of the steering angle δ and the two lumped wheels torques T^f, T^r. A positive torque denotes acceleration and might be available only for driving wheels, whereas negative torque, i.e. braking torque is available for all the wheels, in general. In the following the general superscript $\{i\}$, with $i \in \{f, r\}$, can be used to generalize wheel's notation. With the Assumption 1 in mind, the Eq. (2.4) can be simplified in:

$$\begin{aligned} m\dot{v}_x &= mv_y r + 2F_x^f + 2F_x^r, \\ m\dot{v}_y &= -mv_x r + 2F_y^f + 2F_y^r, \\ I_z\dot{r} &= 2l_f F_y^f - 2l_r F_y^r, \end{aligned} \tag{2.13}$$

where m is the vehicle mass, I_z is the vehicle inertia about the z-axis, l_f and l_r are the distances from the CoG to the front/rear wheel and finally (F_x^i, F_y^i) are the tire forces components, in the body frame. Tire forces can be written as:

$$\begin{aligned} F_x^i &= F_l^i \cos\delta_i - F_c^i \sin\delta_i, \\ F_y^i &= F_l^i \sin\delta_i + F_c^i \cos\delta_i, \end{aligned} \tag{2.14}$$

where F_l^i and F_c^i are the wheel forces components in the i tire frame. The wheel dynamics are:

$$\begin{aligned} I_w\dot{\omega}^f &= -F_l^f R_w + T^f, \\ I_w\dot{\omega}^r &= -F_l^r R_w + T^r, \end{aligned} \tag{2.15}$$

where I_w is the wheel inertia and R_w is the wheel radius. The tire forces F_l^i and F_c^i in Eq. (2.14) are given by the non-linear function:

$$\begin{aligned} F_l^i &= f_l(\lambda^i, \mu, F_n^i), \\ F_c^i &= f_c(\alpha^i, \mu, F_n^i), \end{aligned} \tag{2.16}$$

where λ^i and α^i are the longitudinal and lateral slip respectively, μ is the road grip coefficient and F_n^i is the normal force acting on the tire.
The nonlinear function in Eq. (2.16) are modeled by using the well-known Pacejka magic formula [4], similarly to what is done for Eq. (2.7). First define the tires' slips as:

$$\begin{aligned} \lambda^i &= \frac{\omega^i R_w - v_l^i}{\max(\omega^i R_w, v_l^i)}, \\ \alpha^i &= \arctan\left(\frac{v_c^i}{v_l^i}\right), \end{aligned} \tag{2.17}$$

where v_l^i and v_c^i are the tire speeds components in the i tire frame, as:

$$\begin{aligned} v_l^i &= v_y^i \sin\delta_i + v_x^i \cos\delta_i, \\ v_c^i &= v_y^i \cos\delta_i - v_x^i \sin\delta_i, \end{aligned} \tag{2.18}$$

where v_x^i and v_y^i are the tire speeds coordinates in the body frame, evaluated like in the following:

$$\begin{aligned} v_y^f &= v_y + l_f r, \\ v_y^r &= v_y - l_r r, \\ v_x^i &= v_x. \end{aligned} \tag{2.19}$$

Finally the normal forces acting on the wheels are:

$$\begin{aligned} F_n^f &= \frac{l_r mg}{2(l_f + l_r)}, \\ F_n^r &= \frac{l_f mg}{2(l_f + l_r)}, \end{aligned} \tag{2.20}$$

where $g = 9.81 m/s^2$ is the acceleration of gravity.
The same Remark 1 applies for the two-wheeled model as well.
In order to complete the set of $N = 8$ equations the inertial motion model in Eq. (2.1) must be added to (2.13) and (2.15), or alternatively the model in Eq. (2.2) can be used with a curvilinear reference system, without affecting the dynamics equations.
The two wheels dynamical model will be used for estimation purposes in Ch. 4.

2.2.3 Two Wheels Kinematic Model

In this section the kinematic bicycle model is introduced, referred as *kinematic* because it ignores masses and forces applied to the body, thus describing the vehicle motion only from a geometric standpoint.
In order to retrieve the model, consider the following, to be added to Assumption 1:

Assumption 2

(i) Assume no-slip condition, i.e. there is no longitudinal or lateral wheel slips, thus the wheel velocity and direction of movement are always aligned

(ii) Assume the body reference frame centered in the rear axle

(iii) Assume that the only available steering input is on the forward wheel, i.e. $\delta_r = 0$.

The aforementioned simplifying assumptions let us ignore the wheel dynamics, as well as, simplify the chassis dynamics as [5, 6]:

$$\begin{aligned} \dot{X} &= v \cos \psi, \\ \dot{Y} &= v \sin \psi, \\ \dot{\psi} &= v \frac{\tan \delta}{l_f + l_r}. \end{aligned} \tag{2.21}$$

The inputs to the system, as defined in most of the literature [5, 6], are the velocity v and the steering angle δ. However, in the following the Eq.

(2.21) will be augmented with the following:

$$\begin{aligned} \dot{v} &= a \\ \dot{a} &= \frac{a_{cmd} - a}{\tau}, \\ \dot{\delta} &= \delta_{cmd}, \end{aligned} \tag{2.22}$$

where a is the vehicle acceleration, a_{cmd} is the acceleration command, δ_{cmd} is the steering rate and τ is the power-train time constant, i.e. the parameter characterizing the response to a step input, when modelling the power-train as a first order linear system.
The combination of Eq. (2.21) and (2.22) will be referred as *kinematic bicycle model*, with abuse of notation. Such model is obtained assuming that a low-level controller is regulating the actuation system, resulting in a closed loop linear system.
Additionally, if a curvilinear reference system is taken as inertial frame, Eq. (2.21) can be rewritten as:

$$\begin{aligned} \dot{s} &= \frac{v \cos e_\psi}{1 - \rho e_y}, \\ \dot{e}_y &= v \sin e_\psi, \\ \dot{e}_\psi &= v \left(\frac{\tan \delta}{l_f + l_r} - \frac{\rho \cos e_\psi}{1 - \rho e_y} \right). \end{aligned} \tag{2.23}$$

The kinematic bicycle model will be used for planning purposes in Ch. 5.

2.2.4 Model Predictive Control

Model Predictive Controller (MPC) is one of the most advanced process control methodologies, based on an iterative solution of an optimal control problem. It allows for regulating the process to a set-point while taking the state prediction into account over a short time horizon.
When compared to classical controllers, such as Proportional-Integral-Derivative controllers, MPC offers several advantages, such as optimizing the state and input, satisfying hard constraints and directly quantify performances into a cost function.
At each sampling time T_s, an optimal control problem is solved, thus ob-

taining the control sequence over the control horizon, however only the first element of such sequence is applied to the plant, and the problem is solved again at the following sampling time.
The most general formulation for the optimization problem is:

$$\begin{aligned} &\min_{x_{1,\dots,H_p}, u_{1,\dots,H_c}} && J(x_{1,\dots,H_p}, u_{1,\dots,H_c}) \\ &\forall k \text{ subject to} && x_{k+1} = f(x_k, u_k) \\ & && x_k \leq g(x_k, u_k) \\ & && u_k \leq h(x_k, u_k) \end{aligned} \tag{2.24}$$

where x_k is the state, u_k is the control signal, H_p and H_c are the prediction and control horizons respectively, J is the cost function, f is the state prediction function, finally g and h are the constraint functions on the state and input.
The problem in Eq. (2.24) results in a Linear MPC if the cost function J have a quadratic form, and f g and h are linear, otherwise it is a NMPC. Many variants of the MPC have been used in the AD literature at various control levels. Specifically the MPC can be used for controlling the actuators directly, for path or trajectory planning, or even to tackle both problems by integrating chassis and actuation dynamics in the nominal model prediction f.
In this work a Linear MPC will be used in Ch. 4 for vehicle following purposes, instead a NMPC is defined for trajectory planning in Ch. 5.

2.2.5 Responsibility Sensitive Safety

Responsibility Sensitive Safety (RSS) is a human understandable mathematical formulation of safety, proposed in [7]. It was developed with three goals in mind, namely: the model should be compliant with how humans interpret safety, the model should lead to effective driving policies, the model should be efficient, i.e. it must be easy to verify if an AV is complying to the model.
The idea behind RSS is to translate common sense driving rules in a set of mathematical equations, the rules being:

1. do not hit someone from behind,

2. do not cut-in recklessly,

3. right-of-way is given, not taken,

4. be careful of areas with limited visibility,

5. if you can avoid an accident without causing another one, you must do it.

By using the above, along a set of *reasonable assumptions* about the other vehicles, the AV acts carefully and responsibly.
As an example, consider two vehicles driving in the same direction along the same lane. According to the first rule, the rear vehicle must keep a safe distance d_{min} to the leading vehicle, evaluated as:

$$d_{min} = \left[v_r \rho + \frac{1}{2} \alpha_{max} \rho^2 + \frac{(v_r + \rho \alpha_{max})^2}{2\beta_{min}} - 2\frac{v_f^2}{\beta_{max}} \right]_+ , \tag{2.25}$$

where v_r and v_f are the rear and front vehicles speeds, respectively, ρ is the reaction time of the rear vehicle, α_{max} and β_{min} are the maximal acceleration and minimal braking capability of the rear vehicle, β_{max} is maximal braking capability of the front vehicle and finally the symbol $[]_+$ is used to saturate a negative result to zero. Equation (2.25) is built upon kinematic reasoning, starting from the measurable vehicle speeds, and hypothesis on non-predictable accelerations about road users.
The Eq. (2.25) intuitively means that, as long as the front vehicle breaks with maximum deceleration β_{max}, and the rear vehicle reaches a deceleration of β_{min} in ρ seconds, no collision will happen.
The real challenge of the RSS is to select the set of reasonable assumptions, which translates in the tuning parameters ρ, α_{max}, β_{min} and β_{max}. If these parameters are too restrictive, with the extreme case being $\beta_{max} = \infty$, normal traffic flow is impossible. On the other hand, if the assumptions are not restrictive enough, the risk of accidents increases.
For the complete mathematical formulation and scenarios description, please refer to [7]. In this work the RSS will be leveraged as safety monitor in Ch. 5.

2.3 Machine Learning

2.3.1 Markov Decision Process

A Markov Decision Process (MDP) is a discrete-time stochastic model, providing a framework for decision making problems. A MDP is formally described by the tuple:

$$< S, A, T, R > \tag{2.26}$$

where

- S is the state space, or the set of possible states $s_k \in S$, which can be either continuous or discrete,
- A is the action space, or the set of actions $a_k \in A$, which can be either continuous or discrete,
- $T(s_{k+1}|s_k, a_k)$ is the transition model, i.e. a probability function describing the transition from the previous state s_k to the next s_{k+1}, given an action a_k,
- $R(s_k, a_k)$ is the reward function, which assigns real values to transitions,

where k is the discrete time index. Despite the state transition $T(s_{k+1}|s_k, a_k)$ being partially stochastic, the Markov property holds for MDPs, i.e. the probability distribution of the next state is dependent on the previous state and action alone. For the aforementioned reason, the MDP has been a powerful tool for studying control and optimization problems in combination with dynamic programming [8, 9].
In such problems the goal is to find the optimal policy, either deterministic $a_k = \pi(s_k)$ or stochastic $a_k \sim \pi(a_k|s_k)$, i.e. the one that maximizes the discounted cumulative reward:

$$R^\pi = \sum_{k=0}^{\infty} \gamma^k R(s_k, a_k), \tag{2.27}$$

where γ is the discount factor. The policy performances are measured using the value function:

$$V^\pi(s_k) \coloneqq \mathbb{E}\left[R^\pi | s_0 = s_k, \pi\right], \tag{2.28}$$

which defines the *expected* return of being in a state, for all the actions. For small and known MDP models, the exact optimal policy can be found offline by using dynamic programming. One way of achieving this is the Value Iteration algorithm [10], in which the Bellman operator is iteratively applied for all the states to the value function, thus obtaining the recursive form:

$$V_{i+1}(s_k) = \max_a \left[R(s_k, a) + \gamma \sum_{s_{k+1}} V_i(s_{k+1}) T(s_{k+1}|s_k, a) \right], \tag{2.29}$$

where the subscript i refers to the iteration step. As the iteration number increases, the Eq. (2.29) reaches the optimal value $V^*(s)$, and the optimal policy $\pi^*(s_k)$ can be obtained as:

$$\pi^*(s_k) = \operatorname*{argmax}_a \left[R(s_k, a) + \gamma \sum_{s_{k+1}} V^*(s_{k+1}) T(s_{k+1}|s_k, a) \right], \tag{2.30}$$

However in many problems of interest, and specifically in AD, either the state and action spaces are too large, or the transition model and reward function are unknown, thus the Value Iteration easily becomes intractable or non applicable. For the aforementioned reason, more complex solutions are needed for the AD problem.

2.3.2 Reinforcement Learning

Reinforcement Learning (RL) is a general framework of machine learning in which an agent learns online, how to behave in an unknown environment, in order to maximise the cumulative reward, as defined in Eq. (2.27).
The environment is typically modeled as a MDP (see Eq. (2.26)) in which the transition model is unknown. At each step, the agent chooses an action using the current policy, later it observes the new state and immediate reward. By collecting experiences as sequences of transitions, a new policy can be built at each learning step.
RL poses additional challenges when compared to standard supervised learning, in particular the collected data, i.e. the state transitions, depend

Figure 2.7. Schematic representation of a RL algorithm.

on the same policy which is being learnt.
RL algorithms can be broadly divided in two main classes: model-based and model-free. Model-based algorithms aim at estimating the transition model $T(s_{k+1}|s_k, a_k)$, later to be leveraged in a planning policy. On the other hand, model-free techniques are completely agnostic of the transition model, thus the goal is to directly find the optimal policy. Furthermore model-free algorithms can be divided in three classes: policy-based, value-based and actor-critic. Policy-based approaches aim to estimate the policy directly, whereas value-based estimate the value function (see Eq. (2.28)) first. Finally actor-critic leverages both policy and value-based ideas.
An important alternative to the value function, in Eq. (2.28), is the state-action value function, a.k.a. quality function or Q-function:

$$Q^{\pi}(s_k, a_k) = \mathbb{E}\left[R^{\pi}|s_0 = s_k, a_0 = a_k, \pi\right], \tag{2.31}$$

which measures the expected return for being in a state s_k and choosing the action a_k. The difference between the two is that in the Q-function the starting action is explicit and given, and from that point on, the policy π is followed.
If a Deep Neural Network (DNN) is used to perform the estimation, whether it estimates a policy or a value function, the framework takes the name of

Deep Reinforcement Learning (DRL).
For a comprehensive introduction on RL refer to [11] and reference therein.
In this work DRL will be leveraged in Ch. 5.

2.3.3 Deep-Q Learning

Deep-Q Learning (DQN) [12] is model-free and value-based DRL algorithm, which estimates the Q-function in Eq. (2.31) by using a neural network. The aim is to learn the optimal quality function, $Q^*(s_k, a_k)$ as:

$$Q^*(s_k, a_k) = \max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k) | s_0 = s_k, a_0 = a_k, \pi\right], \tag{2.32}$$

The core of the algorithm is based on the Bellman equation, which allows to recursively define the Q-function, as:

$$Q^*(s_k, a_k) = \max_{\pi} \mathbb{E}\left[R(s_k, a_k) + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1})\right], \tag{2.33}$$

which means that ones Q^* is learned, the optimal policy is the one that maximizes the Q-function, i.e. the *greedy policy*.
In DQN the Q-function is estimated by leveraging a DNN of weights θ, i.e.:

$$Q^*(s_k, a_k, \theta^*) \approx Q^*(s_k, a_k). \tag{2.34}$$

The weights θ are optimized at each iteration by minimization of the temporal difference error in the Bellman equation, i.e. the loss is:

$$L(\theta) = \mathbb{E}[(R(s_k, a_k) + \gamma \max_{a_{+1}} Q(s_{k+1}, a_{k+1}, \theta^-) - Q(s_k, a_k, \theta))^2], \tag{2.35}$$

where θ^- are the weights on the previous iteration. In practical implementation the weights θ^- are kept constant for multiple iterations to stabilize the training process.
The training process is carried out by saving transitions, at each simulation step, as tuples (s_k, a_k, r_k, s_{k+1}) in a matrix called *experience replay*. In order to reduce temporal correlation, at each training step, a set of random experiences is sampled from the replay and the loss function in Eq. (2.35) is built and minimized.

During recent years a number of improvements have been proposed over the vanilla DQN [12], aiming at stabilizing the training or improving performances. Examples are Double DQN [13], and Dueling DQN [14]. The original DQN psudo-code is reported in Alg. 1, for a comprehensive list of improvements and modifications, refer to [15]. In this work the vanilla DQN will be employed for designing planning policies in Ch. 5.

Algorithm 1 Deep-Q Learning with Experience Replay, [12]

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_k
    Otherwise select $a_k = \max_a Q(s_k, a, \theta)$
    Execute action a_k in emulator and observe reward r_k and state s_{k+1}
    Store transition (s_k, a_k, r_k, s_{k+1}) in D
    Sample random minibatch of transitions from D
    Set $y_k = \begin{cases} r_k & \text{for terminal } s_{k+1} \\ r_k + \gamma \max_a Q(s_{k+1}, a, \theta) & \text{for non-terminal } s_{k+1} \end{cases}$
    Perform a gradient descent step on loss in Eq. (2.35)
  end for
end for

2.3.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [16] is a model-free and policy-based DRL algorithm, which directly estimates the optimal policy π^* by using a DNN with weights θ. The loss function to be minimized is:

$$L^{CLIP}(\theta) = \mathbb{E}[\min(r_k(\theta)\hat{A}_k, \text{clip}(r_k(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_k)], \tag{2.36}$$

where $\hat{A}_k = Q(s_k, a_k) - V(s_k)$ is the estimated advantage function, ϵ is the clipping hyper-parameter and $r_k(\theta)$ is the probability ratio:

$$r_k(\theta) = \frac{\pi_\theta(a_k|s_k)}{\pi_{\theta^-}(a_k|s_k)}. \tag{2.37}$$

Intuitively, the minimum function in Eq. (2.36) clips the loss so that consecutive updates are not too drastic on θ, thus stabilizing the learning process. Such formulation was proposed in [16] as an alternative to the TRPO algorithm [17], which solves the same issue through constraints, thus solving a nonlinear constrained problem at each step, which is computationally expensive. Despite its simpler formulation when compared to TRPO, PPO achieves similar performances in most benchmark problems [16].
The advantage function estimation is generally obtained by using the Generalized Advantage Estimation (GAE) [18], namely an additional DNN is used. If the GAE weights are shared with θ the losses must be combined in:

$$L^{CLIP+VF+S}(\theta) = \mathbb{E}[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s_k)] \tag{2.38}$$

where c_1 and c_2 are coefficients, S is an entropy bonus, $L^{VF}(\theta)$ is the value function error squared loss. The PPO psudo-code is reported in Alg. 2, refer to [16] for the complete description of the algorithm. In this work the PPO algorithm will be employed for designing planning policies in Ch. 5.

Algorithm 2 Proximal Policy Optimization, [16]

```
for iteration = 1, M do
  for actor = 1, N do
    Run policy π in the environment for T time-steps
    Compute Advantage estimates Â_k
  end for
  Optimize loss in Eq. (2.38), with K epochs and minibatch size M ≤ NT
end for
```

Chapter 3

A Forward-Collision Warning System for Electric Vehicles: Experimental Validation in Virtual and Real Environment

In this chapter the development and testing of a Forward Collision Warning (FCW) system is presented. The aim is to leverage an affordable sensing system for ADAS development, so that it can be deployed in class L7E vehicles. Moreover a model-based integrated co-simulation platform is used to ease the system design and testing.

3.1 Motivation and Related Works

The FCW is nowadays considered a state-of-the-art system. Its aim is to promptly warn the driver when a collision is imminent, thus it has been proven effective in reducing rear-end crashes.
Nonetheless, nowadays only high-end vehicles are provided with such feature, because of the prohibitive sensor costs, typically RADARs and LIDARs. For this reason, a different design is needed to make it more affordable. This problem can be tackled from two sides: first of, the expensive sensors are replaced with much cheaper ones, such as cameras, second, the development burden can be lowered by appropriately design co-simulation

platforms, so that testing can be done during the early design stages.
Clearly the use of camera sensors requires computer vision algorithm to be used. State-of-the-art designs leverage geometric or appearance features to detect vehicles, e.g. by employing the Sobel edge detector [19, 20], or Haar-like feature detector [21, 22]. More recent works employ ML techniques, such as Support Vector Machines (SVM) [23], Hough Forest [24]. Nonetheless all of the above requires fine tuning of the features for each detected class of objects, thus in this work DL techniques [25, 26, 27] are employed to further improve the performance and reduce the development time. The employment of DL architectures allows for automation of features identification, thus as new labeled data are available, additional classes can be detected.

3.2 Forward Collision Warning

The FCW aim is to classify possible dangerous situations and promptly alert the driver through acoustic, visual or tactile warnings. In particular it is used for possible frontal collision such as rear-end collisions or vehicle-pedestrian collisions. Common causes for such scenarios are driver distraction or altered state, tailgating or panic-stop from the leading vehicle. Despite the fact that FCW is a SAE Level-0 system, i.e. it only provides warnings to the driver and it does not takes control over the vehicle actuation systems, it has been proven effective in completely avoid collisions or, at least, reducing their severity by warning the driver.
Most of the existing systems leverage RADAR technologies, thus resulting in high system cost. The aim of this work is to design the system in a scalable and affordable manner, both in terms of hardware and software, therefore even low-end vehicles can make use of such architecture.
The main idea is to employ a low-end front facing monocular camera in order to build the sensing state. Moreover, through the employment of DNN and an integrated co-simulation platform, software development and testing can be greatly eased, thus reducing software integration time.
In order to measure the collision risk a widely-adopted index is the Time To Collision (TTC) [28], as:

$$T = \frac{d}{\Delta v}, \tag{3.1}$$

where d is the distance to the leading obstacle and Δv is the relative longitudinal velocity. Despite the fact that such information is not directly measured in the camera 2D space, it will be shown how to derive the TTC information by leveraging the scale change of objects projected onto the image frame.
The main components of the FCW are object detection, object tracking and risk evaluation. Each of the aforementioned modules design is shown in the following sections.

3.2.1 Object Detection

Object Detection is a computer vision task which deals with detecting instances of classes in an image frame. In the following the term *obstacle* will refer to four possible classes equivalently, namely vehicle, pedestrian, bicycle and motorcycle.
The object detection problem can be tackled by using classical image processing techniques or, more modern, learning-based techniques. The classical computer vision solutions requires the engineer to hand-craft a set of features for each class of detected objects, e.g. through edge-detection, corner-detection etc. Despite their simple implementation and testing, those technologies are nowadays considered obsolete, because they suffer of the scalability problem (imagine a new class needs to be detected or new features need to be added to existing classes) and their performances are in general poor when compared to state-of-the-art learning-based solutions.
More recently learning-based solutions have been proven greatly effective in tackling the scalability problem. DNNs have replaced classical computer vision methodologies in every image processing task, such as classification, segmentation and object detection. The power of DNN is relative to the fact that the features are not hand-crafted, but learnt from data. Moreover DNNs scales better with more data when compared to classical ML. But from great power comes great computational cost, thus usually more expensive hardware platforms are required.
In this thesis a state-of-the-art Convolutional Neural Network (CNN) is employed , namely You Only Look Once (YOLO) [29]. YOLO is a single stage detector, meaning it predicts objectness, i.e. the probability that an object exists, and its class in a single stage, by incorporating the two tasks in a single loss function. It has been chosen in this work because it

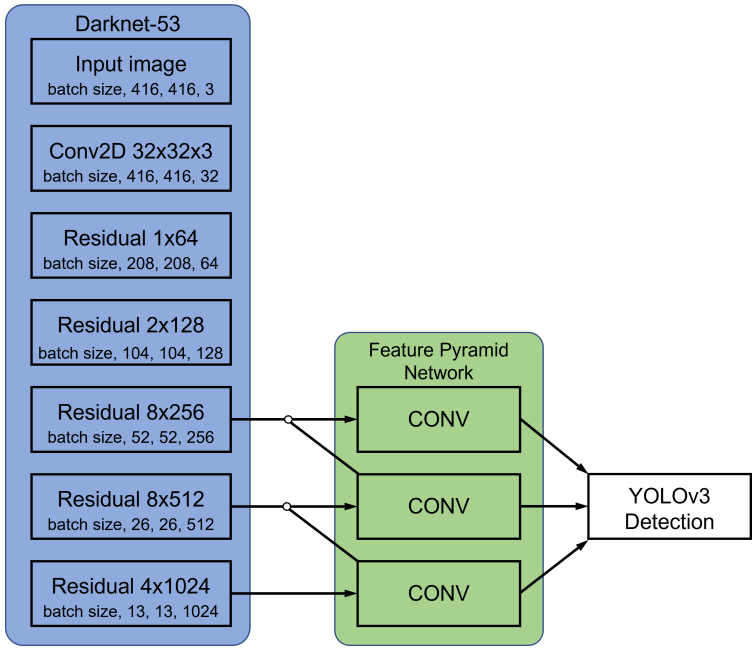


Figure 3.1. YOLOv3 network architecture scheme.

achieves faster computational time when compare to two stage detectors, such as R-CNN [30] or Faster R-CNN [31], still achieving comparable mean Average Precision.

The third version YOLOv3 uses Darknet-53 as CNN backbone to extract features from the input image. A Feature Pyramid Network (FPN) follows the backbone. The input to FPN are the features coming from different depths of the backbone, allowing YOLO to address the multiscale problem, by combining low resolution (semantically strong) features with high resolution (semantically weak) features using top-down pathways and lateral connections. A simplified scheme of YOLO architecture is given in Fig. 3.1. The output of detection system is the, so called, bounding-box, identifying the objects position in the image space as the four dimensional vector:

$$b = [b_x \; b_y \; b_w \; b_h] \tag{3.2}$$

where b_x and b_y are the bounding box top-left corner pixel coordinates, whereas b_w and b_h are the bounding box width and height, respectively.

3.2.2 Multi-Target-Tracking

In order to build a time history of each detected object, a tracking module is needed. Moreover, taking into account that for each frames multiple detections exist, a Multi Target Tracking (MTT) algorithm is needed.
The aim of this module is to associate each new incoming detection to existing tracks, or create new ones if no association is found.
Designing such module is challenging due to multiple reasons, namely sensor detection probability, occlusion, unknown number of detections at any time step. All of the aforementioned problems can lead to both false positives and negatives. Nonetheless, being the FCW a Level-0 ADAS, false positives can be tolerated because the actuation control is still responsibility of the driver.
In this work the Global Nearest Neighbour (GNN) algorithm [32] is chosen as MTT module. The GNN is a single hypothesis tracker, i.e. each detection is assigned to the global nearest track. Due to the fact that conflicts can occur in the association step, a cost function must be defined.
The Intersection Over Union (IOU) ones' complement, between detection and track pairs, is chosen as cost function:

$$J(i,j) = 1 - \mathrm{IOU}(d_i, t_j), \tag{3.3}$$

where d_i is the i-th detection and t_j is the j-th track, whereas the term $\mathrm{IOU}(d_i, t_j)$ is the IOU between the couple (i,j), i.e. the extent of overlap between the two corresponding boxes. The ones' complement is needed so that a minimization problem can be built (the IOU is always between zero and one), otherwise a maximization problem must be solved.
At each sample time a tabular optimization problem is solved and the best (i,j) couples are found, by minimization of the total cost in Eq. (3.3).
The Munkres algorithm [33] is chosen to solve the optimization problem, which ensures convergence in polynomial time. Moreover, in order to reduce the problem complexity, first a gating step is applied, which filters out unlikely detection-tracks association by setting a threshold cost. Due to the small number of detection and tracks (typically below twenty) this

set-up can be solved in real-time on the chosen deployment hardware platform.
Once the association problem is solved, the detection measurements are used to update the tracks, i.e. a bank of constant-velocity linear Kalman Filter (KF) [34]. The KF state is the eight-dimensional vector:

$$x = [b_x \;\; b_{x_v} \;\; b_y \;\; b_{y_v} \;\; b_w \;\; b_{w_v} \;\; b_h \;\; b_{h_v}]\,, \tag{3.4}$$

where b_{x_v} and b_{y_v} are the bounding box top-left corner velocities, whereas b_{w_v} and b_{h_v} are the bounding box width and height velocities, respectively. It is worth to point out here that the KF state is in the 2D image frame which greatly simplifies the problem of measuring three dimensional coordinates from a monocular camera. Retrieving 3D coordinate from the 2D pinhole model is not possible in general. Despite the fact that, by making a simplifying assumption and knowing the camera calibration with high accuracy, this limitation could be overcome, it is chosen here to avoid the problem completely by resorting to scale change in the 2D space, thus improving the final robustness of the algorithm. In the following section additional details on this matter are given. Regarding the KF matrices, the state matrix is easily built by taking into account that the vector in Eq. (3.4) is made of independent position-speed couples. On the other hand the noise matrices have been tuned by trial and error, in particular by fixing the process noise matrix and slowly tuning the measurement noise matrix.

3.2.3 Collision Risk Evaluation

Once the track are updated, the collision risk can be evaluated for each one of them. In the following it is shown how to link the TTC in Eq. (3.1) to the scale change of bounding boxes in consecutive measurements.
The width of an obstacle is projected in the image space by using the pinhole camera model [5]:

$$w_i = \frac{fW}{d_i}, \tag{3.5}$$

where w_i is the obstacle width in the i-th image frame, W is the obstacle width in the three dimensional space and f is the camera focal length.By tracking the objects between two frame i and $i+1$, the scale change is

defined as:

$$S = \frac{w_i}{w_{i+1}} = \frac{d_{i+1}}{d_i}. \tag{3.6}$$

Due to the fact that time interval between the two frames Δt is small ($\simeq 1/30s$), constant relative velocity is assumed here, hence:

$$d_{i+i} = d_i + \Delta v \Delta t. \tag{3.7}$$

By substitution of Eq. (3.7) into Eq. (3.6), the following is obtained:

$$S = \frac{d_i + \Delta v \Delta t}{d_i}, \tag{3.8}$$

and hence, from Eq. (3.1), the TTC can be written as:

$$\widehat{T} = \frac{\Delta t}{S - 1}, \tag{3.9}$$

where $\widehat{T}$ is the estimation of T in Eq. (3.1), obtained by using image frames only.
It is pointed out here again, that $\widehat{T}$ in Eq. (3.9) is independent of the actual distance between the obstacle and the camera, thus it allows to ignore camera calibration and road properties, such as flatness, banking and slope angles. The total accuracy of the estimation of Eq. (3.9) is mainly related to the detection and tracking accuracy and to the choice of Δt. In particular, by increasing Δt, the input noise from the detection system can be attenuated but less samples are available to update the KF states which can lead to false negatives. Discussions on the theoretical bounds of Eq. (3.9) accuracy can be found in [35].
If $\widehat{T}$, as evaluated in (3.9), is below a chosen threshold (usually between two and three seconds), a collision might occur. A warning is issued only if the obstacle is on the ego-vehicle path. In order to check such condition the KF states can be leveraged again, considering that they contain information about the velocity of the bounding box. In particular, the future position of the bounding box can be predicted as:

$$b_{x_{pr}} = b_x + \widehat{T} \cdot b_{x_v}, \tag{3.10}$$

where $b_{x_{pr}}$ is the predicted abscissa of the top left of the bounding box. Using the same kind of reasoning, the right corner can be predicted by using the width of the box. If the predicted box is inside a pre-designed region of the image frame the warning is issued. The Eq. (3.10) is obtained from the constant velocity assumption which results in a good approximation in the scenarios of interest, such as rear-end collisions.

3.3 Validation and Discussion

In this section the testing results are shown. The effectiveness of the approach is first numerically evaluated in a co-simulation platform, through Model In the Loop (MIL) testing. A well design platform can assist the software development from the first design phase to the final coding and testing, hence easing the whole process. Next, Vehicle In the Loop (VIL) testing results are shown. The FCW has been deployed on the embedded platform and tested on an experimental L7E class electric vehicle.

3.3.1 Model-in-the-Loop Testing

In order to validate driving assistance systems, it is required to drive in potentially dangerous scenarios, such as near-miss collisions. For this reason it is advisable to first test functionalities in a purposely designed co-simulation platform, in which the environment can be controlled in terms of road network, road users behaviour, environment conditions etc. This first round of testing allows practitioners to tune controllers and remove bugs by isolating software defects.
In this work the co-simulation platform has been built by leveraging two main components:

- Matlab/Simulink has been used to ease the algorithm coding by first using a high level programming language which can be later automatically translated in Embedded C through the Embedded Coder toolbox

- the open-source AV simulator CAR Learning to Act (CARLA) [36] has been used to design traffic scenarios and simulate synthetic sensor measurements, i.e. monocular camera image frames.

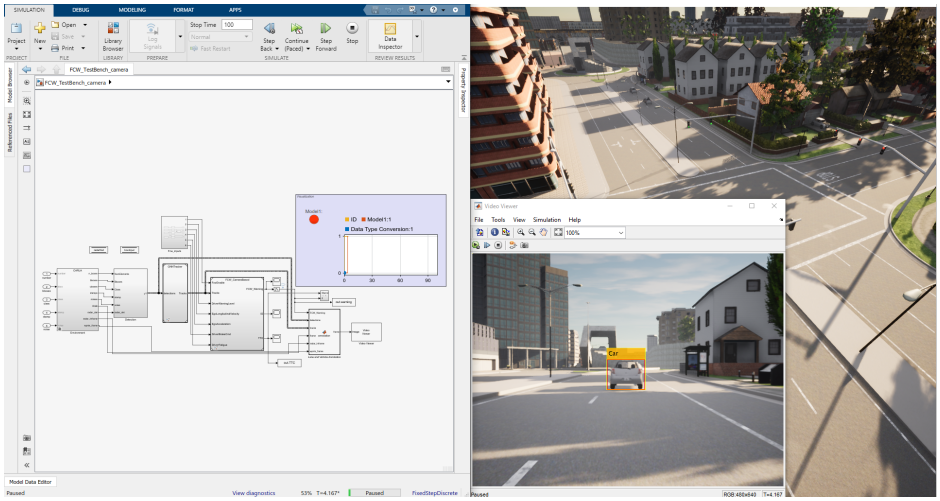


Figure 3.2. Screenshot of the co-simulation platform during a use case. On the left the Simulink window, in which the core FCW is implemented along with the APIs calling the simulator server. On the right, the Carla server window running the road environment and creating synthetic sensor measurements.

CARLA is developed as an open-source layer on top of Unreal Engine, thus providing state of the art render quality. It works in client-server fashion, where the server runs the scenarios and create the simulated measurements, whereas the client is a set of Python API which can be used to retrieve the generated data.
On the other end, Matlab can be used to call Python scripts as well, thus enabling us to link the two main components, by using is as a client for CARLA. Additionally the FCW has been implemented in Matlab, which can be leveraged to generate embedded C code at a later stage. Figures 3.2 and 3.3 show screenshots of the proposed co-simulation platform during a use case. Scenarios prescribed by EuroNCAP [37] have been designed in the simulation platform, namely:

- Car to Car Rear Stationary (CCRS): the ego vehicle moves towards a stationary leading vehicle,
- Car to Car Rear Moving (CCRM): the ego vehicle moves towards a

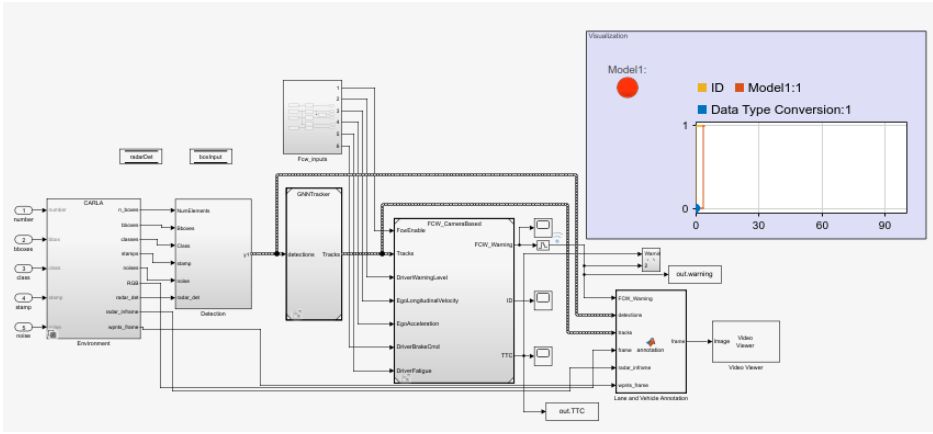


Figure 3.3. Detail of the co-simulation platform. From the left, the CARLA module to interact with the server, the detection module, tracking module, the inputs module, which emulates possible user configurations, and finally the collision risk evaluator module.

slower constant velocity moving vehicle,

- Car to Car Rear Braking (CCRB): the ego vehicle moves towards a braking vehicle.

All the aforementioned scenarios are repeated by varying vehicles velocities and lateral overlap ranging from -50% to $+50\%$, as prescribed from the safety protocol.
Results are shown for two exemplary scenarios, namely:

- scenario 1: CCRS with starting distance $d \simeq 67m$ and the ego vehicle traveling at constant speed $v = 13.9m/s \simeq 50km/h$,

- scenario 2: CCRM with starting distance $d \simeq 30m$, ego vehicle and leading vehicle traveling at constant speeds $v = 13.9m/s \simeq 50km/h$ and $v = 5.55m/s \simeq 20km/h$, respectively.

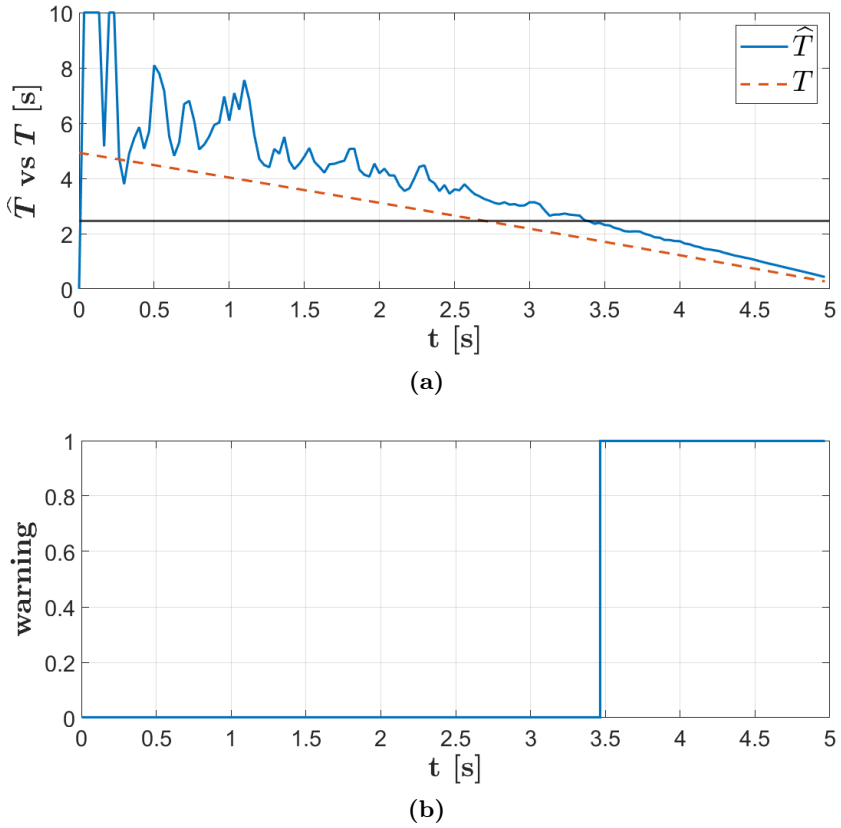


Figure 3.4. Simulation results, through MIL testing, for Scenario 1. **(a)** Time-history of the estimated TTC, $\widehat{T}$, and the real TTC, T. The warning threshold is shown as constant black horizontal line. **(b)** Time-history of the FCW activation.

Scenario 1: Results are shown in Figs. 3.4a and 3.4b. The estimated TTC is compared to the real one in Fig. 3.4a in order to assess the accuracy of the algorithm. In the proposed scenario, the relative velocity Δv is constant, thus the TTC decreases linearly with time. Despite the lack of accuracy at high distances, the estimated TTC is accurate enough in the range of interest, i.e. below $4s$, which results in both zero false positives and negatives. Moreover, from Fig. 3.4a, a small constant positive percentage bias can be appreciated in the estimation. This is due to the constant

distance between the monocular camera mounting position and the front bumper of the ego-vehicle, where the actual TTC should be taken. Note that, in order to compensate for this bias, the distance to the front obstacle should be measured [38], but this is not possible in the two dimensional image frame.
In Figure 3.4b the activation of the warning is shown, which happens when $TTC \leq 2.2s$ for at least two out of three consecutive steps.

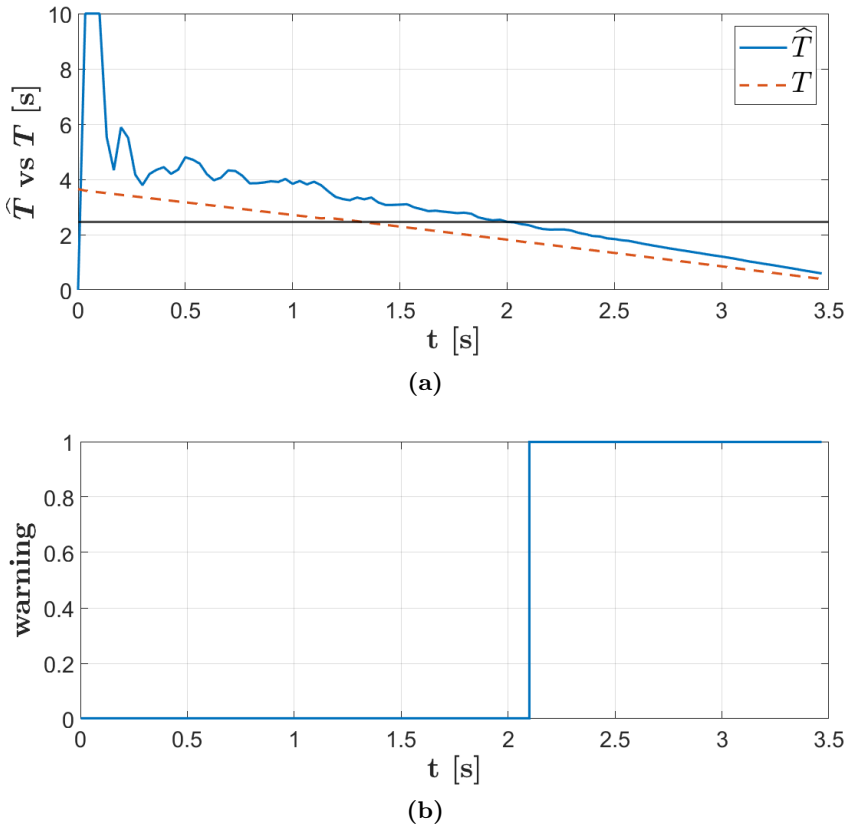


Figure 3.5. Simulation results, through MIL testing, for Scenario 2. **(a)** Time-history of the estimated TTC, $\widehat{T}$, and the real TTC, T. The warning threshold is shown as constant black horizontal line. **(b)** Time-history of the FCW activation.

Scenario 2: Results are shown in Figs. 3.5a and 3.5b. Similarly to Scenario 1, the estimated TTC is compared to the real one in Fig. 3.5a. Results disclose that the lack of accuracy at high distances does not lead to false positives, and the estimation is accurate enough below $4s$ so that no false negatives are reported. The same results were obtained by varying lateral overlap and longitudinal velocities. Figure 3.5b shows the activation of the actual warning. The outcome of CCRM scenarios leads to similar conclusion taken from CCRS scenarios.

3.3.2 Vehicle-in-the-Loop Testing

Despite the practical convenience of simulation-based testing, it is hard to simulate the complexity of real scenarios. For this reason in-vehicle testing has been carried out in order to validate the entire set-up with respect to the embedded hardware platform and real challenges that may arise, such as low visibility, road roughness etc.
The camera-based algorithm has been deployed on a NVIDIA Jetson AGX Xavier Developer board equipped with 8 CPU cores, 512 GPU cores and 32GB of RAM. The chose platform is capable of achieve 30fps, thus complying to real-time constraints. The object detection module is publicly available[1] implementation of YOLOv3. The open-source repository leverages CUDA and cuDNN libraries for the fastest possible CNN inference on GPU cores. The rest of the modules, i.e. MTT and risk assessment, have been coded in Matlab/Simulink for rapid prototyping and, at later stage, C code has been automatically generated by leveraging the toolbox Embedded Coder. The chosen camera sensor is a High Dynamic Range (HDR) 2MP Starlight Camera. The HDR technology, combined to the Ultra Low Light, allows us to capture images in harsh light conditions, hence enabling FCW also at night or inside tunnels.

[1]YOLOv3 code: https://github.com/AlexeyAB/darknet

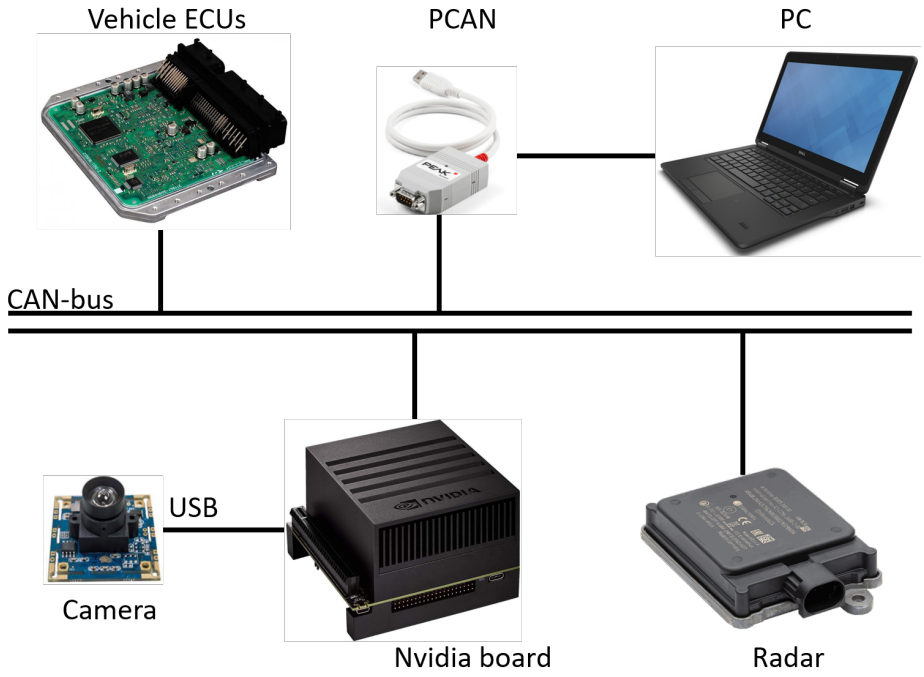


Figure 3.6. In vehicle experimental hardware platform employed during VIL testing.

During MIL testing one can use the simulated data to assess the algorithm performances, similarly to what has been done in Figs. 3.4a and 3.5a. However, this is not possible for VIL testing. For the aforementioned reason, a second highly reliable source of information is needed. The automotive RADAR ARS 404-21 from Continental has been used. RADARs allows us to measure directly distances and relative velocities to obstacles, hence Eq. (3.1) can be reliably used to measure TTC, in place of its estimation in Eq. (3.9).
The camera has been connected to the NVIDIA board directly through USB, whereas the RADAR has been connected to the vehicle Control Area Network (CAN)-bus. Finally by using a PCAN-USB from Peak System and a standard laptop, all the needed data can be collected. The schematics of the experimental hardware platform is shown in Fig. 3.6.

Figure 3.7. Exemplary image frame captured from the in-vehicle validation CCRS scenario. The yellow bounding box is the output of YOLOv3, whereas the red box is the filtered measure obtained from the KF.

Once again, CCRS scenarios have been carried out during experimental validation. The ego vehicle starts from standstill, then it accelerates up to $v \simeq 8.3m/s \simeq 30km/h$ (see Fig. 3.8b) towards a stationary obstacle, i.e. a vehicle in this case study. The FCW issues a warning when the TTC goes below a threshold, which was set to $2.45s$. The test is considered successful if the algorithm generates the alert soon enough to brake and avoid impact with the forward obstacle. Figure 3.7 shows an exemplary image frame captured during the experiments.
Figure 3.8a shows that the performance achieved by using a low-cost monocular camera are comparable to the one achieved with a RADAR. Despite the lower accuracy of the detection system when compared to the RADAR, it is still accurate enough for the FCW algorithm, i.e. neither false positives nor negatives are reported during the experiments, as shown in Fig. 3.8c. It is pointed out here that, Fig. 3.8b shows a deceleration before $t = 10s$, i.e. before the warning activation, this is only due to extra care taken by the safety driver to avoid potentially dangerous situations.

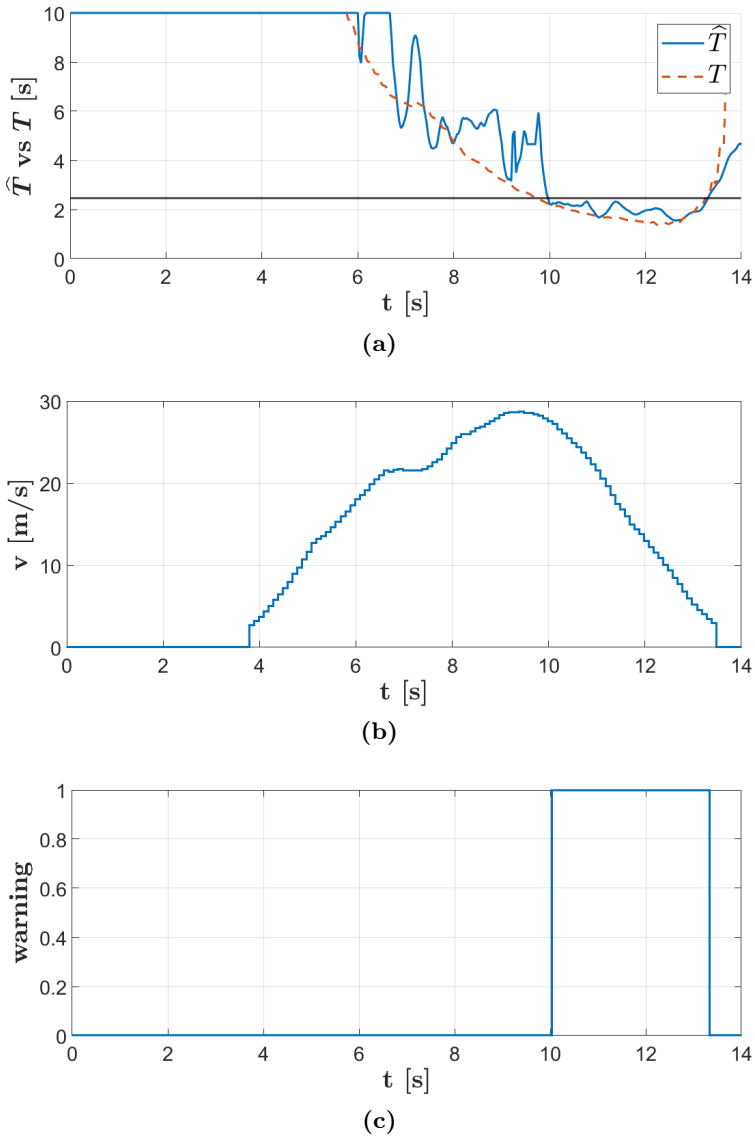


Figure 3.8. Experimental validation results in a CCRS scenario. **(a)** Time-history of the estimated Time-To-Collision, $\widehat{T}$, compared to the estimated through the RADAR sensor, T. The warning threshold is shown as a constant horizontal line. **(b)** Time-history of the ego-vehicle speed v. **(c)** Time-history of the FCW activation.

In conclusion, both MIL and VIL testing, disclose that a monocular camera based FCW achieves performances comparable to a RADAR based one. Moreover, by resorting to a camera space estimation of $\widehat{T}$, no calibration is needed, thus the system can be mounted and used effortlessly on any vehicle. These results are really promising because they enable low-end L7E vehicles with ADAS modules.

Chapter 4

On-Board Road Friction Estimation Technique for Autonomous Driving Vehicle-Following Maneuvers

In this chapter a control architecture responsible for vehicle longitudinal dynamics regulation, is shown. It includes the ACC and AEB, in addition to ABS. The aim is to improve state of the art longitudinal ADAS by embedding road-tire grip information in the control laws. This information can be retrieved on-board by using commonly available sensors in combination to model-based estimation methodologies. First the road-grip estimation technique is presented, followed by the integration of control designs and finally numerical validation results are shown.

4.1 Motivation and Related Works

Longitudinal ADAS modules such as ACC, AEB and ABS are nowadays considered state-of-the-art on most high end vehicles. All of the above are used as modular components leveraged to build a SAE Level 2 vehicle, i.e. partial control is achieved, hence the human driver must keep an eye on the road, in order to promptly intervene in potentially dangerous situations.

This limitation is due to the assumption under which these systems are designed, namely ideal road and environment conditions, such as a perfectly dry flat road, without local imperfections.
Nonetheless, these assumptions are rarely true, thus the nominal safety cannot be guaranteed. In order to increase the autonomy level, environment conditions must be explicitly taken into account since the design phase, so that the vehicle adapt its behavior to the real-time road state.
In this context the main contributor to the overall safety is the road-tire interaction, in terms of maximum available friction, which is the main factor for the maximum achievable deceleration in case of danger. On the emergence of dangerous situations, where extreme braking is required, the tire dynamics can be pushed to the unstable region, thus negating stability and safety.
However, directly measuring road-tire interaction requires expensive sensors, so in order to make the solution scalable to low-end vehicles as well, on-board estimation techniques are required.
For the aforementioned reasons, a combination of estimation methods and novel longitudinal ADAS designs are proposed here.
As specified in [39], friction estimation can be divided in two main groups, namely expertiment-based and model-based. The experiment based typically use expensive sensors, usually optical or acoustical, to directly measure road roughness [40]. Nonetheless common vehicles are not provided with such sensors, which makes the solution hard to scale. On the other hand, model-based solutions estimate the road state by using mathematical models and commonly available chassis sensors. In [41], the authors evaluate a set of parameters through experiments, for different conditions, to be later saved in the Electronic Control Unit (ECU) memory. At runtime, a switching logic selects the appropriate set of parameters, however the accuracy of such system is limited by the number of parameters stored in the memory. In [42] the friction estimation is carried online by implementing different control strategies on the front and rear tire, so that the total chassis speed is not affected. However, this cannot be achieved on most common vehicles and such control laws can damage the tires in the long run. Finally, in [43] the slip-slope methodology has been leveraged, which assumes a linear tire behaviour to estimate friction, however this model is not accurate in the non-linear region which is the most important

one when dealing with extreme acceleration and steering commands.
Regarding the ADAS modules design, the MPC has been successfully employed for the ACC [44, 45, 46, 47, 48], whereas event-based controllers are typically leveraged for the AEB [49, 50, 51]. However, none of the above explicitly take into account the road-tire interaction and maximum available braking, which is of utmost importance when dealing with dangerous scenarios.
With the aim in mind of increasing the SAE autonomy level, it will be shown how the estimated road-tire grip can be leveraged by longitudinal ADAS, so that safety can be guaranteed in adverse road state scenarios.

4.2 Control Architecture

Let us consider a front-driven vehicle equipped with proprioceptive sensors for the measurement of ego-state variable, e.g. speed, acceleration, yaw-rate etc., as well as exteroceptive sensors for environment sensing, e.g. camera, RADAR, LIDAR (details on AV sensor architectures can be found in [52]). The aim is to design a control architecture capable of performing vehicle-following maneuvers in a safe and comfortable fashion, despite any adverse road conditions, such as the presence of water, snow or ice on the road asphalt. In order to achieve that, the tyre-asphalt friction coefficient must be estimated on-line, in a computationally fast and reliable way.
To achieve the aforementioned capabilities the ACC has to adjust the distance to the leading vehicle by taking into account the road friction, in addition to the leading vehicle speed, as:

$$\begin{aligned} &d \to d_{des}(\hat{\mu}_{max}), \\ &\Delta v = v_{lead} - v \to 0, \end{aligned} \tag{4.1}$$

where d is the distance to the leading vehicle, $d_{des}(\hat{\mu}_{max})$ is the desired distance, $\hat{\mu}_{max}$ is the estimated road friction coefficient, Δv is the relative speed,and finally v_{lead} and v are the leading and ego-vehicle speed respectively.
In Eq. (4.1) the desired gap $d_{des}(\hat{\mu}_{max})$ can be set according to the well-

known headway time rule [53], as:

$$d_{des}(\hat{\mu}_{max}) = d_0 + \tau_H(\hat{\mu}_{max})v, \tag{4.2}$$

where d_0 is the constant standing-still spacing and $\tau_H(\hat{\mu}_{max})$ is the heading time. This value is tuned to trade off efficiency and safety in most practical applications and papers, usually by assuming a $\hat{\mu}_{max} \simeq 1$, but in this work it is adapted relatively to the estimated road-grip coefficient $\hat{\mu}_{max}$, as:

$$\tau_H(\hat{\mu}_{max}) = \begin{cases} \tilde{\tau}_H/0.2 & \hat{\mu}_{max} \leq 0.2, \\ \tilde{\tau}_H/\hat{\mu}_{max} & 0.2 < \hat{\mu}_{max} \leq 1, \\ \tilde{\tau}_H & \hat{\mu}_{max} > 1, \end{cases} \tag{4.3}$$

where $\tilde{\tau}_H$ is the constant headway for an ideal dry road [48]. The design in Eq. (4.3) results in increasing desired distance as the maximum available grip decreases, additionally saturating minimum and maximum range to avoid infeasibility.

In order to further enhance safety, ACC works jointly with AEB to avoid or mitigate longitudinal accidents. The AEB constantly monitors the area right in front of the vehicle by sharing the same sensing state used by the ACC. In case a possible or unavoidable collision is detected, the AEB issues a braking command (usually to the ABS), thus decelerating or stopping the vehicle. It follows that, in contrast to the ACC, the AEB is an asynchronous module, i.e. activated only on specific events. In this work the well-known TTC index [28] is leveraged, as:

$$T = \frac{d}{\Delta v} < T_{th}(\hat{\mu}_{max}), \quad \text{when } \Delta v < 0, \tag{4.4}$$

where $T_{th}(\hat{\mu}_{max})$ is the positive threshold at which AEB is activated. Similarly to the headway time in Eq. (4.2), the AEB threshold is adapted relatively to the estimated road condition so that road asphalt condition is taken into account.

If an emergency braking is issued, the maximum available torque is applied to the wheels by the ABS module. The longitudinal slip ratio is decreased fast, resulting in a braking force on the vehicle chassis. However, if the slip goes below the optimal value, the wheel dynamics will transition to

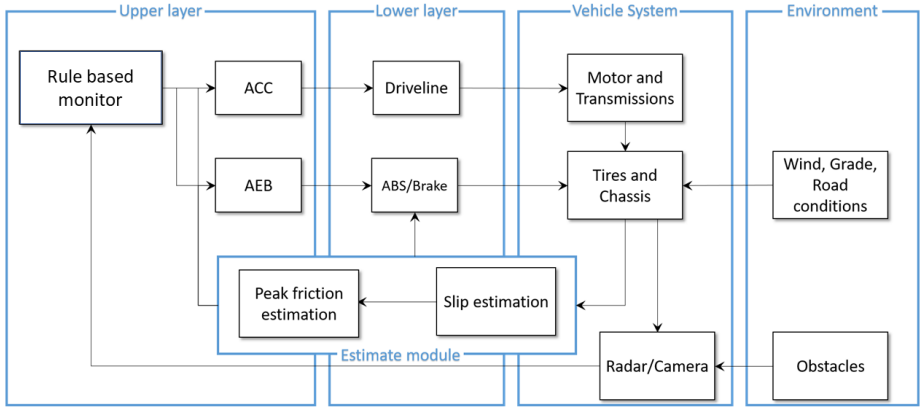


Figure 4.1. On-board ADAS control architecture

the unstable region, which will lead to wheel-locking, thus reducing the effectiveness of the braking system. This phenomenon is typically prevented by the ABS by periodically reducing the braking torque on the wheel if a locking condition is detected or predicted. Nevertheless, the optimal slip value is road-grip dependent, therefore its online estimation can greatly enhance safety and performance.

The road-grip aware ADAS functionalities are integrated in the control architecture in Fig. 4.1. The on-board road estimation module is capable of returning the estimation $\hat{\mu}_{max}$ online which is then exploited downstream by the ADAS modules, in order to enhance comfort and safety performances.

Note that the role of the rule-based scheduler module in Fig. 4.1 is to classify scenarios and behaviours and select the ADAS function accordingly [54].

4.3 In-Vehicle Road-Grip Estimation

4.3.1 From Vehicle Sensors to Tire's State

The number of sensor installed in the commercial vehicles has increase drastically in recent year. The presence of such low-cost sensor suite has transformed the vehicle in small mobile laboratories. Starting from physical measurements it is possible to build smart estimators on-board and in real-time, by leveraging ECU, which in turns, are becoming more affordable with time. Moreover, starting from global information about the chassis, it is possible to estimate the state of its sub-components, such as the tires.
Starting from the signals commonly available on the CAN-bus, the tire kinematics and dynamics can be estimated in real-time by employing existing computational platforms or small dedicated ones.
An example of such estimation process is given by the TRICK algorithm [55], which by leveraging CAN-bus signals and a quadricycle model, estimates tire dynamics in its complete state space.
Due to the fact that in this work vehicle following and emergency braking scenarios, i.e. only longitudinal movements, are under investigation, the quadricycle model can be simplified into the bicycle model, such as defined in Sec. 2.2.2. This simplified assumption allows us to greatly reduce the computational burden per step, as well as reducing the number of parameters needed to reproduce the state physical dynamics under estimation.
The obtained estimation algorithm, to be presented later, feeds the ADAS modules with the estimation of the current and potential road friction coefficients in real-time.
The following assumptions are made on the environment and on the model:

Assumption 3

(i) the road is modeled as flat without local imperfections, such as potholes or roughness, but potentially affected by the presence of water, ice or snow,

(ii) the tires are modeled by using the dynamic-kinematic functions, i.e. ignoring the transient. Moreover, multi-physical dynamics are neglected, such as thermal wear or degradation,

(iii) the steering command δ is assumed to be constant and null, due to the fact that only longitudinal dynamics are under investigation,

(iv) the vehicle is considered in terms of its global mass-inertia parameter. The longitudinal dynamics are described with reference of its CoG,

(v) the vertical load on each axle is evaluated in terms of the static load, load transfer and aerodynamic drag,

(vi) the suspension and steering system kinematics are taken into account as static curves acquired by means of a multi-body physical simulator.

The input to the simplified TRICK estimation methodology are the following signals to be acquired on-board using sensors or CAN-bus signals:

- wheel angular speed $\omega^i[rad/s]$,
- vehicle longitudinal speed $v[m/s]$,
- vehicle longitudinal acceleration $a[m/s^2]$,
- throttle pedal position, as a percentage,
- brake pedal position, as a percentage.

The estimation module outputs are:

- axle slip ratios λ^i,
- axle vertical load $F_n^i[N]$,
- axle longitudinal interaction force $F_l^i[N]$,
- axle current friction coefficient $\hat{\mu}^i$,
- axle potential friction coefficient $\hat{\mu}^i_{max}$,

where slips and forces are related to axles, instead of wheels, because of the bicycle model simplified assumptions, namely slip and forces on left and right tires are considered equal. The schematic model simplification employed by the estimator module is given in Fig. 4.2.

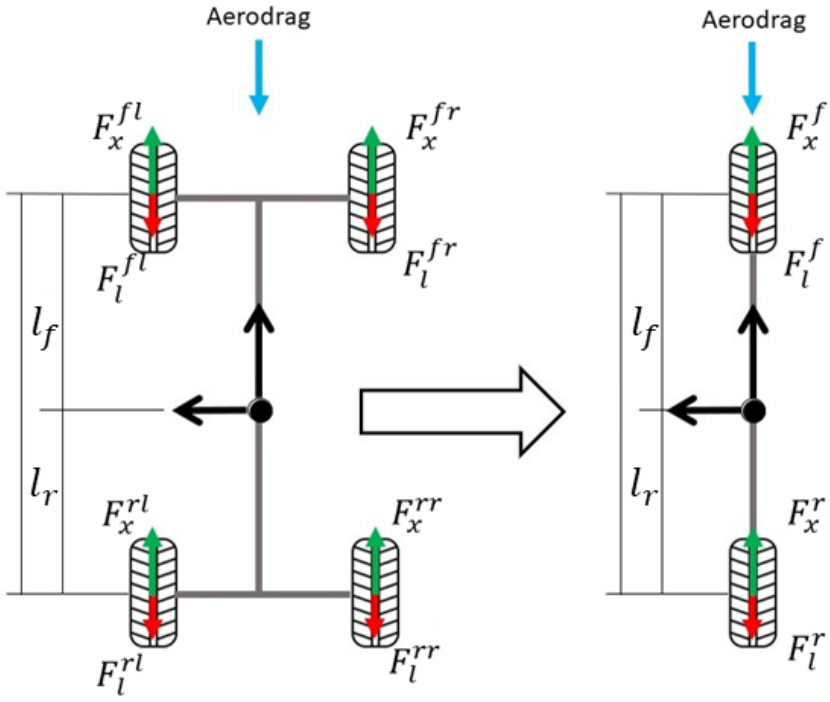


Figure 4.2. Friction estimator: Model simplification, from four wheels quadricycle to two wheels bicycle model.

The vertical static load acting on the axles is evaluated as in Eq. (2.20), restated here for completeness:

$$\begin{aligned} F_n^f &= \frac{l_r mg}{2(l_f + l_r)}, \\ F_n^r &= \frac{l_f mg}{2(l_f + l_r)}. \end{aligned} \tag{4.5}$$

In order to obtain a more accurate final prediction, in real applications the longitudinal transfer ΔF_n and the downforce F_z^D are added to the model in Sec. 2.2.2. In particular, the longitudinal load transfer is evaluated as:

$$\Delta F_n = \frac{mh\dot{v}_x}{l_f + l_r}, \tag{4.6}$$

with h being the CoG height, whereas the vertical aerodynamic downforce is estimated as follow:

$$F_z^D = \frac{1}{2}\rho A v^2 C_L, \tag{4.7}$$

where $\rho[kg/m^3]$ is the air density coefficient, $A[m^2]$ is the vehicle cross sectional area, C_L is the lift coefficient. The total axle load is given by:

$$F_z^i = -(F_n^i - \Delta F_n + F_z^D). \tag{4.8}$$

Moreover, the inertia effect on the axle is:

$$F_{inertia}^i = \frac{I_w \dot{\omega}^i}{R_w}, \tag{4.9}$$

where I_w and R_w are the wheel inertia and radius, respectively. The forces estimated in Eqs. (4.6)-(4.9) can be estimated by a combination of constant parameters and time-varying information commonly available on the vehicle CAN-bus network.
The longitudinal interaction force between the tire and the asphalt is a complicated function of vehicle speed, normal loads, wheel speed:

$$F_l^i = f_l(\lambda^i, \mu, F_n^i). \tag{4.10}$$

where the slip ratio λ^i is:

$$\lambda^i = \frac{\omega^i R_w - v_l^i}{\max(\omega^i R_w, v_l^i)}. \tag{4.11}$$

4.3.2 On-Board Friction Coefficients Estimation

It is widely known that tire behavior can vary drastically relatively to environmental global conditions, e.g. rain, snow or ice, or relatively to local road irregularities, e.g. oil, puddles or potholes. In order to enhance safety performances of ADAS modules in presence of adverse environmental conditions, it is of utmost importance to provide them the current road friction coefficient $\hat{\mu}$, i.e. the instantaneous tire-asphalt friction, and potential road friction $\hat{\mu}_{max}$, i.e. the maximum available grip achievable when forces are applied to the wheels, either accelerating or braking.

The first step is to evaluate the tire characteristic, i.e. Eq. (4.10), through real experimentation, on various types and states or the road asphalt. Starting from the pre-calibrated set of parameters and the on-line current road grip coefficient, it is possible to estimate the potential friction in a later step. It is assumed here, that the potential friction can be obtained by varying the longitudinal wheel slip alone, and keeping the other information constant, such as vehicle speed, normal loads, wheel alignment among others. The typical tire characteristic is given in Fig. 4.3, where

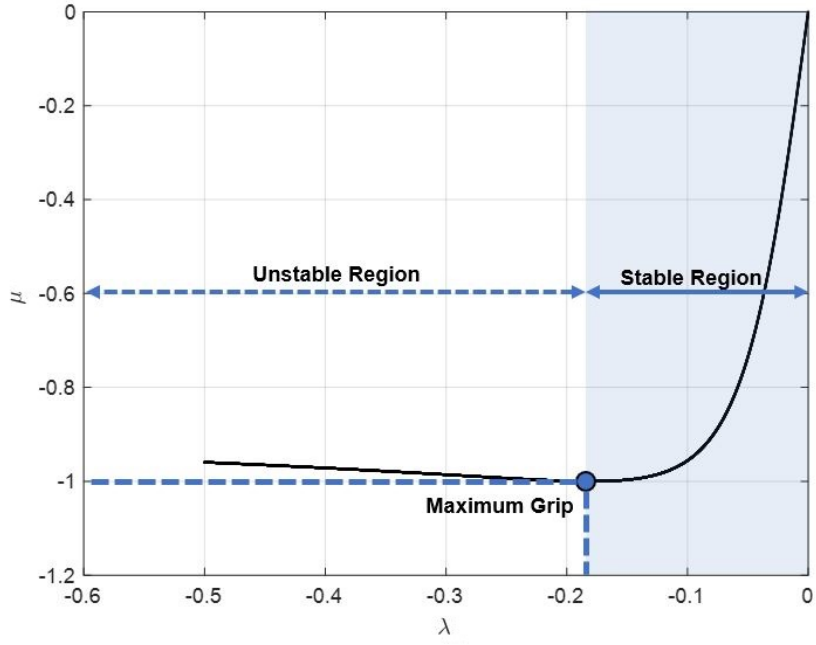


Figure 4.3. Exemplary typical tire characteristic curve and maximum potential friction.

the stable and unstable regions are shown, which are separated by the optimal slip λ^* and relative potential friction μ_{max}. The characteristic shows that, starting from $\lambda = 0$, increasing the braking torque leads to increasing grip (in absolute value) until the optimal value. From that point on, an increasing braking torque does not lead to any additional grip, nonetheless

the slip increases, which leads to wheel locking and eventually complete sliding of the vehicle.
The ratio between the longitudinal force and normal load gives the current friction, i.e. the instantaneous real time friction between tire and road asphalt, namely:

$$\hat{\mu}^i = \frac{F_l^i}{F_z^i}. \tag{4.12}$$

The friction in Eq. (4.12) is related to both changes of the road state and asphalt condition, nonetheless, as previously stated, it is assumed that only the longitudinal slip leads to friction changes, therefor a slip value can be associated to each value of the current friction. On the other hand, changes in the friction are associated to the road asphalt only, because the wheel model is pre-calibrated through experiments.
Starting from the current friction, pre-calibrated models and model as-

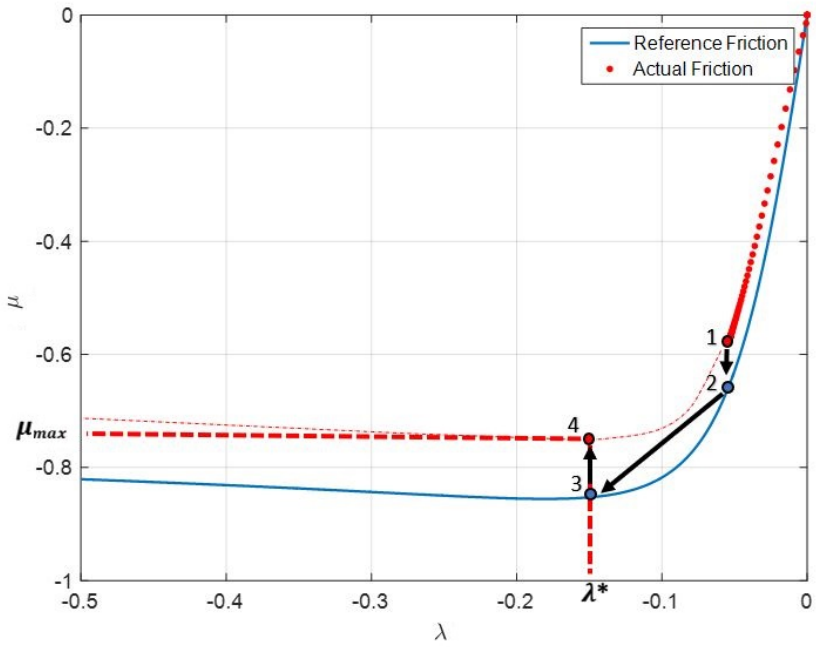


Figure 4.4. Procedure to evaluate potential friction coefficient.

sumptions, the potential friction can be estimated using the following steps:

1. once the current friction has been evaluated as in Eq. (4.12) (point 1 in Fig. 4.4) the equivalent grip for the pre-calibrated road model (point 2 in Fig. 4.4) is evaluated as:

$$\hat{\mu}^i_{ref} = \frac{F^i_{l_{ref}}}{F^i_z}, \tag{4.13}$$

where $F^i_{l_{ref}}$ is the pre-calibrated model longitudinal force associated to the the current slip.

2. using the same pre-calibrated model, a maximum available longitudinal force $F^i_{l_{ref,max}}$ is retrieved from the tire characteristic, which is associated to the optimal slip ratio λ^* (point 3 Fig. 4.4)

$$\hat{\mu}^i_{ref,max} = \frac{F^i_{l_{ref,max}}}{F^i_z}, \tag{4.14}$$

3. the final potential friction (point 4 Fig. 4.4) is evaluated using the proportionality law already employed between in step 1, assuming linearity in the tire behaviour under the vehicle working conditions.

$$\hat{\mu}^i_{max} = \frac{\frac{F^i_l}{F_{z_i}}}{\frac{F^i_{l_{ref}}}{F_{z_i}}} \hat{\mu}^i_{ref,max}. \tag{4.15}$$

The final estimated potential friction $\hat{\mu}_{max}$ is obtained by averaging the front and rear estimations $\hat{\mu}^f_{max}$ and $\hat{\mu}^r_{max}$. This information is returned to the longitudinal ADAS modules to be presented in the following sections.

4.4 Road-Grip Aware Advanced Driving Assistance Systems

4.4.1 Predictive Adaptive Cruise Control

The Adaptive Cruise Control (ACC) is responsible of performing vehicle following maneuvers by continuously tracking the speed of the preceding vehicle, often named leading vehicle in the technical literature. The ACC is usually made of two independent layers, namely upper and lower layer. The upper layer is a reference governor selecting the appropriate acceleration in order to solve the problem stated in Eq. (4.1), rewritten here for completeness.

$$\begin{aligned} &d \to d_{des}(\hat{\mu}_{max}), \\ &\Delta v = v_{lead} - v \to 0. \end{aligned} \tag{4.16}$$

The role of the lower layer is to map the acceleration command to longitudinal actuators commands, i.e. throttle and brakes, thus tracking the reference signal. This last controller is strongly linked to the vehicle powertrain architecture so it is out of the scope of this work, therefore the upper layer is the focus of this chapter.
In the state-of-the-art technical practice, the ACC is designed by using PID controllers, usually empirically tuned based on the expertise of practitioners. A more advanced and recent approach is to design optimal controllers by embedding predictions of the leading and ego-vehicle dynamics, e.g. by using LQR. Despite their effectiveness, both the approaches cannot take into account state and input constraints, potentially resulting in uncontrolled constraints violations.
For the aforementioned facts, this thesis proposes a Predictive ACC designed following the MPC approach, allowing for the continuous constrained optimization of the vehicle following maneuvers. At each sample time the MPC solves a finite horizon constrained optimization problem by selecting the *best* control input sequence, nevertheless, according to the receding horizon principle, only the first element of this sequence is applied to the system under control, and, at the following sampling step, the process is repeated again.
Let us first design the continuous time state-space control oriented model,

according to the vehicle-following paradigm [6]:

$$\begin{aligned} \dot{d} &= \Delta v, \\ \dot{\Delta v} &= a_{lead} - a, \\ \dot{v} &= a, \\ \dot{a} &= \frac{1}{\tau}(-a + u), \end{aligned} \tag{4.17}$$

where a_{lead} is the acceleration of the leading vehicle, a is the acceleration of the ego vehicle, τ is the power-train time constant and finally u is control command.
Equation (4.17) must be discretised, e.g. using forward Euler method, in order to implement the controller, thus resulting in:

$$\begin{aligned} d_{k+1} &= d_k + T_s \Delta v_k, \\ \Delta v_{k+1} &= \Delta v_k + T_s(a_{lead_k} - a_k), \\ v_{k+1} &= v_k + T_s a_k, \\ a_{k+1} &= a_k + \frac{T_s}{\tau}(-a_k + u_k), \end{aligned} \tag{4.18}$$

where T_s is the discretisation time step and k is the discrete time index. In order to rewrite Eq. (4.18) in a more compact matrix notation, let us define the state vector $x_k = [d_k \ \Delta v_k \ v_k \ a_k]^T \in \mathbb{R}^4$, the output vector as $y_k = [d_k - d_{des} \ \Delta v_k \ a_k]^T \in \mathbb{R}^3$ and $w_k \in \mathbb{R}$ as the leader acceleration, i.e. $w_k = a_{lead_k}$. Therefore, rewrite Eq. (4.18) as:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Gw_k, \\ y_k &= Cx_k + Z, \end{aligned} \tag{4.19}$$

where

$$A = \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & -T_s \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 - \frac{T_s}{\tau} \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{T_s}{\tau} \end{bmatrix}, G = \begin{bmatrix} 0 \\ T_s \\ 0 \\ 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 & -\tau(\hat{\mu}) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, Z = \begin{bmatrix} -d_0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{4.20}$$

Note that the leading vehicle acceleration is treated as a constant disturbance because it depends on the future chosen command of the respective vehicle user, which is unknown and non-observable in general.
Moreover, by augmenting the state vector as $[x_k\ u_k]$, and the output vector as $[y_k\ u_k]$, it is possible to resort to the delta-input offset free formulation [56], where the input command is moved into the state vector and the new input signal is:

$$u_k = u_{k-1} + \delta u_k. \tag{4.21}$$

This formulation is particularly convenient where biases or drift exist in the actuators, e.g. the gain or model is uncertain. Therefore the matrices in Eq. (4.19) can be rewritten as:

$$A = \begin{bmatrix} A & B \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} B \\ 1 \end{bmatrix}, G = \begin{bmatrix} G \\ 0 \end{bmatrix}, C = \begin{bmatrix} C & 0 \\ 0 & 1 \end{bmatrix}, Z = \begin{bmatrix} Z \\ 0 \end{bmatrix}, \tag{4.22}$$

where the matrix symbols have been reused for simplified notation.
This formulation is additionally beneficial for the ACC problem because the delta-input is the discrete time difference of an acceleration command, thus making it the jerk command. Therefore, by integrating delta-input constraints, the comfort can be greatly improved.
The control goal stated in Eq. (4.16) is achieved by regulating the output vector to the origin. The MPC allows to balance multiple objectives by appropriately designing a cost function to be minimized, namely, at the k-th time instant:

$$J_k(y_k, \delta u_{k-1}) = \overbrace{y_k^T Q y_k}^{\text{tracking}} + \overbrace{\delta u_{k-1}^T R \delta u_{k-1}}^{\text{comfort}}, \tag{4.23}$$

where y_k is the prediction at k, which can be obtained by integrating Eq. (4.18), given the $(k-1)$-th value and the chosen control command. Additionally, in Eq. (4.23), $Q = \text{diag}(q_1, q_2, q_3, q_4) \in \mathbb{R}^4$ is a positive definite diagonal matrix and $R = r \in \mathbb{R}^+$ is a positive scalar. By appropriately selecting the two aforementioned matrices, a trade-off between tracking performance and control effort, i.e. acceleration and jerk, can be achieved. The cumulative cost function is obtained by summing up Eq. (4.23) across consecutive H_p prediction steps, thus obtaining :

$$J(y_{1,\dots,H_p}, \delta u_{0,\dots,H_c-1}) = \sum_{i=0}^{H_p-1} \left[y_{i+1}^T Q y_{i+1} + \delta u_i^T R \delta u_i \right], \tag{4.24}$$

where H_p and H_c are the prediction and control horizons, respectively. The chosen value for H_c should be obviously bounded by H_p, i.e. $H_c \leq H_p$, and the bigger it is, the higher the computational cost. In the following the horizons will be set as $H_c = H_p$ because, by using efficient solvers, modern embedded systems are perfectly capable of solving linear MPCs even under strict real-time requirements.
With respect to safety, road-grip constraints are introduced for the desired and actual acceleration, as:

$$\begin{aligned} a_{min}(\hat{\mu}_{max}) \leq a_k \leq a_{max}(\hat{\mu}_{max}), \ \forall k, \\ u_{min}(\hat{\mu}_{max}) \leq u_k \leq u_{max}(\hat{\mu}_{max}), \ \forall k, \end{aligned} \tag{4.25}$$

with $a_{max}(\hat{\mu}_{max}) = u_{max}(\hat{\mu}_{max}) = \min(2, \hat{\mu}_{max} g)$ and $a_{min}(\hat{\mu}_{max}) = u_{min}(\hat{\mu}_{max}) = \max(-4, -\hat{\mu}_{max} g)$, being $\hat{\mu}_{max}$ the estimated maximum available grip and g the acceleration of gravity. The saturation values (i.e, $2m/s^2$ and $-4m/s^2$) are chosen as upper and lower limit, related to the ideal value for the grip, set as $\mu_{max} = 1$.
The constraints on the spacing and the maximum velocity are given as:

$$\begin{aligned} d_{min} \leq d_k, \ \forall k, \\ v_k \leq v_{max}, \ \forall k, \end{aligned} \tag{4.26}$$

where d_{min} is set to the standstill value d_0 (see Eq. (4.2)) and v_{max} is the maximum admissible speed depending on the legal requirements on the specific traveled road (urban, extra-urban, etc.). Note that this

information can be easily acquired from a combination of map and GNSS or from cameras. Finally, additional constraints on the control input are defined for further improving the driving comfort as:

$$\delta u_{min} \leq \delta u_k \leq \delta u_{max}, \ \forall k. \tag{4.27}$$

Finally by combining the aforementioned cost, equality and inequality constraints, the complete constrained optimization problem can be written as:

$$\begin{aligned} &\min_{y_{1,\dots,H_p}, \delta u_{0,\dots,H_p-1}} && J(y_{1,\dots,H_p}, \delta u_{0,\dots,H_p-1}) \\ &\forall k \text{ subject to} && x_{k+1} = Ax_k + B\delta u_k + Gw_k \\ & && y_k = Cx_k + Z \\ & && d_{min} \leq d_k \\ & && v_k \leq v_{max} \\ & && a_{min} \leq a_k \leq a_{max} \\ & && u_{min} \leq u_k \leq u_{max} \\ & && \delta u_{min} \leq \delta u_k \leq \delta u_{max} \end{aligned} \tag{4.28}$$

The aforementioned MPC problem can be recast as a quadratic programming problem [57]. In the following, the needed steps to obtain such formulation are shown.
Let us first define the auxiliary vector obtained by stacking future state and output predictions, and delta-input control commands, namely:

$$X \triangleq \begin{bmatrix} x_{k+1} \\ \vdots \\ x_{k+H_p} \end{bmatrix}, Y \triangleq \begin{bmatrix} y_{k+1} \\ \vdots \\ y_{k+H_p} \end{bmatrix}, \Delta U \triangleq \begin{bmatrix} \delta u_k \\ \vdots \\ \delta u_{k+H_p-1} \end{bmatrix}, W \triangleq \begin{bmatrix} w_k \\ \vdots \\ w_{k+H_p-1} \end{bmatrix}. \tag{4.29}$$

Starting from the previous vectors, the complete state-output prediction along the horizon H_p can be written in matrix form as:

$$\begin{aligned} X &= \bar{A}x_k + \bar{B}\Delta U + \bar{G}W, \\ Y &= \bar{F}x_k + \bar{D}\Delta U + \bar{E}W + \bar{Z}, \end{aligned} \tag{4.30}$$

where the matrices

$$\bar{A} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{H_p} \end{bmatrix}_{(n_x H_p \times n_x)} \qquad \bar{C} = \begin{bmatrix} C & 0 & \dots & 0 \\ 0 & C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C \end{bmatrix}_{(n_y H_p \times n_x H_p)}$$
$$\bar{B} = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{H_p-1}B & A^{H_p-2}B & \dots & B \end{bmatrix}_{(n_x H_p \times H_p)} \tag{4.31}$$

$$\bar{G} = \begin{bmatrix} G & 0 & \dots & 0 \\ AG & G & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{H_p-1}G & A^{H_p-2}G & \dots & G \end{bmatrix}_{(n_x H_p \times H_p)} \qquad \bar{Z} = \begin{bmatrix} Z \\ Z \\ \vdots \\ Z \end{bmatrix}_{(n_y H_p \times 1)} \tag{4.32}$$
$$\bar{F} = \bar{C}\bar{A}, \qquad \bar{D} = \bar{C}\bar{B}, \qquad \bar{E} = \bar{C}\bar{G},$$

where $n_x = 5$ and $n_y = 4$ are the sizes of state x_k and output y_k vector, respectively. Additionally the cost in Eq. (4.24) can be recast in matrix form as:

$$J(Y, \Delta U) = Y^T \bar{Q} Y + \Delta U^T \bar{R} \Delta U, \tag{4.33}$$

where

$$\bar{Q} = \begin{bmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q \end{bmatrix}_{(n_y H_p \times n_y H_p)} \qquad \bar{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R \end{bmatrix}_{(H_p \times H_p)} \tag{4.34}$$

By substitution of the output prediction from Eq. (4.29) in Eq. (4.33) and usage of standard linear algebra manipulations:

$$\begin{aligned} J(x_k, \Delta U) &= (\bar{F}x_k + \bar{D}\Delta U + \bar{E}W + \bar{Z})^T \bar{Q}(\bar{F}x_k + \bar{D}\Delta U + \bar{E}W + \bar{Z}) + \\ &+ \Delta U^T \bar{R} \Delta U = \\ &= \Delta U^T (\bar{R} + \bar{D}^T \bar{Q} \bar{D}) \Delta U + 2(\bar{F}x_k + \bar{E}W + \bar{Z})^T \bar{Q} \bar{D} \Delta U \\ &= \Delta U^T H \Delta U + F \Delta U, \end{aligned} \tag{4.35}$$

where $H \triangleq \bar{R} + \bar{D}^T \bar{Q} \bar{D}$ and $F \triangleq 2(\bar{F}x_k + \bar{E}W + \bar{Z})^T$. In Eq. (4.33) the delta-input independent term have been removed because they do not contribute to the final cost.
Similarly, the constraints in Eq. (4.25), (4.26) and (4.27) can be recast in matrix form, $\forall k$, as:

$$\begin{aligned} L_M x_k &\leq x_{max}, \\ L_m x_k &\leq x_{min}, \\ \delta u_k &\leq \delta u_{max}, \\ -\delta u_k &\leq -\delta u_{min}, \end{aligned} \tag{4.36}$$

where $x_{max} = [v_{max}\ a_{max}\ u_{max}]$, $x_{min} = [-d_{min}\ -a_{min}\ -u_{min}]$ and:

$$L_{max} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, L_{min} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{4.37}$$

Again, by stacking the complete state X and control ΔU, as defined in Eq. (4.29) the complete set of box constraints is:

$$\begin{aligned} \bar{L}X &\leq \bar{M}, \\ \Delta U &\leq \bar{N}, \end{aligned} \tag{4.38}$$

where

$$\bar{L} = \begin{bmatrix} L \\ \vdots \\ L \end{bmatrix}, \; L = [L_{max} \; L_{min}]^T,$$
$$\bar{M} = \begin{bmatrix} M \\ \vdots \\ M \end{bmatrix}, \; M = [x_{max} \; x_{min}]^T, \tag{4.39}$$
$$\bar{N} = \begin{bmatrix} N \\ \vdots \\ N \end{bmatrix}, \; N = [\delta u_{max} \; -\delta u_{min}]^T.$$

Finally by substitution of the state prediction from Eq. (4.30) in Eq. (4.38) the following is obtained:

$$\begin{aligned} \bar{L}\bar{B}\Delta U &\leq \bar{M} - \bar{L}\bar{A}x_k - \bar{L}\bar{G}W, \\ \Delta U &\leq \bar{N}, \end{aligned} \tag{4.40}$$

in which every constraints is now only relative to the optimal control decision variables, i.e. ΔU.
In conclusion the dense quadratic programming problem is:

$$\begin{aligned} \min_{\Delta U} \quad & \Delta U^T H \Delta U + F \Delta U \\ \text{subject to} \quad & \bar{L}\bar{B}\Delta U \leq \bar{M} - \bar{L}\bar{A}x_0 - \bar{L}\bar{G}W \\ & \Delta U \leq \bar{N} \end{aligned} \tag{4.41}$$

which is particularly convenient because many efficient embedded solvers are available today, both commercial (e.g. ForcesPRO) and open source (e.g. ACADO), to solve such formulation in real-time. The aforementioned solvers are able to solve quadratic programming problems in milliseconds, depending on the matrices size and CPU performances.

About stability and feasibility

In practice additional effort is necessary to ensure feasibility of the optimization problem.
The optimization problem may be momentary unfeasible because of vio-

lation of constraints, which will in turn make the solver fail. This might happen especially when the leading vehicle has high deceleration, or when the estimated road friction coefficient $\hat{\mu}_{max}$ changes abruptly, and therefore the reference distance changes as well. For the aforementioned reasons state constraints are formulate as soft-constraints [58] and a slack variable is added to the cost function. When constraints are not active, the slack variable is null and the original problem is obtained.
The stability of Linear MPC is a complicated function of its hyper-parameters. The most common approach to ensure convergence, is to set a terminal cost, i.e. the last term of the diagonal matrix $\bar{Q}$, so that it is a solution of the Riccati equation for the corresponding unconstrained LQR problem [57]. This empirically means that in the neighborhood of the origin, i.e for a sufficiently small α, constraints are not active and the MPC controller is identical to the LQR unconstrained counterpart.

4.4.2 Autonomous Emergency Brake

Road accidents and fatalities statistics are reported annually, [59, 60], showing the relation between accidents and drivers behaviour. Moreover, [61] showed that the collision risk increases with the degradation of road conditions. The AEB is one of the most effective driving functionalities for collision prevention and social cost lowering linked to accidents. Nonetheless, EuroNCAP tests are being carried on roads with friction peaks of at least 0.9, even if in real situations a lower value reduces the safety and the robustness of the whole system.
In this perspective, the aim of the grip-aware AEB system proposed by the authors is to identify the collision risk depending on the actual road conditions and, hence, to take control of the brakes to avoid possible accidents or at least to reduce their severity. Here, the decision-making of AEB is make according to the TTC in Eq. (4.4), where the detection threshold depends on the estimated road-grip as:

$$T_{th}(\hat{\mu}_{max}) = \frac{v}{\hat{\mu}_{max} a_{brk}}, \tag{4.42}$$

being a_{brk} the deceleration value commanded to the ABS in case of emergency, i.e. $9.8 m/s^2$.

4.4.3 Anti-Lock Braking System

Once an emergency braking is commanded from the AEB, the ABS has to drive the brake system, preventing wheels from locking during the hard braking maneuver. Here a Sliding Mode Controller (SMC) ABS controller is proposed, which leverages the on-line estimation of the road-grip in order to provide a safe braking automatic maneuver for a vehicle-following process, also in the presence of hard rainy or icy pavement. This choice is due to SMC's enhanced stability performances with respect to classical control architectures [62] (e.g. proportional action). In particular, it can be shown that matched disturbances (uncertainties entering the system through the same channel as the control) are rejected, at least below the actuation limits, moreover due to the controller nonlinear nature, larger stability margins can be achieved.
First, let us define a control-oriented model in which the weight transfer, lateral motion and yaw motion are neglected, thus obtaining a model dealing with the wheel rotational dynamics and longitudinal vehicle dynamics. The model is the same for each wheel, i.e. the apex i, referring to a specific wheel, will be ignored. The rotational dynamics of the wheel is described by

$$I_w\dot{\omega} = -T_b - R_r F_l, \tag{4.43}$$

where I_w is the moment of inertia about the wheel axis of rotation, ω is the angular velocity, T_b is the braking torque, R_w is the wheel rolling radius and F_l is the force produced by the friction reaction. The longitudinal vehicle dynamics are simply modeled as

$$m\dot{v} = -F_l, \tag{4.44}$$

where m is the vehicle mass.
The control goal is to yield λ (see Eq. (4.11)) to a reference value λ^* during braking [63]. To this aim the following sliding surface is defined as:

$$\sigma = \lambda - \lambda^\star, \tag{4.45}$$

where $\lambda^\star$ is the optimal slip obtained from the friction estimator and λ is the longitudinal slip. By taking the derivative of Eq. (4.11), the following

slip dynamics are obtained:

$$\dot{\lambda} = -\frac{1}{v}\left(\frac{1-\lambda}{m} + \frac{R_w^2}{I_w}\right) F_l + \frac{R_w}{vI_w} T_b. \tag{4.46}$$

Due to the inertia differences between wheel and vehicle, the velocity v is taken as slowly varying parameter, thus reducing Eq. (4.46) to a single-input single-output system, where the control law can be defined as:

$$u = T_b = u_c + u_{sw}, \tag{4.47}$$

being u_c the continuous term, or *equivalent control* [62], and u_{sw} the discontinuous term. The equivalent control input is is responsible for keeping the trajectories on σ, i.e.

$$\dot{\sigma} = 0 \Rightarrow u_c = \left(\frac{(1-\lambda)I_w}{mR_w} + R_w\right) F_n \hat{\mu}, \tag{4.48}$$

where the force $F_l = F_n\hat{\mu}$, being F_n the tire vertical load, and $\hat{\mu}$ the current friction value provided by the estimation module. Closed-loop stability can be easily proven by considering the following Lyapunov function $V(\lambda) = \frac{1}{2}\sigma^2$ and its derivative $\dot{V}(\lambda) = \sigma\dot{\sigma}$.
By substitution of Eqs. (4.47)-(4.48) into the expression of $\dot{V}$, the following is obtained:

$$\dot{V}(\lambda) = \sigma\dot{\sigma} = \sigma\left(\frac{R_w}{vI_w} u_{sw}\right). \tag{4.49}$$

Hence, selecting $u_{sw} = -\frac{vI_w}{R_w}\eta sgn(\sigma)$ it follows that

$$\dot{V}(\lambda) = -\eta\sigma sgn(\sigma) = -\eta|\sigma| < 0, \tag{4.50}$$

being $\eta > 0$. In so doing, the surface σ is attractive and the closed-loop is asymptotically stable.
Note that, in order to avoid the well-known chattering problem of the sliding mode controllers, for its practical implementation the sign function in Eq. (4.50) has been substituted by the hyperbolic tangent function. Furthermore, since controllability is lost when the vehicle speed is approaching zero (see Eq. (4.46)), following a common practice for implementing the ABS, the controller is disabled at the very low velocities, i.e. $v < 1m/s$.

4.5 Validation and Discussion

Numerical validation has been carried out by using a purposely designed co-simulation platform. Model In the Loop (MIL) testing is a powerful tool for validating software prior to embedding it into hardware platform, thus saving development time.
The proposed platform has been developed by leveraging two main components, namely:

- Matlab/Simulink has been used to both simulate ego-vehicle dynamics and design controllers.

- Simulation of Urban MObility (SUMO) [64], an open-source road traffic simulation package, enabling engineers to model road network, traffic lights rules, vehicle routing and behaviours. Each entity is simulated microscopically built upon realistic driving models [65, 66].

Despite the realistic behaviours implemented in SUMO, the vehicles are still simulated following kinematic update rules, thus Simulink has been used to describe a highly detailed ego-vehicle dynamic behaviours in terms of chassis and actuation dynamics (see Sec. 2.2.1). On the other hand, SUMO has been used to design the road network and the driving scenarios, as well as the road asphalt conditions by defining different grip coefficients along the road.
The platform works following the client-server paradigm, where SUMO, acting as a server, waits for command requests from Simulink, i.e. the client. The communication is possible thanks to TRACI4Matlab, a Matlab implementation of the original TRACI, a set of communication API provided by SUMO itself. The simulation and control parameters are summed up in Tab. 4.2 and 4.3, towards the end of this chapter.

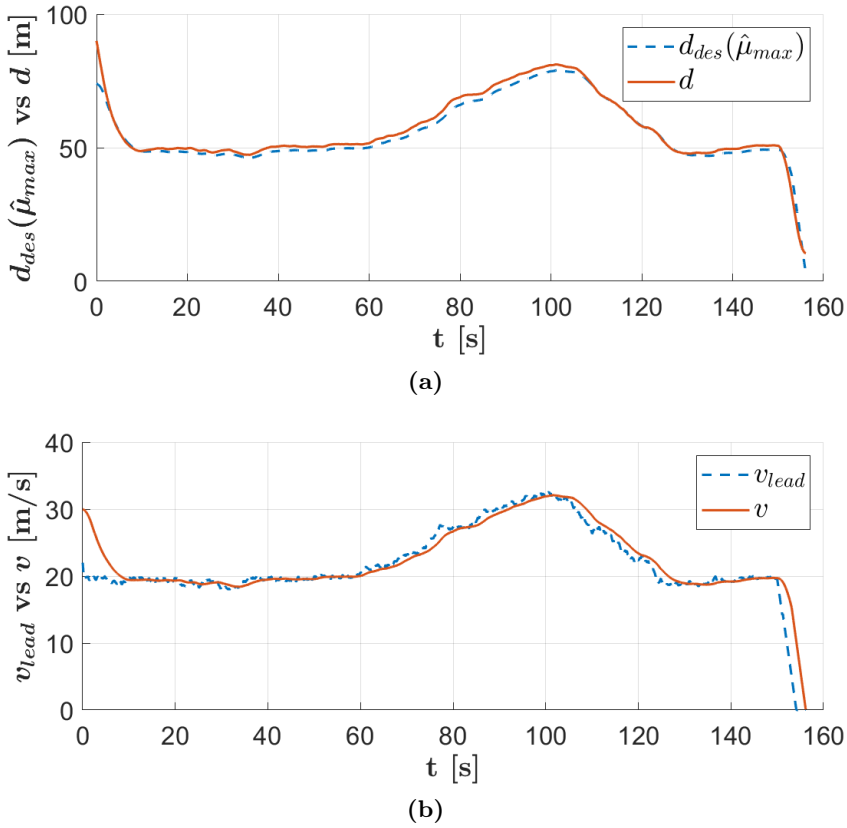


Figure 4.5. Simulation results of Scenario 1: vehicle-following with constant road grip. **(a)** Time-history comparison of vehicle distance, d, and desired distance, $d_{des}(\hat{\mu}_{max})$. **(b)** Time-history comparison of the ego velocity v and leader velocity v_{lead}.

In order to assess the performances of the proposed ADAS solution, numerical results will be shown for the following scenarios:

- scenario 1: vehicle-following scenario on a typical motorway. Ego-vehicle travels with starting speed $v(0) = 30m/s$, while having a starting gap of $d(0) = 90m$ to its predecessor, which travels with a starting speed of $v_{lead}(0) = 20m/s$. The leading vehicle is a human-driven vehicle simulated by SUMO using its internal behavior models. The simulated realistic driving profile takes into account for road

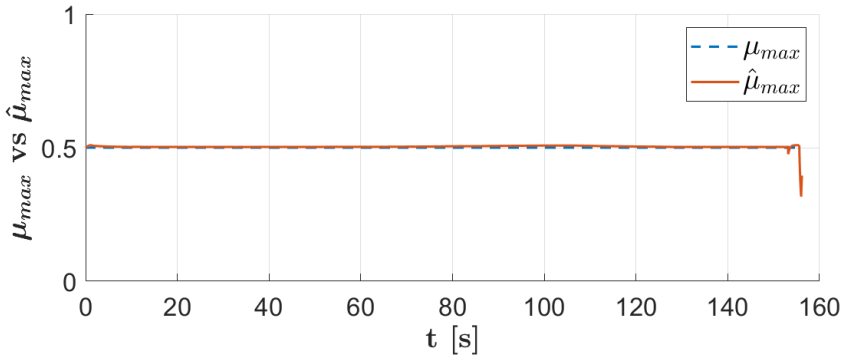


Figure 4.6. Simulation results of Scenario 1: vehicle-following with constant road grip. Time-history comparison of the real potential road-grip μ_{max} and its on-board estimate $\hat{\mu}_{max}$.

limits and human driving imperfections. Due, to the presence of an obstacle, the leading vehicle performs an emergency braking inducing the maximum negative acceleration allowed by the road grip, i.e. $g\mu_{max}m/s^2$. In this scenario the actual road grip in constant $\mu_{max} = 0.5$, which is representative of heavy rain.

- scenario 2: vehicle-following scenario with starting vehicle dynamics conditions analogous to scenario 1, however in presence of varying road grip conditions. Specifically, the road grip starts from ideal dry road conditions, i.e. $\mu_{max}(0) = 1$, and decreases in two steps, namely $\mu_{max} = 0.75$ and $\mu_{max} = 0.5$.

- scenario 3: Stop&Go scenario at urban speed and gaps, with constant road grip coefficient $\mu_{max} = 0.5$.

Scenario 1: Results are depicted in Figs. 4.5a-4.5b and 4.6. The ego vehicle is able to track the leading vehicle speed, as shown in Fig. 4.5b, meanwhile preserving the desired road-grip dependent desired distance $d_{des}(\hat{\mu}_{max})$, Fig. 4.5a. Moreover, the road grip estimate $\hat{\mu}_{max}$, starting from the value 0.5, is shown in Fig. 4.6. The aforementioned figures show that the vehicle is capable of safely avoid the collision, when the leading vehicle perform harsh braking, at $t = 150s$, by combining the three proposed ADAS modules.

Specifically, the Predictive ACC design ensures tracking capability while enforcing comfort constraints during vehicle-following, i.e. until $t = 150s$. Indeed, at this time instant, the ACC tries to handle the hazardous scenario, but the high necessity of high accelerations lead to the activation of the AEB, which command the emergency braking. This command ignores the ACC constraints, which have lower priority w.r.t. safety.

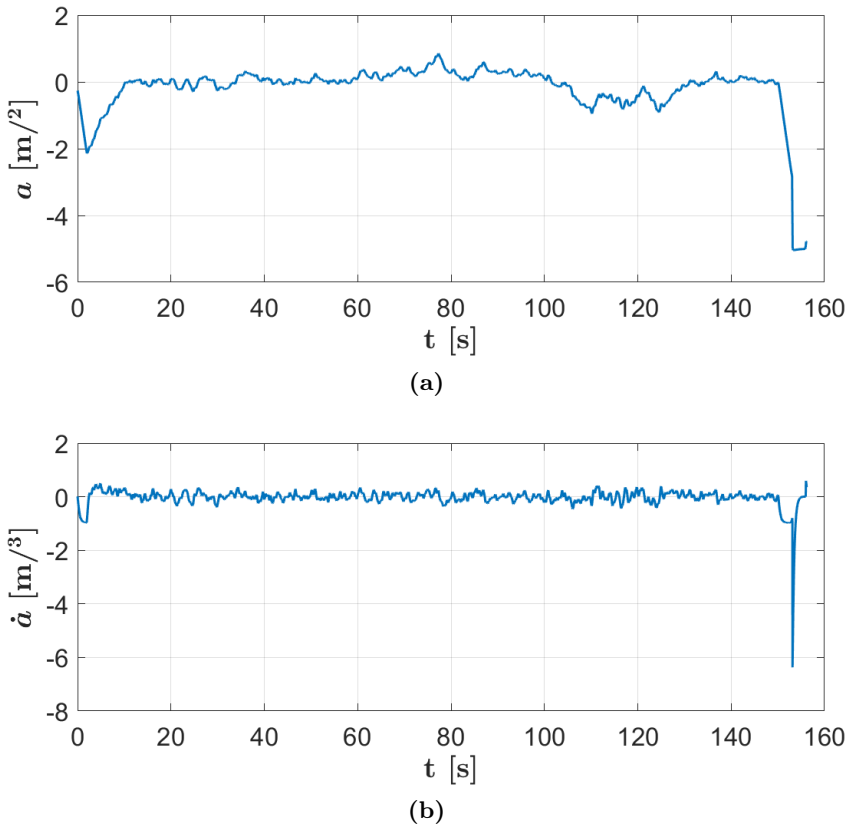


Figure 4.7. Simulation results of Scenario 1: vehicle-following with constant road grip. **(a)** Time-history of the ego-vehicle acceleration a. **(b)** Time-history of the ego-vehicle jerk j.

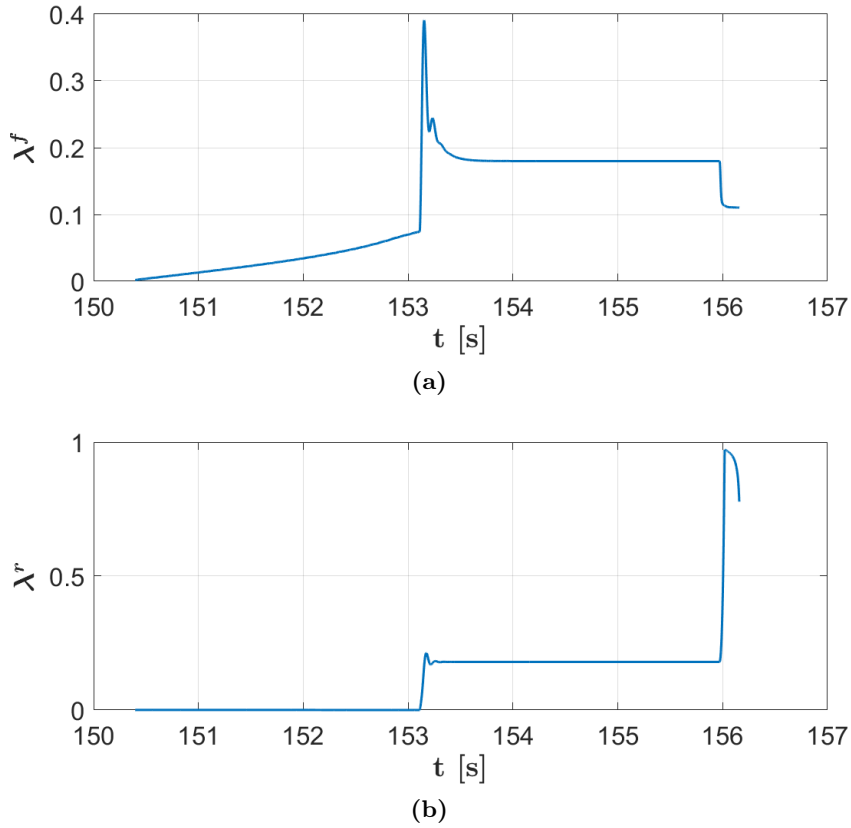
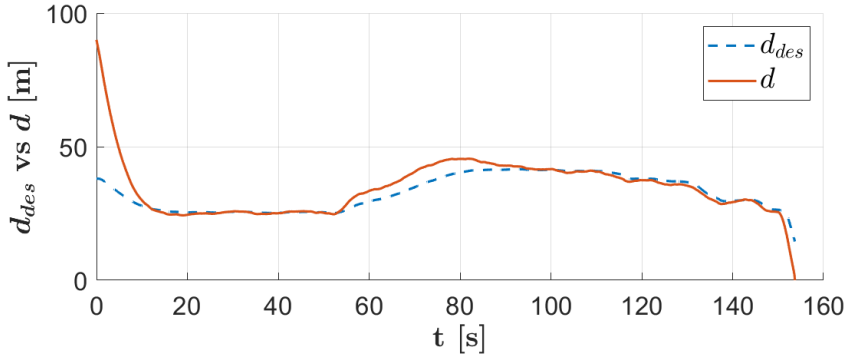


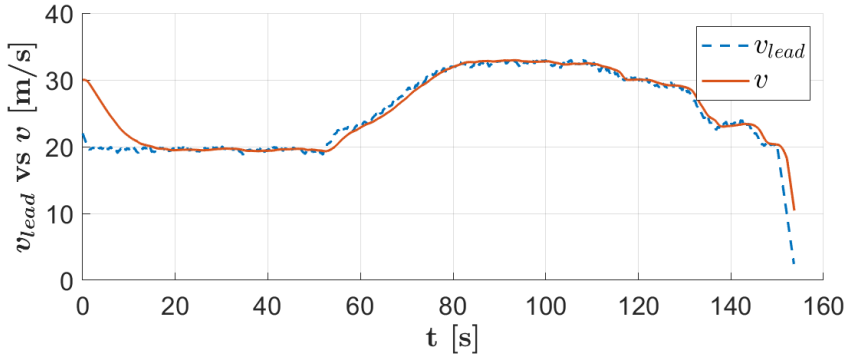
Figure 4.8. Simulation results of Scenario 1: vehicle-following with constant road grip. **(a)** Time-history of the ego-vehicle front tire longitudinal slip ratio λ^f. **(b)** Time-history of the ego-vehicle front tire longitudinal slip ratio λ^r.

In so doing the collision is safely avoided, at the cost of lower comfort during the emergency braking, i.e. higher absolute values of acceleration and jerk (see Figs. 4.7a and 4.7b). Moreover, when the emergency braking is commanded, the ABS is responsible for regulating longitudinal wheel slip to the estimated optimal value λ^*. Results in Figs. 4.8a and 4.8b shows the performances of the ABS during the emergency maneuver, i.e. $t \geq 150s$. Both the longitudinal wheel slips are regulated to the optimal value during the braking maneuver, nonetheless, the last portion of the plots shows divergence, because the ABS is disabled when the vehicle speed

is low, namely $v \leq 1m/s$ in this work.
The scenario simulation is repeated by disabling the road grip adaptation, i.e. by assuming $\hat{\mu}_{max} = 1$, and the results are shown in Fig. 4.9a and 4.9b. In particular the figures show that, the ego vehicle is not capable of avoiding the collision during the emergency brake maneuver because the AEB is activate too late. Note that the distance d in Fig. 4.9a goes to zero, meanwhile the velocity v is positive in Fig. 4.9b.



(a)



(b)

Figure 4.9. Simulation results of Scenario 1 without grip adaptation: vehicle-following with constant road grip. **(a)** Time-history comparison of vehicle distance, d, and desired distance, d_{des}. **(b)** Time-history comparison of the ego velocity v and leader velocity v_{lead}.

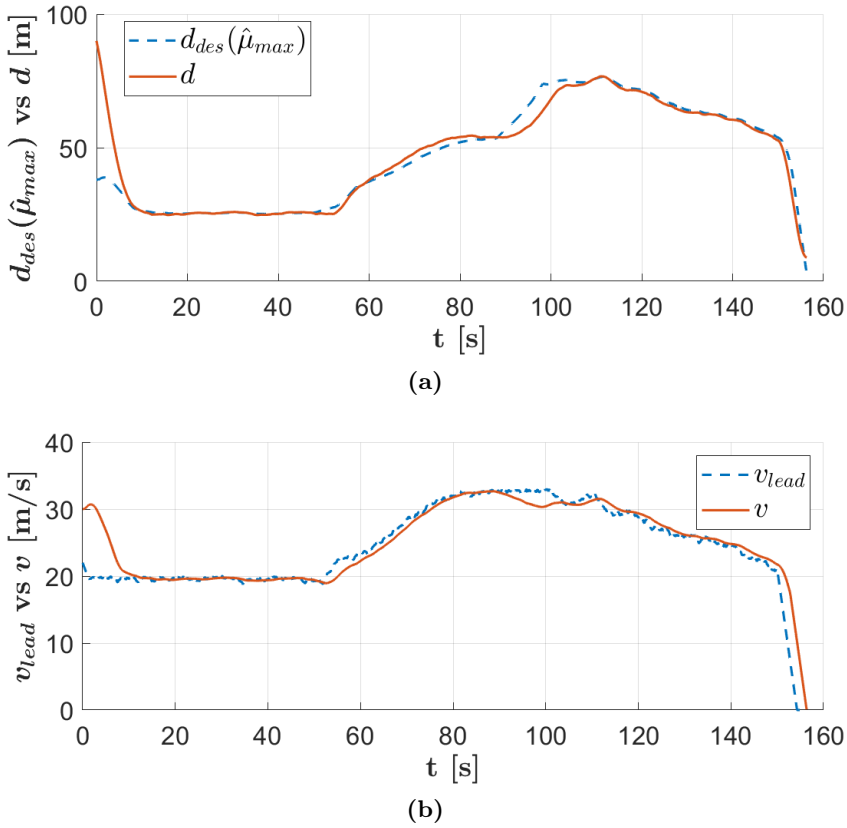


Figure 4.10. Simulation results of Scenario 2: vehicle-following with variable road grip. **(a)** Time-history comparison of vehicle distance, d, and desired distance, $d_{des}(\hat{\mu}_{max})$. **(b)** Time-history comparison of the ego velocity v and leader velocity v_{lead}.

Scenario 2: Results are depicted in Figs. 4.10a-4.11. Similarly to Scenario 1, the ego vehicle is capable of tracking the leading vehicle speed (see Fig. 4.10b), despite the step-like variation of the tire-asphalt grip. The on-line road grip estimation results are shown in Fig. 4.11. Results disclose that the estimation is performed with good enough accuracy, in fact the relative error is always below 1% at steady state. Additionally, as the road grip decreases the desired distance increase accordingly, thus the ACC tracks the new reference without constraint violations. Similarly to

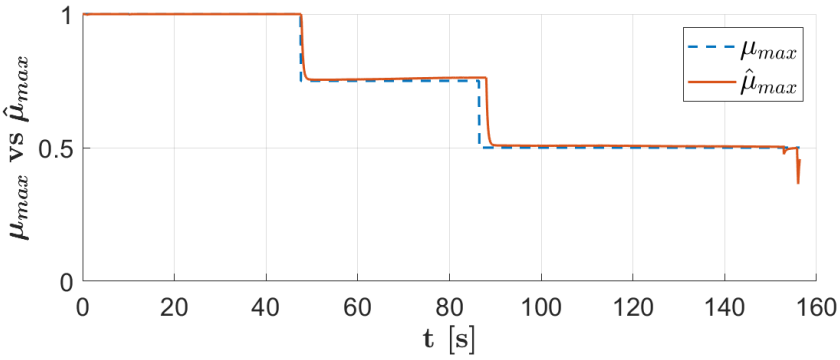


Figure 4.11. Simulation results of Scenario 2: vehicle-following with variable road grip. Time-history comparison of the real potential road-grip μ_{max} and its on-board estimate $\hat{\mu}_{max}$.

Scenario 1, at $t = 150s$, the leading vehicle perform hard braking and the combination of grip estimation and grip-aware ADAS allow the ego vehicle to safely avoid the potential collision.

Scenario 3: Stop&Go scenario results are shown in Fig. 4.12a-4.13c, both by activating and disabling the estimation module. In order to numerically assess safety performances, the following non-dimensional collision index γ [67] is defined as:

$$\gamma = \frac{d - d_{br}}{d_w - d_{br}}, \tag{4.51}$$

where d is the actual distance between the vehicles, d_{br} is the breaking critical distance and d_w is the warning critical distance.

The aforementioned index is commonly used to asses dangerous driving situation and classify possible incoming collisions. Specifically, in case of safe situation it take values greater than the unity, i.e. $\gamma \geq 1$, instead in case of dangerous situation it is positive and smaller than the unity, i.e. $0 \leq \gamma \leq 1$.

Figure 4.12c shows that γ never goes below the unity threshold with the estimation module. On the other hand, if the on-board module is disabled, despite no collision occurs, the safety index goes below the threshold meaning that the driving style is dangerous (see Fig. 4.13c for reference).

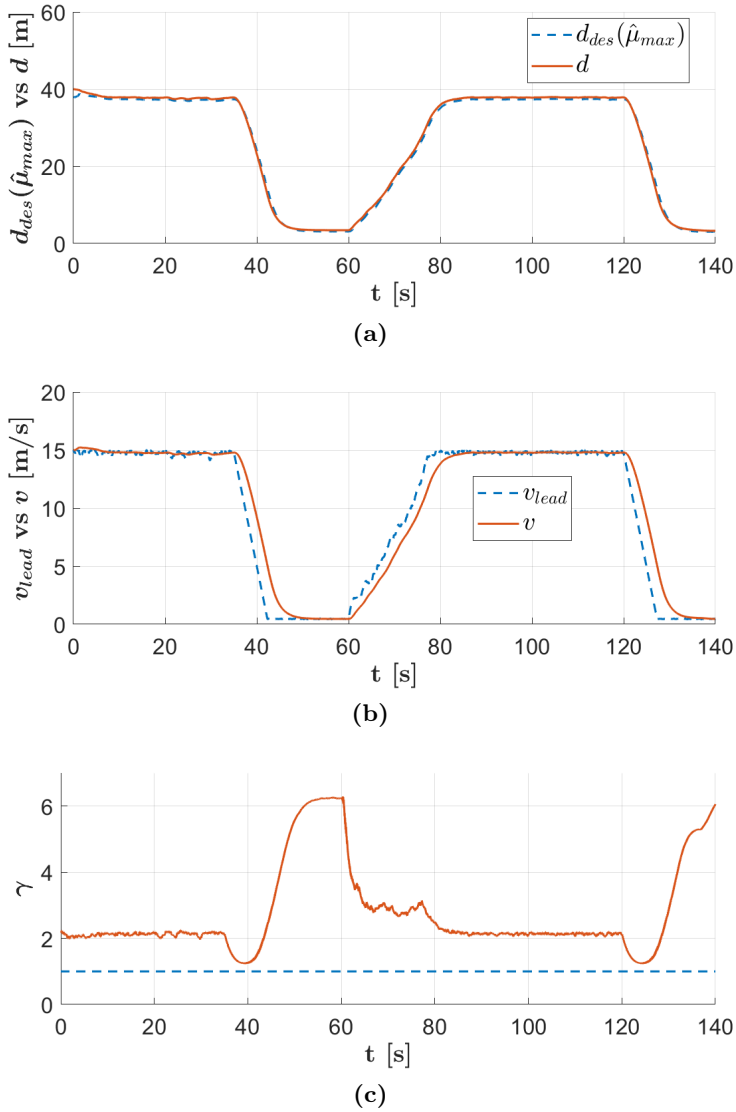


Figure 4.12. Simulation results of Scenario 3 with road grip adaptation. **(a)** Time-history comparison of vehicle distance, d, and desired distance, $d_{des}(\hat{\mu}_{max})$. **(b)** Time-history comparison of the ego velocity v and leader velocity v_{lead}. **(c)** Time-history of the safety index, γ.

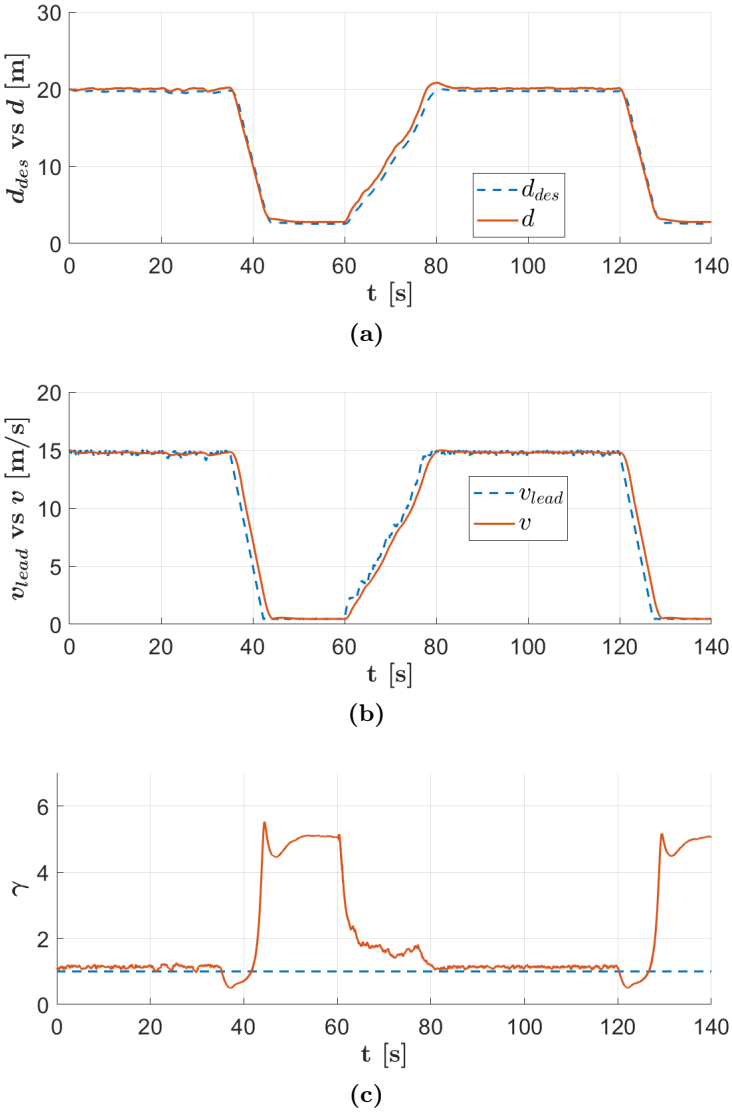


Figure 4.13. Simulation results of Scenario 3 without road grip adaptation. **(a)** Time-history comparison of vehicle distance, d, and desired distance, $d_{des}(\hat{\mu}_{max})$. **(b)** Time-history comparison of the ego velocity v and leader velocity v_{lead}. **(c)** Time-history of the safety index, γ.

Table 4.1. Summary of the numerical validation. The proposed driving scenarios are compared by using three safety indexes.

Scenario	min |T|	min γ	min d
Vehicle following with estimation	2.02	-	10.30
Vehicle following w/o estimation	0	-	0
Stop&Go with estimation	2.73	1.22	3.20
Stop&Go w/o estimation	2.13	0.48	2.70

The effectiveness of the proposed approach is finally summarized in Tab. 4.1. Here, numerical evaluation is carried out with respect to three safety indexes, namely the minimal TTC, minimal collision index and minimal vehicle distance. The table shows that, by leveraging the on-board estimation module, no collision occur and safety can be enhanced in general, being the safety indexes comparable to the one obtained in ideal dry road scenarios [67], i.e. $\mu = 1$.
In conclusion it is shown how, by embedding road-tire grip data into state of the art longitudinal ADAS it is possible to enhance the overall safety, even in dangerous scenarios, such as wet or snowy road asphalt. Formally addressing safety in such scenarios is a needed step to move towards autonomous driving.

Table 4.2. Ego vehicle parameters.

Parameter	Description	Value
m	vehicle mass	1521 [kg]
l_f	distance CoG - front axle	1.2 [m]
l_r	distance CoG - rear axle	1.6 [m]
C_L	lift coefficient	0.28 [-]
A	cross sectional area	2.7 [m^2]
I_w	wheel moment of inertia	1 [kg m^2]
R_w	wheel radius	0.315 [m]
h	CoG height	0.54 [m]

Table 4.3. Control tuning parameters.

Parameter	Description	Value
τ	driveline constant	5 [s]
$\tilde{\tau}_H$	headway time	1.1 [s]
d_0	minimum spacing	2 [m]
T_s	ACC sampling time	0.1 [s]
H_p	ACC prediction horizon	15 [-]
H_c	ACC control horizon	15 [-]
δu_{min}	ACC minimum control	-0.1 [m/s^2]
δu_{max}	ACC maximum control	0.1 [m/s^2]
q_1	ACC spacing tracking weight	2 [-]
q_2	ACC velocity tracking weight	5 [-]
q_3	ACC acceleration weight	20 [-]
q_4	ACC control effort weight	20 [-]
r	ACC incremental control effort weight	20 [-]

Chapter 5

Hierarchical Highway Planning via Deep Reinforcement Learning and Optimal Control

In this chapter a novel solution for autonomous driving planning is presented. The control architecture is built upon a combination of classical control and machine learning techniques, in order to obtain the advantages of both. First the problem statement is given along with the proposed solution, and later its performances are evaluated on highway driving scenarios.

5.1 Motivation and Related Works

Planning for autonomous driving can be divided broadly in two components, namely behavioural planner and motion planner. In particular the behavioral planner maps the sensing state to behaviours, such as desired speed or desired lane, whereas the motion planner maps those into continuous time trajectories. Clearly a strong coupling between the two exists, by taking into account that the top layer can affect the vehicle state through the lower level (see Sec. 2.1 for an introduction on the AD software stack). Despite the recent advancements on this matter, AD planning is still considered an open problem (refer to [68] for a comprehensive state

of the art review), moreover it is the last piece needed in order to close the gap between assisted driving and autonomous driving. Indeed in a SAE Level 2 vehicle, the human driver is the one acting as a behaviour planner, and even motion planner in complex situations, therefore a methodological improvement on planning technologies is needed, in order to substitute the human driver.
Designing a planning module is formally equivalent to designing a driving policy:

$$a_k = \pi(s_k) \quad \vee \quad a_k \sim \pi(a_k|s_k) \tag{5.1}$$

where s_k is the state, a_k is the action, the first policy is deterministic, whereas the second is stochastic. The state s_k is a combination of the ego vehicle state and outside world state, whereas the action a_k could be either a behavior, or a control signal, if the planning module is designed in end-to-end fashion.
Whatever the design choice, the planning module should strive for three fundamental properties:

- **safety**: the autonomous vehicle must not cause any accidents while following the policy in Eq. (5.1),
- **effectiveness**: the autonomous vehicle must be able to reach its goal,
- **scalability**: the autonomous vehicle should be able to drive in any scenario encountered worldwide.

Despite sounding obvious, the property of effectiveness must be taken into account in order to make the problem non trivial. In fact, guaranteeing safety alone is as simple as not letting the AV to move at all. Additionally, the property of scalability [7, 69] is needed for practical and economical reasons, indeed designing and validate a policy for each driving scenario is not a viable solution in the long term.
From an historical point of view, the planning problem has been tackled through classical robotics techniques, e.g. the authors in [70] used Finite State Machines to select driving behaviours, whereas Petri Nets were used in [71] to tackle the same issue. Moreover, Fuzzy Logic was proposed in [72, 73] to issue lane change commands. Regarding motion planning, the

authors in [74] proposed Rapidly-Expanding Random Trees to plan trajectories in urban scenarios, whereas in [75] Artificial Potential Fields were used for collision avoidance. All of the aforementioned techniques, along with many variation of them, are not scalable enough for the AD problem. Specifically they usually require some enumeration of available behaviours or motion primitives, later to be evaluated with respect to a cost function. Manually enumerating each possible solution poses a great burden on development teams, and, by taking into account the unpredictability of everyday driving, it is clear how, pursuing such solutions, leads to unscalable driving policies.
In order to tackle the scalability problem, more recently, some alternatives have been proposed, mainly relying on optimization or game theory. Specifically MPC have been used on various level of the planning stack, such as obstacle avoidance [76, 77] or trajectory planning [78, 79]. On the other hand, game theory is typically used for behaviour planning, such as proposed by the authors in [80, 81]. The aforementioned techniques result in improved scalability, since the optimal solution is evaluated online by optimizing some cost function. Nonetheless, these solutions require some prediction model about the dynamic obstacles in the environment. Predicting the action of other road users is still an open problem in itself, usually tackled through ML (see [82] for a review on this matter), and it is not the scope of this work. Nonetheless, it is argued here that, building driving policies on non verified prediction models, might lead to unsafe behaviors, thus it will be shown how, in the proposed architecture, no prediction model is needed.
The scalability issue naturally suggests the use of ML solutions, which have been proposed in recent years. Specifically, Behaviour Cloning, a form of supervised learning, has been used to clone the behaviour of human drivers [83]. Despite being very effective, this solution requires a great amount of data, and it is not clear if a small sample of locally acquired data can be used to train policies, later to be deployed world-wide. A powerful alternative is model-free DRL, which allows scientists to build policies without any manual data-labeling and prediction model (see Sec. 2.3.2 for an introduction on RL). In model-free DRL, the future outcome for the chosen action a_k, is summarized in a single value, e.g. a value function or a Q-function, without the need for predicting the behaviour of the other agents

in the environment. It is effectively been proposed in many works, e.g. the authors in [84, 85] proposed DQN for selecting behaviours during highway driving, whereas in [86, 87] Dueling DQN has been investigated. Similarly the authors in [88] solved highway driving by using PPO, whereas authors in [89] proposed Soft Actor-Critic. Nonetheless, all of the aforementioned works do not take safety explicitly into account, yet the authors resort to posterior statistical analysis in order to assess safety performances. However, resorting to such designs is not advisable for practical AD, as argued in [90], because as soon as a single line of code is changed, the analysis becomes invalid. Substantially, the naive application of DRL would shift the scalability problem on to the non-scalable posterior safety analysis.
In the following a novel planning architecture is presented. It is obtained by leveraging machine learning and classical control in order to get the advantages of both. The concept of safety is completely decoupled from the learning part so that it can be formally verified since the design phase. It is claimed here that, in comparison to the state of the art, such hierarchical solution achieves safety, effectiveness and scalability while being formally sound, i.e. not requiring any specific behaviour prediction on dynamic obstacles.

5.2 Safe Hierarchical Planning for Autonomous Driving

The crucial part for the AV development is the ability of taking complex decisions in highly varying and unknown scenarios. The aim is to obtain a driving policy which is safe, effective, and scalable. Moreover, the policy should result in human-like behaviour to make sure that the AV can negotiate the right of way within mixed autonomous-human scenarios, in a socially acceptable way. The driving policy can be defined as in Eq. (5.1).
These challenges naturally suggest the use of DRL to solve this problem. On the other hand it is widely known that DRL comes with issues itself, related to:

- safety: the naive application of DRL algorithms of any kind, cannot give guarantees about safety,

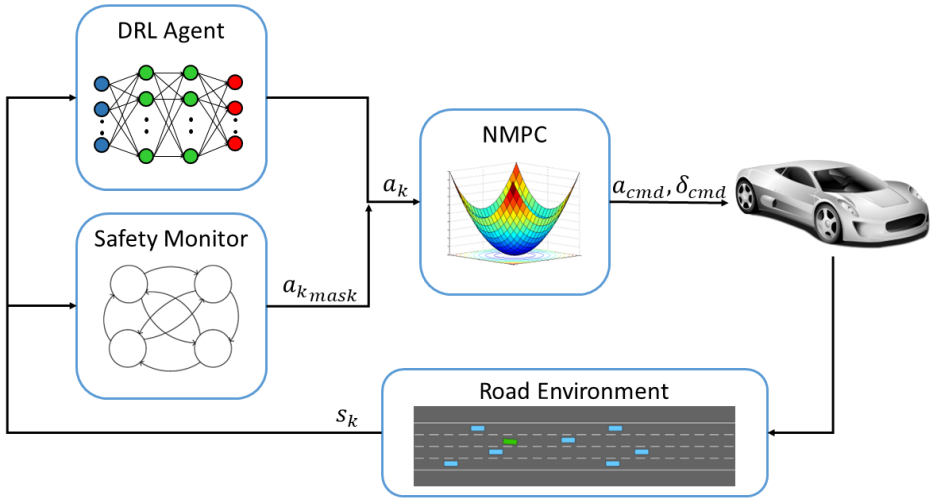


Figure 5.1. Overall hierarchical planning architecture.

- explainability: the choices of a DRL based planner cannot be explained in a human-like form, i.e. in a way that makes sense for social acceptance,
- dimensionality: because of the search spaces involved, a huge amount of data is need at training time, especially if we take into consideration, *edge cases*, i.e. exceptionally rare situations, namely the ones that affect safety the most.

In order to solve the aforementioned limitations the planning problem is tackled here, by first splitting the driving policy as the composition of a discrete and a continuous part, namely:

$$\pi = \pi_D \circ \pi_C. \tag{5.2}$$

The discrete policy π_D is a mapping from the sensing state to behaviours, thus in practice implemented as a behaviour planner, whereas π_C is the mapping from behaviours to continuous commands, i.e. the motion planner. The learning algorithms will be restricted to π_D in order to reduce the search space, thus improving the signal to noise ratio and partially reducing the dimensionality issue. This kind of reasoning has been successfully

applied by authors in [91], nonetheless safety is not explicitly taken into account there, moreover alternative designs are proposed here for both π_D and π_C.
In order to model safety, the discrete policy π_D is further split into the composition of a learnable and a non-learnable part, namely:

$$\pi_D = \pi_L \circ \pi_N. \tag{5.3}$$

In particular the non-learnable part π_N is responsible for safety and acts as a monitor, or constraints, on π_L. The schematics of the overall planning architecture are given in Fig. 5.1.
It is worth to point out here that, attempt on modeling the safety directly into the DRL algorithms have been proposed in the literature, e.g. risk-aware DQN has been proposed in [92, 93] or constrained DQN in [94]. In the aforementioned techniques the safety is still learnt, either in the form of risk, i.e. an additional term to the Q-function, or in the form of constraints, therefore posterior analysis is still needed.
Despite the fact that π_N will act as a constraint on π_L, it is not learnt in Eq. (5.3), thus safety can be validated using formal methods, i.e. without being limited to posterior statistical analysis. Moreover, because of the complete decoupling of the learnable and non-learnable part, any DRL algorithm can be used for π_L.
In the next sections the design of each policy components is shown.

5.2.1 Trajectory Planner via Optimal Control

The trajectory planner module is responsible for defining the policy π_C, i.e. a non-learnable mapping from behaviours to continuous time control commands $u(t) = [a_{cmd}(t), \delta_{cmd}(t)] \in \mathbb{R}^2$. For this purpose a NMPC has been employed.
At each sample time the NMPC receives inputs in the form of behaviours from π_D, and maps them into accelerations and steerings, by evaluating an optimal trajectory.
The prediction model is the non-linear kinematic bicycle model in the

Frenet frame, as defined in Sec. 2.2.3, rewritten here for completeness:

$$\begin{aligned}
\dot{\sigma}(t) &= \frac{v(t)\cos e_\psi(t)}{1-\rho e_y(t)},\\
\dot{e}_y(t) &= v(t)\sin e_\psi(t),\\
\dot{e}_\psi(t) &= v(t)\left(\frac{\tan\delta(t)}{l_f+l_r} - \frac{\rho\cos e_\psi(t)}{1-\rho e_y(t)}\right),\\
\dot{v}(t) &= a(t),\\
\dot{a}(t) &= \frac{a_{cmd}(t)-a(t)}{\tau},\\
\dot{\delta}(t) &= \delta_{cmd}(t).
\end{aligned} \tag{5.4}$$

Despite its simplifying assumptions 2, the kinematic bicycle model has been proven effective for planning [95], at least far from handling limits, i.e. low enough longitudinal and lateral accelerations. In any case a dynamical model, such as the ones in Sec. 2.2.1 or 2.2.2, can be used if the additional computational burden can be handled and the additional needed parameters are known.
The Frenet frame has been chosen, in place of the Cartesian reference, because it allows for standardization of the road structure. Specifically the road is modeled in terms of its curvature $\rho(\sigma)$ and left and right bounds, which naturally become constraints for the NMPC.
In order to implement the controller, Eq. (5.4) must be discretised in

$$\xi_{k+1} = f^{2wk}(\xi_k, u_k) \tag{5.5}$$

The non-linear optimal control problem, to be solved at each sample time T_s, is:

$$\begin{aligned}
&\min_{\xi_{1,\dots,H_p}, u_{0,\dots,H_p-1}} && J(\xi_{1,\dots,H_p}, u_{0,\dots,H_p-1})\\
&\forall k \text{ subject to} && \xi_{k+1} = f^{2wk}(\xi_k, u_k)\\
& && \xi_k \in \mathscr{X}\\
& && u_k \in \mathscr{U}
\end{aligned} \tag{5.6}$$

where ξ_k is the ego vehicle state, $u_k = [a_{cmd_k}, \delta_{cmd_k}]$ is discrete time control, J is the cost function, H_p is the prediction horizon, $\mathscr{X}$ and $\mathscr{U}$ are

the sets of allowed states and inputs. The function J is a quadratic cost defined as:

$$J(\xi_{1,\dots,H_p}, u_{0,\dots,H_p-1}) = \sum_{i=0}^{H_p-1} \left[(\xi_{i+1} - \xi_{ref})^T Q(\xi_{i+1} - \xi_{ref}) + u_i^T R u_i\right] \tag{5.7}$$

where $Q = \text{diag}(0, q_1, q_1, q_1, q_2, q_2)$, $R = \text{diag}(r, r)$ and $\xi_{ref} \in \mathbb{R}^6$ is the reference signal obtained from the behaviour planner policy π_D. Specifically the reference ξ_{ref} is the vector:

$$\xi_{ref} = [0 \; e_{y_{ref}} \; 0 \; v_{ref} \; 0 \; 0] \tag{5.8}$$

where v_{ref} and $e_{y_{ref}}$ are the reference velocity and lateral offset, respectively, to be obtained from π_D. It is clear that the error values e_y and e_ψ will instantly change when a different curve, i.e. a different lane, is chosen as reference.
The constraints in Eq. (5.6) are relative to minimum and maximum lateral and heading angle, in addition to ego vehicle dynamics, i.e. speed, acceleration, steering, and command u_k. Namely

$$\begin{aligned} e_{y_{min}} \leq & e_{y_k} \leq e_{y_{max}} \\ e_{\psi_{min}} \leq & e_{\psi_k} \leq e_{\psi_{max}} \\ & v_k \leq v_{max} \\ a_{min} \leq & a_k \leq a_{max} \\ \delta_{min} \leq & \delta_k \leq \delta_{max} \\ a_{cmd_{min}} \leq & a_{cmd_k} \leq a_{cmd_{max}} \\ \delta_{cmd_{min}} \leq & \delta_{cmd_k} \leq \delta_{cmd_{max}} \end{aligned} \tag{5.9}$$

It is worth to note that in the optimal control problem in Eq. (5.6) there is no reference to vehicles other than ego, thus safety is considered here only in terms of road boundaries, by appropriately selecting constraints on e_{y_k}. This choice is due to the fact that, in order to integrate safety in π_C, assumptions on future vehicle trajectories and behaviours must be taken. Nonetheless, trajectory prediction is still an open problem, moreover it is usually tackled through DL which would contradict our initial design in Eq. (5.2), where the policy π_C is declared as non-learnable. The solution

to the safety problem will be given in the following sections.

5.2.2 Behavioural Planner via Safe Deep Reinforcement Learning

The behavioural planner is responsible for defining the policy π_D in Eq. (5.2), i.e. a mapping from the sensing state to discrete behaviours.
An AV need to take decision in an unpredictable environment, for that reason the policy component π_L in Eq. (5.3) is designed using model-free DRL because it allows for the extraction of the policy from data, without strict modelling assumptions.
The AD problem, as typically done in the recent literature, e.g. [84, 88, 89], can be modeled as an MDP (see Eq. (2.26)) where the state evolution stochasticity relies in the behaviour of other road users.
In this thesis the action space A is the collection of seven high-level actions or behaviours a_i with $i \in [0, 6]$, summed up in Tab. 5.1. The high-level decisions comprise of *change lane to the left*, *change half lane to the left*, *change half lane to the right*, *change lane to the right*, *faster*, *slower* and finally *idle*. The *faster* action increases the velocity reference v_{ref}, whereas the *slower* action reduces it. Finally the *idle* action keeps the velocity v_{ref} and lane reference constant.

Table 5.1. Description of high-level actions a_i available in the action space A.

Action	Description
a_0	change lane to the left, keep same speed
a_1	change half lane to the left, keep same speed
a_2	keep same lane, keep same speed
a_3	change half lane to the right, keep same speed
a_4	change lane to the right, keep same speed
a_5	keep same lane, increase speed
a_6	keep same lane, decrease speed

The following step is to design the state space S. A common choice for

S is to use a collection of features regarding the ego vehicle and other N closest traffic participants, i.e.:

$$s = \begin{bmatrix} 0 & Y^0 & v_X^0 & v_Y^0 & \psi^0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X^N - X^0 & Y^N & v_X^N & v_Y^N & \psi^N \end{bmatrix} \in S \triangleq \mathbb{R}^{(N+1)\times 5}, \tag{5.10}$$

where X and Y are the position coordinates in the Cartesian frame, v_X and v_Y are the velocities in the same frame, ψ is the angle and, finally, the superscript refers to the i-th vehicle, where $i = 0$ is the ego vehicle. The longitudinal coordinates X are expressed relatively to the ego vehicle so that they can be bounded by the maximum sensing range.
The third, and last, step is to define the reward function $R(s_k, a_k)$ for the MDP. The assignment of a suitable reward is fundamental to steer the driving policy toward the desired behaviour.
The most common design for the reward function is the combination of a positive reward r_+, typically driving speed proportional, and a sparse negative reward r_- issued only when a collision happens. Thus the reward at each time-step is:

$$r_k = r_+ + \alpha r_-, \tag{5.11}$$

where α is a tuning parameter, the positive reward is the normalized driving speed, namely:

$$r_+ = \frac{v_X}{v_{max}}, \tag{5.12}$$

whereas the negative reward is:

$$r_- = \begin{cases} 0 & \text{if no collision happened,} \\ 1 & \text{if collision happened,} \end{cases} \tag{5.13}$$

The main idea of the proposed MDP setup is that the agent can choose to accelerate in order to increase the speed v_X, thus obtaining higher rewards, and, simultaneously, changing lane if a slower vehicle is ahead of the ego vehicle (by checking the state in Eq. (5.10)). Clearly the two reward components in Eq. (5.11) are traded off, namely increasing the positive reward will lead to increasing risks of collisions.
For the aforementioned reason, if any DRL approach is naively applied,

the policy π_D aggressiveness and safety is highly related to the reward function (5.11) shaping.
It is argued here that thanks to the policy design as in Eq. (5.3), i.e. safety and effectiveness are completely decoupled, the negative reward is not actually needed, thus $\alpha = 0$. This design is effective in improving safety, as well as simplifying the learning process, because by eliminating the sparse reward r_- the reward variance is greatly reduced.
On the other hand the policy component π_N in Eq. (5.3) is not learnt and it is responsible for safety, thus effectively acting as a monitor on π_L, by masking behaviours. Specifically, the policy π_N is a collection of boolean action masks a_{i_m}:

$$a_{i_m} = \begin{cases} 0 & \text{if } a_i \text{ is not safe} \\ 1 & \text{if } a_i \text{ is safe} \end{cases}, \tag{5.14}$$

where the safety classification of a_i is obtained from the RSS (see Sec. 2.2.5). At each policy π_D sample time, the masks in Eq. (5.14) are used to disable corresponding actions, thus the agent can only choose between safe behaviours, even during the exploration phase. This guarantees safety during the whole learning phase, thus reducing variance on the reward and increasing sample efficiency.
The proposed action masking design is independent of the DRL algorithm, thus any can be chosen for learning π_L, whether it is value-based or policy based. In order to demonstrate this, both DQN [12] and PPO [16] agents have been trained using the same MDP formulation (see Sec. 2.3.3 and Sec. 2.3.4 for a brief introduction on such algorithms).

5.3 Validation and Discussion

In this section the simulation environment is described, along with the training numerical results.
The environment scenario is a straight *endless* four lane highway with fifty vehicles randomly generated in terms of positions and speeds. The ego vehicle is spawned in one random lane at the beginning of each episode. In the highway environment, the vehicles other than the ego, are modeled and simulated by leveraging the Intelligent Driver Model (IDM) [65] and Min-

Table 5.2. NMPC constant parameters

Parameter	Value	Parameter	Value
T_s	0.1 [s]	H_p	50 [-]
l_f	1.25 [m]	l_r	1.25 [m]
τ	5 [s]	r	0.5 [-]
q_1	5 [-]	q_2	0.5 [-]
$e_{\psi_{min}}$	-0.6 [rad]	$e_{\psi_{max}}$	0.6 [rad]
a_{min}	-5 [m/s^2]	a_{max}	2 [m/s^2]
δ_{min}	-0.6 [rad]	δ_{max}	0.6 [rad]
$a_{cmd_{min}}$	-5 [m/s^2]	$a_{cmd_{max}}$	2 [m/s^2]
$\delta_{cmd_{min}}$	-0.06 [rad]	$\delta_{cmd_{max}}$	0.06 [rad]
v_{max}	35 [m/s]		

imizing Overall Braking Induced by Lane changes (MOBIL) [66]. These models are used for simulation purposes only, therefore they are not known by the controlled ego vehicle.
The simulation environment, behaviour and trajectory planners have been implemented using Python. In particular the scenarios are built on top of the publicly available *highway_ env*[1]. This allows for code reuse and results comparison to state-of-the-art DRL implementations, on the same environments.
The RSS C implementation[2] is publicly available as well, along with APIs for usage in Python.
The non-linear optimization problem in Eq. (5.6) is implemented by leveraging ForcesPRO [96], a commercial toolbox for fast and efficient solution of optimal control problems. It allows for fast controller design and tuning, as well as automatic generation of highly efficient C code, later to be interfaces with Python. The set of parameters and constraints for the NMPC is given in Table 5.2.
Finally the DRL algorithms, both DQN and PPO, have been trained by using stable baselines 3[3]. Stable Baselines 3 is a set of stable and reliable

[1]highway_env code: https://github.com/eleurent/highway-env
[2]RSS code: https://github.com/intel/ad-rss-lib
[3]SB3 code: https://github.com/DLR-RM/stable-baselines3

implementation of DRL algorithms in Pytorch.
The reuse of public state-of-the-art code has been maximized in this work, thus allowing for code reproducibility, with the aim of pushing this line of research.
A total of five agents have been trained and then numerically compared, namely:

- safe PPO: PPO agent trained for π_L, and masked by π_N
- safe DQN: DQN agent trained for π_L, and masked by π_N
- PPO: state of the art PPO agent [16] trained for π_D
- DQN: state of the art DQN agent [12] trained for π_D
- Naive: simple naive policy, the ego vehicle goes as fast as possible by keeping a safe distance to the leading vehicle and without changing lane

The learnt policies leverage a Multi-Layer Perceptron (MLP) with two hidden layers of 256 neurons. The learning process have been carried out by extensive Monte Carlo simulations across two million steps and multiple random seeds. Each episode lasts maximum $n_s = 400$ steps and the reward is normalized at each step, thus the maximum total cumulative reward across the episode is $R_{max} = 400$. The simulation and algorithms hyper parameters are given in Table 5.3.

All of the implemented policies will be compared quantitatively on two performance indexes, namely normalized return and collision rate. The cumulative return is an important index related to the effectiveness of the policies, specifically the higher the index the better. It is formally defined here as:

$$R = \frac{1}{R_{max}} \sum_{k=1}^{n_s} r_{k_n}, \tag{5.15}$$

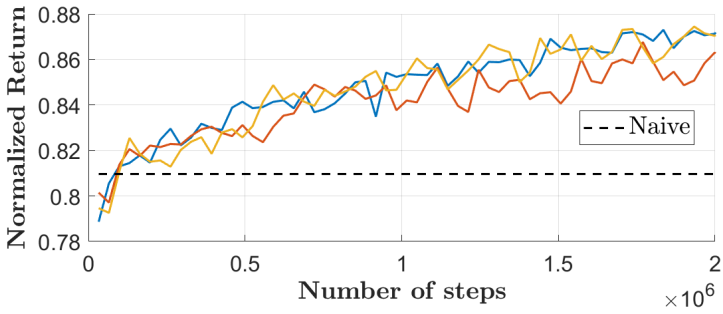
where r_{k_n} is the normalized step reward. On the other hand the collision rate is the main index related to the safety of the policies, specifically the lower the better. It is formally define here as:

$$c_r = \frac{n_c}{n_e}, \tag{5.16}$$

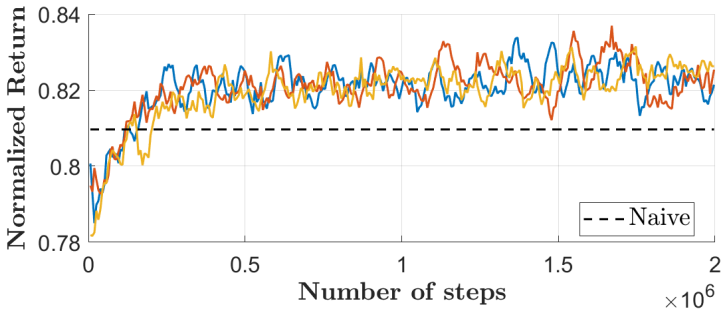
Table 5.3. DRL algorithm training parameters

Agent	Parameter	Value
	Number of training steps	2M
	Policy Scheduling Time	0.1s
	Input neurons	55
	Hidden layers	2
	Hidden layers neurons	256
	Output neurons	5
	Discount factor	0.8
	Learning rate	5e-4
DQN	Replay Memory size	15k
	Initial exploration constant	1
	Final exploration constant	0.1
	Target Network update frequency	50
	Batch size	32
PPO	number of steps	10
	Batch size	64
	GAE λ	0.95
	clipping coefficient	0.2
	value-function coefficient	0.5

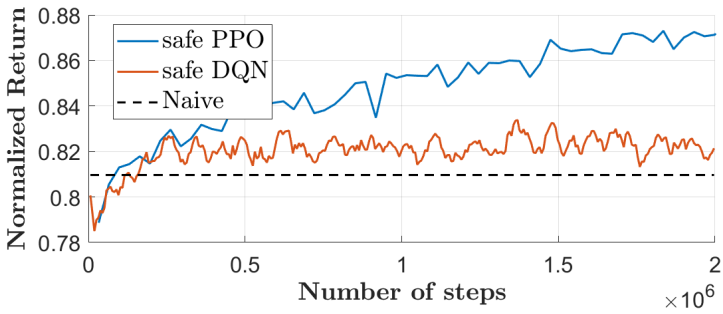
where n_c is the number of episodes in which a collision happened, whereas n_e is the total number of episodes.
Figure 5.2a shows the training result for the safe PPO agent, whereas Fig. 5.2b shows training result for the safe DQN agent, both trained on three different seeds and compared to the Naive agent. The two figures show that both the agents are capable to achieve higher return when compared to the Naive agent, i.e. by changing the reference lane, the autonomous vehicle can achieve higher longitudinal speed. Moreover, the results on show that changing the seed, i.e. changing the simulated scenarios and random exploration, does not affect the results, therefore the learning process is stable. On the other hand, Figure 5.2c shows that the best results in terms of normalized return, i.e. the effectiveness, are achieved by the safe PPO.



(a)



(b)



(c)

Figure 5.2. Safe Reinforcement Learning results. **(a)** Safe PPO agent training result. Normalized total return comparison between the naive policy and the agent trained on three different seeds. **(b)** Safe DQN agent training result. Normalized total return comparison between the naive policy and the agent trained on three different seeds. **(c)** Safe policies comparison. Normalized total return comparison between the naive policy, safe PPO policy and safe DQN policy.

The last agent obtained after two million steps is saved and evaluated on 100 episodes in order to assess the final performances, against the state of the art PPO and DQN, i.e. without the safety mask. The results are summed up in Tab. 5.4. The table shows that the best performances in terms of effectiveness are achieved by the safe PPO, nonetheless perfect safety, i.e. 0% collision rate, is achieved by both safe PPO and safe DQN, which is expected because of the decoupling between in Eq. (5.3).
In conclusion, it is shown how by properly decoupling the driving policy as in Eq. (5.3), it is possible to design a framework in which the policies can be safely learnt. When compared to the state of the art, the proposed approach results in formally guaranteed safety, meanwhile reaching a competitive level of effectiveness. Moreover, the policy decomposition in Eq. (5.2), results in a faster and more stable learning process when compared to end-to-end solutions, thus improving scalability and allowing for its automation.

Table 5.4. Performances of the implemented agents in terms of averaged normalized return and collision rate. The best achieved performance indexes are highlighted in boldface.

Agent	Averaged Normalized Return	Collision Rate
safe PPO	**0.881 ± 0.050**	**0%**
safe DQN	0.835 ± 0.037	**0%**
PPO	0.810 ± 0.172	9%
DQN	0.824 ± 0.175	13%
Naive	0.809 ± 0.003	0%

Chapter 6

Conclusions

This thesis tackles the issue of scaling and increasing the overall safety level of ADAS functionalities, in perspective of autonomous driving.
Chapter 3 shows that by leveraging a monocular front-facing camera, a low-cost FCW system can be designed. The overall system leverages 2D camera frame information only, thus no accurate calibration is needed. Moreover, a co-simulation platform is introduced which greatly ease the software development and testing. Results disclose that, the proposed system performances are comparable to the one obtained by using a RADAR sensor, but for a much cheaper cost, therefore the FCW can be scaled to low-end commercial vehicles.
In Chapter 4 a road-grip aware set of longitudinal ADAS, namely ACC, AEB and ABS is proposed. By implementing model-based estimation techniques, expensive adhesion sensors can be avoided, thus reducing the overall system cost. The potential grip estimation is then leveraged downstream by the ADAS modules, so that safety can be guaranteed with respect to adverse road-grip scenarios, e.g. presence of water, ice or snow on the road asphalt. The proposed system is then validated on a co-simulation platform, on both common highway and urban scenarios. Results disclose that the estimation is very accurate and the system can guarantee safety for any maximum road grip, therefore increasing the safety with respect to the state of the art.
Finally, in Chapter 5 a novel hierarchical planning architecture for AD is proposed. In order to increase effectiveness and scalability, with re-

spect to the state of the art, DRL algorithms are leveraged. Nonetheless, by splitting the policy in a learnable and non-learnable part, it is shown how, safety can be guaranteed formally at all times by employing classical control techniques. When compared to the DRL state of the art, results disclose that safety can be achieved and formally verified, thus avoiding the common posterior statistical analysis.

Bibliography

[1] "Nhtsa data resource website," https://www-fars.nhtsa.dot.gov/Main/index.aspx, accessed: 31-10-2022.

[2] O.-R. A. D. O. Committee, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, apr 2021.

[3] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[4] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005.

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

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

[7] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[8] R. A. Howard, *Dynamic programming and markov processes*. John Wiley, 1960.

[9] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[10] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[13] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[14] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[15] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[17] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[18] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[19] J.-F. Liu, Y.-F. Su, M.-K. Ko, and P.-N. Yu, "Development of a vision-based driver assistance system with lane departure warning and forward collision warning functions," *2008 Digital Image Computing: Techniques and Applications*, pp. 480–485, 2008.

[20] Y.-C. Kuo, N.-S. Pai, and Y.-F. Li, "Vision-based vehicle detection for a driver assistance system," *Computers & Mathematics with Applications*, vol. 61, no. 8, pp. 2096–2100, 2011.

[21] J. Cui, F. Liu, Z. Li, and Z. Jia, "Vehicle localisation using a single camera," *2010 IEEE Intelligent Vehicles Symposium*, pp. 871–876, 2010.

[22] E. L. Raphael, R. Kiefer, P. Reisman, and G. Hayon, "Development of a camera-based forward collision alert system," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 4, pp. 467–478, 2011.

[23] E. Salari and D. Ouyang, "Camera-based forward collision and lane departure warning systems using svm," *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1278–1281, 2013.

[24] H. Kim, Y. Lee, T. Woo, and H. Kim, "Integration of vehicle and lane detection for forward collision warning system," *2016 IEEE 6th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pp. 5–8, 2016.

[25] M. Liu, C.-B. Jin, D. Park, and H. Kim, "Integrated detection and tracking for adas using deep neural network," *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 71–76, 2019.

[26] A. Mulyanto, R. I. Borman, P. Prasetyawan, W. Jatmiko, and P. Mursanto, "Real-time human detection and tracking using two sequential frames for advanced driver assistance system," *2019 3rd International Conference on Informatics and Computational Sciences (ICICoS)*, pp. 1–5, 2019.

[27] W. Hardt, C. Rellan, J. Nine, S. Saleh, and S. P. Surana, "Collision warning based on multi-object detection and distance estimation," *International Symposium on Computer Science, Computer Engineering and Educational Technology*, p. 68, 2021.

[28] J. C. Hayward, "Near-miss determination through use of a scale of danger," in *Highway Research Record*, 1972.

[29] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[32] P. Konstantinova, A. Udvarev, and T. Semerdjiev, "A study of a target tracking algorithm using global nearest neighbor approach," *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'03)*, pp. 290–295, 2003.

[33] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[34] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960.

[35] G. P. Stein, O. Mano, and A. Shashua, "Vision-based acc with a single camera: bounds on range and range rate accuracy," *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings*, pp. 120–125, 2003.

[36] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *Conference on robot learning*, pp. 1–16, 2017.

[37] EURONcap, "Safety assist protocol," https://cdn.euroncap.com/media/56143/euro-ncap-aeb-c2c-test-protocol-v302.pdf, 2020.

[38] E. Raphael, R. Kiefer, P. Reisman, and G. Hayon, "Development of a camera-based forward collision alert system," *SAE International Journal of Passenger Cars-Mechanical Systems*, vol. 4, no. 2011-01-0579, pp. 467–478, 2011.

[39] S. Khaleghian, A. Emami, and S. Taheri, "A technical survey on tire-road friction estimation," *Friction*, vol. 5, no. 2, pp. 123–146, 2017.

[40] B. Leng, D. Jin, L. Xiong, X. Yang, and Z. Yu, "Estimation of tire-road peak adhesion coefficient for intelligent electric vehicles based on camera and tire dynamics information fusion," *Mechanical Systems and Signal Processing*, vol. 150, p. 107275, 2021.

[41] K. Berntorp, R. Quirynen, and S. Di Cairano, "Friction adaptive vehicle control," Sep. 17 2020, uS Patent App. 16/299,285.

[42] Y. Chen and J. Wang, "Adaptive vehicle speed control with input injections for longitudinal motion independent road frictional condition estimation," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 839–848, 2011.

[43] R. Rajamani, N. Piyabongkarn, J. Lew, K. Yi, and G. Phanomchoeng, "Tire-road friction-coefficient estimation," *IEEE Control Systems Magazine*, vol. 30, no. 4, pp. 54–69, 2010.

[44] A. Weißmann, D. Görges, and X. Lin, "Energy-optimal adaptive cruise control combining model predictive control and dynamic programming," *Control Engineering Practice*, vol. 72, pp. 125 – 137, 2018.

[45] A. Weißmann, D. Görges, and X. Lin, "Energy-optimal adaptive cruise control based on model predictive control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 563–12 568, 2017.

[46] C. Sun, L. Chu, J. Guo, D. Shi, T. Li, and Y. Jiang, "Research on adaptive cruise control strategy of pure electric vehicle with braking energy recovery," *Advances in Mechanical Engineering*, vol. 9, no. 11, p. 1687814017734994, 2017.

[47] S. Zhang, Y. Luo, K. Li, and V. Li, "Real-time energy-efficient control for fully electric vehicles based on an explicit model predictive control method," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 4693–4701, 2018.

[48] S. Zhang and X. Zhuan, "Model-predictive optimization for pure electric vehicle during a vehicle-following process," *Mathematical Problems in Engineering*, vol. 2019, 2019.

[49] D. Lee, S. Kim, C. Kim, and K. Huh, "Development of an autonomous braking system using the predicted stopping distance," *International Journal of Automotive Technology*, vol. 15, pp. 341–346, 03 2014.

[50] H. Xiong, Z. Ling, R. Yue, L. Yinong, Z. Zhenfei, L. Yusheng, Z. Qiang, and X. Zhoubing, "Research on control strategy of automatic emergency brake system based on prescan," in *IET International Conference on Intelligent and Connected Vehicles (ICV 2016)*, 2016, pp. 1–6.

[51] S. Naseralavi, N. Nadimi, M. Saffarzadeh, and A. R. Mamdoohi, "A general formulation for time-to-collision safety indicator," *Proceedings of the ICE - Transport*, vol. 166, pp. 294–304, 10 2013.

[52] C. Ilas, "Electronic sensing technologies for autonomous ground vehicles: A review," in *2013 8th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. IEEE, 2013, pp. 1–6.

[53] W. V. WINSUM and A. Heino, "Choice of time-headway in car-following and the role of time-to-collision information in braking," *Ergonomics*, vol. 39, no. 4, pp. 579–592, 1996.

[54] G. Zhenhai, W. Jun, H. Hongyu, Y. Wei, W. Dazhi, and W. Lin, "Multi-argument control mode switching strategy for adaptive cruise control system," *Procedia engineering*, vol. 137, pp. 581–589, 2016.

[55] F. Farroni, "T.r.i.c.k.-tire/road interaction characterization & knowledge - a tool for the evaluation of tire and vehicle performances in outdoor test sessions," *MechanicalSystemsandSignalProcessing*, vol. 72, pp. 808–831, May 2016.

[56] U. Maeder, F. Borrelli, and M. Morari, "Linear offset-free model predictive control," *Automatica*, vol. 45, no. 10, pp. 2214–2222, 2009.

[57] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[58] G. Prasath and J. B. Jørgensen, "Soft constraints for robust mpc of uncertain systems," *IFAC Proceedings Volumes*, vol. 42, no. 11, pp. 225–230, 2009.

[59] USDOT, "Early estimate of motor vehicle traffic fatalities for the first 9 months of 2019," https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812874, 2019.

[60] WHO, "Global status report on road safety 2018," https://www.who.int/publications-detail/global-status-report-on-road-safety-2018, 2018.

[61] J. B. Edwards, "The relationship between road accident severity and recorded weather," *Journal of Safety Research*, vol. 29, no. 4, pp. 249 – 262, 1998.

[62] V. Utkin, J. Guldner, and M. Shijun, *Sliding mode control in electro-mechanical systems*. CRC press, 1999, vol. 34.

[63] W. Pasillas-Lépine, A. Loría, and M. Gerard, "Design and experimental validation of a nonlinear wheel slip control algorithm," *Automatica*, vol. 48, no. 8, pp. 1852–1859, 2012.

[64] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[65] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[66] M. Treiber and D. Helbing, "Mobil: General lane-changing model for car-following models," *Disponıvel Acesso Dezembro*, 2016.

[67] S. Moon, I. Moon, and K. Yi, "Design, tuning, and evaluation of a full-range adaptive cruise control system with collision avoidance," *Control Engineering Practice*, vol. 17, pp. 442–455, 04 2009.

[68] O. Sharma, N. C. Sahoo, and N. B. Puhan, "Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey," *Engineering applications of artificial intelligence*, vol. 101, p. 104211, 2021.

[69] P. Cai, H. Wang, Y. Sun, and M. Liu, "Dignet: Learning scalable self-driving policies for generic traffic scenarios with graph neural networks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8979–8984.

[70] U. Ozguner, C. Stiller, and K. Redmill, "Systems for safety and autonomous behavior in cars: The darpa grand challenge experience," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 397–412, 2007.

[71] A. Furda and L. Vlacic, "Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 1, pp. 4–17, 2011.

[72] E. Balal, R. L. Cheu, and T. Sarkodie-Gyan, "A binary decision model for discretionary lane changing move based on fuzzy inference system," *Transportation Research Part C: Emerging Technologies*, vol. 67, pp. 47–61, 2016.

[73] J. Tang, F. Liu, W. Zhang, R. Ke, and Y. Zou, "Lane-changes prediction based on adaptive fuzzy neural network," *Expert systems with applications*, vol. 91, pp. 452–463, 2018.

[74] K. Macek, M. Becker, and R. Siegwart, "Motion planning for car-like vehicles in dynamic urban scenarios," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 4375–4380.

[75] P. Wang, S. Gao, L. Li, B. Sun, and S. Cheng, "Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm," *Energies*, vol. 12, no. 12, p. 2342, 2019.

[76] D. Madås, M. Nosratinia, M. Keshavarz, P. Sundström, R. Philippsen, A. Eidehall, and K.-M. Dahlén, "On path planning methods for automotive collision avoidance," in *2013 IEEE intelligent vehicles symposium (IV)*. IEEE, 2013, pp. 931–937.

[77] I. Batkovic, U. Rosolia, M. Zanon, and P. Falcone, "A robust scenario mpc approach for uncertain multi-modal obstacles," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 947–952, 2020.

[78] S. Dixit, U. Montanaro, M. Dianati, D. Oxtoby, T. Mizutani, A. Mouzakitis, and S. Fallah, "Trajectory planning for autonomous high-speed overtaking in structured environments using robust mpc," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2310–2323, 2019.

[79] I. Batkovic, M. Ali, P. Falcone, and M. Zanon, "Safe trajectory tracking in uncertain environments," *IEEE Transactions on Automatic Control*, 2022.

[80] P. Hang, C. Lv, Y. Xing, C. Huang, and Z. Hu, "Human-like decision making for autonomous driving: A noncooperative game theoretic approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2076–2087, 2020.

[81] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 9590–9596.

[82] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 33–47, 2020.

[83] S. K. Saksena, B. Navaneethkrishnan, S. Hegde, P. Raja, and R. M. Vishwanath, "Towards behavioural cloning for autonomous driving," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 560–567.

[84] C.-J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2148–2155.

[85] A. Alizadeh, M. Moghadam, Y. Bicer, N. K. Ure, U. Yavas, and C. Kurtulus, "Automated lane change decision making using deep reinforcement learning in dynamic and uncertain highway environment," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1399–1404.

[86] J. Liao, T. Liu, X. Tang, X. Mu, B. Huang, and D. Cao, "Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 177 804–177 814, 2020.

[87] T. Liu, X. Mu, X. Tang, B. Huang, H. Wang, and D. Cao, "Dueling deep q network for highway decision making in autonomous vehicles: A case study," *arXiv preprint arXiv:2007.08343*, 2020.

[88] F. Ye, X. Cheng, P. Wang, C.-Y. Chan, and J. Zhang, "Automated lane change strategy using proximal policy optimization-based deep reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1746–1752.

[89] X. Tang, B. Huang, T. Liu, and X. Lin, "Highway decision-making and motion planning for autonomous driving via soft actor-critic," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4706–4717, 2022.

[90] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[91] B. Brito, A. Agarwal, and J. Alonso-Mora, "Learning interaction-aware guidance policies for motion planning in dense traffic scenarios," *arXiv preprint arXiv:2107.04538*, 2021.

[92] G. Li, Y. Yang, S. Li, X. Qu, N. Lyu, and S. E. Li, "Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness," *Transportation research part C: emerging technologies*, vol. 134, p. 103452, 2022.

[93] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller, "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1205–1212.

[94] G. Kalweit, M. Huegle, M. Werling, and J. Boedecker, "Deep constrained q-learning," *arXiv preprint arXiv:2003.09398*, 2020.

[95] P. Polack, F. Altché, B. Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE intelligent vehicles symposium (IV)*, 06 2017, pp. 812–818.

[96] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, pp. 1–17, 2017.

Author's Publications

1. Santini, S., Albarella, N., Arricale, V. M., Brancati, R., & Sakhnevych, A. (2021). On-board road friction estimation technique for autonomous driving vehicle-following maneuvers. Applied Sciences, 11(5), 2197.
2. Albarella, N., Masuccio, F., Novella, L., Tufo, M., & Fiengo, G. (2021). A Forward-Collision Warning System for Electric Vehicles: Experimental Validation in Virtual and Real Environment. Energies, 14(16), 4872.
3. Arricale, V. M., Maiorano, A., Mosconi, L., Napolitano Dell'Annunziata, G., Rocca, E., & Albarella, N. (2021, August). Improved Anti-Lock Braking System With Real-Time Friction Detection to Maximize Vehicle Performance. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 85369, p. V001T01A002). American Society of Mechanical Engineers.