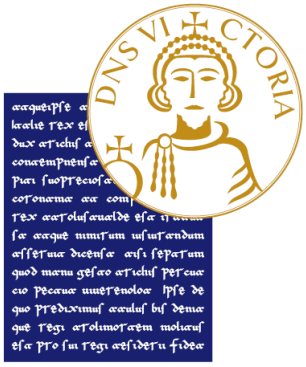
UNIVERSITY OF SANNIO

DEPARTMENT OF ENGINEERING

Ph.D. in
Information Technology for Engineering



Advancements in Perception, Learning and Cooperation for Autonomous Vehicles

SUPERVISOR:
Prof. **Luigi Glielmo**

CO-SUPERVISOR:
Dr. **Davide Liuzza**

COORDINATOR:
Prof. **Massimiliano Di Penta**

Ph.D. CANDIDATE:
Mario Terlizzi
Matr:D50030055

ACADEMIC YEAR 2022/2023

Contents

I dedicate this thesis to my wife Mariangela,
who has always supported me through these years,
to my daughter Beatrice, who gave me the strength to keep going,
and to my friend Enea, who has made my days brighter.

Abstract

This thesis investigates critical advancements in perception, learning, and cooperation mechanisms for autonomous vehicles, addressing key challenges in their development and deployment. It explores how autonomous vehicles can better perceive their environment, learn from dynamic conditions, and cooperate effectively to enhance performance, safety, and reliability.

To tackle these challenges, a comprehensive approach combining theoretical analysis, algorithm development, and practical validation was employed. The research developed an enhanced vision-based path following algorithm for UAVs, a real-time lane detection method, and a reinforcement learning strategy for pedestrian collision avoidance. Additionally, a cybersecurity framework was introduced to detect and mitigate cyber-attacks in real-time, and a robust control system was designed to enhance cooperation between autonomous vehicles in complex environments.

These advancements contribute to the broader goal of integrating autonomous vehicles into everyday life, paving the way for safer, more efficient, and resilient autonomous vehicle systems. This research underscores the potential of combining perception, learning, and cooperation mechanisms to drive the future of autonomous vehicles.

Acknowledgments

I would like to thank everyone who has been involved either directly or indirectly in this Ph.D journey. I would like to express my deepest appreciation to all those who provided me the possibility to complete this thesis. A special gratitude I give to my supervisor, Dr. Luigi Glielmo, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my research.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of the staff of Department od Engineering, who provided invaluable guidance and direction for my studies and research. Their expertise and insights were pivotal in shaping the trajectory of my academic journey, offering both support and challenge when needed to refine my work and focus. The environment they fostered not only facilitated my research but also significantly contributed to my personal and professional growth. I am deeply grateful for their mentorship and the opportunities they have provided me

I would like to extend my deepest gratitude to Davide Liuzza, Valerio Mariani, and Massimo Tipaldi for their invaluable collaboration in the writing of the contributions presented within this work. Their expertise not only enriched this research with profound scientific insights but also provided me with the encouragement and human support necessary to overcome the challenges encountered along the journey. Their dedication and commitment to excellence have been a constant source of inspiration and have significantly contributed to my personal and professional growth.

I am grateful for the beautiful moments spent throughout my Ph.D journey with colleagues Luigi, Enrico, Kishan, and Amit. Because of them, the journey was filled with friendship; thanks to our diversity of cultures, we have enriched each other. All the walks spent talking about our lives, our dreams, and the experiences we had outside the university were varied and very beautiful, and I will carry them in my heart forever.

I extend my deepest gratitude to my family, who have been my cornerstone

and guiding light throughout this journey. They have instilled in me the invaluable virtues of sacrifice and resilience, teaching me to face life's challenges with grace and without a word of complaint. Through their, I have learned the profound beauty of freedom-the freedom to choose my own path, to carve out my destiny, and to live a life that is authentically mine.

Finally, I extend my heartfelt thanks to all the friends who have stood by my side, who have given me strength, and spurred me on to pursue this goal amidst all difficulties, whether they expressed their support directly or indirectly. Their presence, encouragement, and unwavering faith in me have been sources of immense comfort and motivation, pushing me to strive further even when the path seemed insurmountable.

Introduction

The field of Autonomous Vehicle (AV) is rapidly evolving, with significant advancements transforming various industries. As these technologies become more sophisticated, they promise to revolutionize transportation, healthcare, and numerous other sectors by enhancing efficiency and safety. This thesis delves into the structure and contributions of such systems, providing a detailed exploration of their current state, challenges, and future potential.

The contributions detailed in this thesis are instrumental in advancing the state of AV technology across various fields, including: automotive, drones, and railways. By improving perception systems, learning algorithms, and cybersecurity measures, this research addresses both operational unpredictability and digital security threats, paving the way for safer and more reliable autonomous vehicles.

AV stands at the vanguard of technological innovation, spearheading profound transformations across a diverse array of industries. Autonomous systems, which include groundbreaking technologies such as Unmanned Aerial Vehicles (UAVs), self-driving cars and Virtual Coupling (VC) hold the promise of significantly enhancing efficiency, safety, and reliability across numerous applications. These applications span critical sectors, including but not limited to transportation, healthcare, agriculture, and security. This PhD thesis, titled "Advancements in Perception, Learning, and Cooperation for Autonomous Vehicles," delves deeply into the fundamental challenges and pivotal breakthroughs in the development of AV. It places a particular emphasis on advancing perception systems, refining machine learning algorithms, and developing cooperative control architectures, which collectively underpin the next generation of intelligent, autonomous technologies.

AV represent a significant leap forward in the evolution of technology. These systems extend beyond traditional automation by incorporating advanced decision-making capabilities that allow them to operate independently, learn

from their environments, and adapt to new situations. Unlike automated systems, which operate under pre-defined rules, AV possess the ability to learn and evolve, making them capable of handling complex tasks without human intervention.

The impact of AV is profound and far-reaching. In the transportation sector, for instance, self-driving cars and UAVs are poised to revolutionize the way people and goods are transported. These technologies promise to reduce traffic congestion, lower accident rates, and increase fuel efficiency. In healthcare, AV are being developed to assist in surgeries, deliver medications, and provide remote patient monitoring. In agriculture, UAVs are used for precision farming, monitoring crop health, and managing livestock. In security, AV are employed for surveillance, border control, and disaster response.

1 Contributions

Figure 1 presents the primary fields of my contributions on AV in this thesis: computer vision, learning, cybersecurity, and cooperation. These interconnected domains are crucial for advancing the capabilities and ensuring the reliability of autonomous technologies.

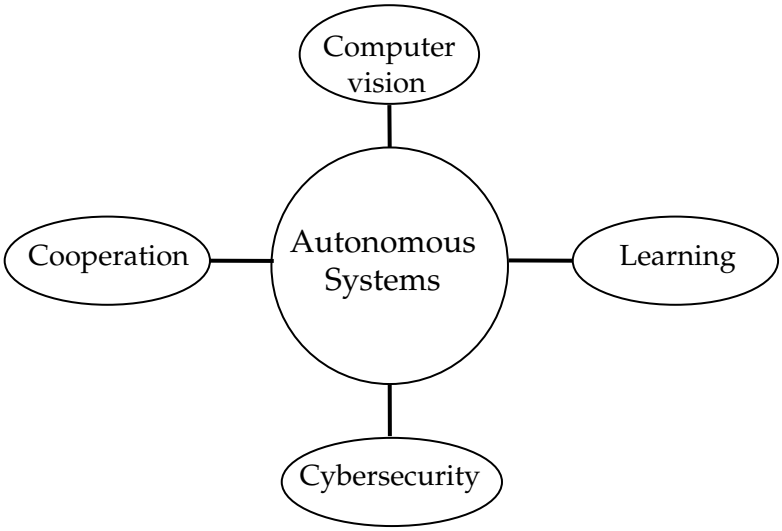


Figure 1: Relevant topics to the Contributions.

The relentless pursuit of autonomous vehicle technology has reached a pivotal stage where perception systems define the frontier of innovation. Within this purview, the contributions presented in Chapters 3 and 4 of this thesis underscore pivotal enhancements in the perception capabilities of unmanned aerial vehicles (UAVs) and autonomous driving systems, respectively.

Chapter 3 delineates a novel perception algorithm for UAVs, which marries the pure pursuit method with advanced image processing techniques. This hybrid approach leverages the computational simplicity of the pure pursuit method while significantly enhancing the UAV's environmental perception and path tracking ability. The algorithm's distinction lies in its capacity to adapt to varied and complex operational environments—a critical factor for autonomous navigation. By integrating real-time image processing, the algorithm can discern and follow paths with unprecedented accuracy and reduced computational demand. This innovation is not only a testament to the robustness required for autonomous vehicular technology but also aligns with the open-source ethos, enabling wide-scale replication and iterative improvements.

Chapter 4 introduces an Iterative Tree Search (ITS) approach for real-time lane detection, a quintessential task for autonomous driving. The ITS algorithm deviates from traditional lane detection methods that rely on extensive feature extraction and complex models. Instead, it proposes a pixel-level analysis, which streamlines the detection process, rendering it time-efficient and more suited for embedded systems within autonomous vehicles. This method reflects an evolution of perception systems towards greater computational efficiency, enabling the detection process to operate within the stringent temporal constraints of autonomous driving.

The common thread uniting these contributions is their focus on improving perception—a sensory cornerstone for AV. Perception is not merely about sensing but involves the interpretation of environmental data to make split-second decisions. In autonomous vehicles, the quality of perception directly influences decision-making algorithms, impacting safety, reliability, and the overall user experience.

The significance of these contributions can be viewed through the lens of their practical applicability. The enhanced path tracking algorithm for UAVs is particularly relevant for search and rescue operations, agricultural monitoring, and surveillance—domains where UAVs must navigate autonomously with high precision. Similarly, the ITS algorithm's implications for autonomous driving extend to urban and highway scenarios where reliable lane detection is crucial for maintaining vehicle trajectory and safety.

Furthermore, these advancements contribute to the overarching goal of creating autonomous vehicles that can operate seamlessly alongside human-driven vehicles. By improving perception systems, we edge closer to harmonizing the coexistence of autonomous and manually controlled vehicles, ensuring a safer and more efficient transportation ecosystem.

In conclusion, the thesis presents two pivotal contributions to the field of autonomous vehicle perception. These innovations not only advance the technical capabilities of AV but also offer insights into the trajectory of future research and development. As the autonomous vehicle industry continues to evolve, the emphasis on perception systems will undoubtedly remain at the forefront, with the potential to redefine transportation as we know it.

This portion of the thesis underscores the advancements in autonomous vehicle (AV) control systems amidst unpredictable scenarios and cybersecurity threats. Chapters 5 and 6 are critical in bolstering the intelligence and safety

frameworks that underpin AV technology.

Chapter 5 presents an innovative approach to AV control using Deep Reinforcement Learning (DRL), specifically adopting the Deep Deterministic Policy Gradient (DDPG) method. This method optimizes the AV's ability to navigate complex environments where pedestrian behavior is unpredictable. The DDPG algorithm is a breakthrough, enabling AVs to learn optimal control policies offline, which are crucial for real-time adaptive driving strategies. The algorithm's efficacy is demonstrated through a series of simulations where the AV must balance pedestrian safety with adherence to its driving trajectory. The results indicate a robust performance in terms of learning speed and quality of the driving policy, ensuring the AV's operational safety in pedestrian-rich environments.

Chapter 6 delves into the realm of cybersecurity, introducing a proactive cyber-attack detection and countermeasure strategy, particularly during critical driving maneuvers such as overtaking. It proposes the Intelligent Model Predictive Control System (I-MPS), an architecture that integrates control, prevention, and mitigation mechanisms against cyber threats. The I-MPS utilizes machine learning algorithms to predict and neutralize cyber-attacks in real-time, preserving the integrity of the AV's control systems. This system is pivotal in ensuring that AVs can maintain safety and performance standards, even in the face of sophisticated cyber-attacks. The chapter illustrates the I-MPS's effectiveness through simulations, highlighting its rapid response to threats and its capacity to uphold safe driving practices.

The contributions detailed in these chapters are instrumental in advancing the state of AV technology. The DDPG algorithm's application to pedestrian unpredictability addresses a critical aspect of urban AV deployment, providing a pathway to more nuanced and responsive driving behaviors. Concurrently, the I-MPS framework equips AVs with the necessary defenses against evolving cybersecurity risks, a concern of increasing significance as AVs become more interconnected.

Moreover, these chapters contribute to the broader discourse on AV reliability and trustworthiness. By enhancing learning algorithms and cybersecurity measures, the thesis advocates for a multifaceted approach to AV safety that accommodates both operational unpredictability and digital security threats. These advancements serve as a foundation for future research, potentially influencing regulations and standards for AV deployment.

In essence, the research presented in these sections not only propels technological capabilities but also addresses societal expectations for secure and intelligent AVs. As the AV industry progresses, the insights from this thesis will be vital in shaping the trajectory of AV development, prioritizing resilience, adaptability, and security in an ever-evolving vehicular landscape.

2 Structure of thesis

The thesis is divided into three parts, each containing a detailed literature review. These reviews synthesize current research, identify existing knowledge gaps, and set the context for the thesis. They not only provide a backdrop for the subsequent analysis but also highlight the innovative contributions of this research in addressing key issues within the field.

Chapter 1 - Background.
This chapter explores the evolution and impact of autonomous systems, emphasizing their transformative role across various sectors. Autonomous systems form the foundation of AV, driving their evolution and transformative impact across various sectors. Autonomous systems excel at perceiving, planning, and controlling within complex environments, using multi-source sensors, advanced software, and machine learning algorithms. They make intelligent, autonomous decisions, enhancing efficiency and safety. By spotlighting their architecture, technological breakthroughs, and future trends, this chapter contextualizes how autonomous systems shape the future of automation, revolutionizing transportation, healthcare, and beyond, while addressing societal implications and challenges.

Part I PERCEPTION

- **Chapter 2 - Enhanced Vision-Based UAV Path Following Strategy.**
  Reports the development of a novel algorithm that synergizes the pure pursuit path tracking methodology with advanced image processing techniques. This integration is specifically tailored to address the computational constraints of micro aerial vehicles (MAVs), enabling them to navigate and complete missions with improved accuracy and efficiency. The theoretical underpinnings of the pure pursuit algorithm are outlined, along with its adaptation to UAV flight dynamics and dynamic path detection

through image processing. The mathematical models and simulation frameworks used to analyze the numerical results provide a solid foundation for the subsequent analysis. Practical validation was achieved through participation in the IFAC2020 MathWorks Minidrone competition, benchmarking the algorithm's performance against international standards.

- **Chapter 3 - An Iterative Approach for Real-Time Lane Detection.**
  Involves introducing an innovative lane detection approach for autonomous driving systems using an Iterative Tree Search (ITS) algorithm. Given the growing need for advanced driver-assistance systems (ADAS) and the evolution of autonomous vehicles, lane detection remains a critical challenge demanding high accuracy and computational efficiency. The ITS algorithm operates directly at the pixel level, bypassing traditional feature extraction methods and complex mathematical modeling. Its lower computational complexity makes it ideal for low-power embedded systems, offering resource-efficient adaptability across various driving conditions. Experimental evaluation in real-world driving scenarios demonstrates the algorithm's robustness in fluctuating illumination and diverse road conditions.

Part II LEARNING

- **Chapter 4 - Reinforcement Learning for Collision Avoidance.**
  Focuses on enhancing pedestrian collision avoidance through Deep Reinforcement Learning (DRL) with the Deep Deterministic Policy Gradient (DDPG) algorithm. This integration of DRL in autonomous driving systems marks a paradigm shift towards adaptive, intelligent, and safer navigation strategies that dynamically respond to the unpredictability of urban environments. The DDPG algorithm's continuous state and action space learning capabilities enable autonomous vehicles to make nuanced decisions that significantly reduce collision risks.

- **Chapter 5 - A Machine Learning-Control Approach for Cybersecurity.**
  Introduces an Informative Model Predictive Scheme (I-MPS) to protect autonomous vehicles from cyber-attacks during overtaking maneuvers. The architecture leverages a Constrained Support Vector Machine classifier informed by Nonlinear Model Predictive Control data to distinguish between normal operation and attack scenarios effectively. This approach is innovative in its adaptability to time-varying or nonlinear dynamics,

ensuring safe and reliable operation amidst increasingly sophisticated cyber threats.

Part III COOPERATION

- **Chapter 6 - Robust and Safe Control Architectures for Virtual Coupling.**
  The increasing demand for railway transportation has strained high-speed railway lines, requiring innovative solutions like VC to enhance capacity and efficiency. VC allows trains to operate at reduced headways by dynamically adjusting their spacing using real-time data, improving both capacity and reducing delays. ERTMS Level 3 sets the stage for VC by transitioning from fixed-block to dynamic systems. This chapter proposes a robust control architecture to manage VC operations, addressing communication delays, packet losses, and uncertainties, with a focus on safety. A simulation tool demonstrates the system's effectiveness in improving railway safety and capacity.

Appendices

- **Appendix A - Optimization and Optimal Control.**
  It explores optimization and optimal control methods, focusing on finding optimal control policies for dynamical systems. It covers problem formulation, including state and control vectors and cost functions, and discusses numerical methods like direct and indirect approaches. Additionally, Model Predictive Control (MPC) is presented as an optimization-based control strategy for dynamic systems.

- **Appendix B - Learning Algorithms.**
  This appendix provides an overview of key learning algorithms in autonomous systems, focusing on Support Vector Machines and Deep Deterministic Policy Gradient. SVMs are supervised learning methods for classification and regression, finding the optimal hyperplane to separate data into two classes by solving an optimization problem. DDPG is a reinforcement learning algorithm within Markov Decision Processes (MDPs) that combines deep learning with reinforcement learning. It uses an actor-critic architecture, where the actor suggests actions and the critic evaluates them, aiming to maximize cumulative rewards. These algorithms enhance classification, decision-making, and overall system efficiency in autonomous applications.

3 Author's Publications

The following list includes my scientific publications, showcasing the contributions and advancements made during my research:

- [1] **Terlizzi, M.**, Silano, G., Russo, L., Aatif, M., Basiri, A., Mariani, V., Glielmo, L. (2021, June). *A Vision-Based Algorithm for a Path Following Problem*. In 2021 International Conference on Unmanned Aircraft Systems (ICUAS) (pp. 1630-1635). IEEE.
- [2] **Terlizzi, M.**, Russo, L., Picariello, E., Glielmo, L. (2021, July). *A novel algorithm for lane detection based on iterative tree search*. In 2021 IEEE International Workshop on Metrology for Automotive (MetroAutomotive) (pp. 205-209). IEEE.
- [3] Russo, L., **Terlizzi, M.**, Tipaldi, M., Glielmo, L. (2021, September). *A Reinforcement Learning approach for pedestrian collision avoidance and trajectory tracking in autonomous driving systems.* In 2021 5th International Conference on Control and Fault-Tolerant Systems (SysTol) (pp. 44-49). IEEE.
- [4] **Terlizzi, M.**, Mariani, V., Glielmo, L. (2021, September). *A Model Predictive Scheme for Autonomous Vehicles Cybersecurity.* In 2021 5th International Conference on Control and Fault-Tolerant Systems (SysTol) (pp. 66-71). IEEE.

The work presented in Chapter 6 is submitted on IEEE Transactions on Control Systems Technology:

- **Terlizzi, M.**, Liuzza, D., Glielmo, L. *A Safe and Robust Control System Architecture for Virtual Coupling.*

Chapter 1

Background

This chapter begins by tracing the origins of Autonomous Systems, from their early conceptualization to their current advanced implementations, emphasizing their transformative impact across various sectors. The motivations behind their ascent are multifaceted, encompassing the need for enhanced efficiency, safety, and the ability to tackle complex tasks beyond human capabilities. We then transition to a detailed examination of the architectural foundational to autonomous systems, discussing its pivotal role in enabling autonomy. This architecture is dissected to reveal its core components and the strategic considerations tailored to this thesis.

Furthermore, the discourse extends to interconnected themes and ongoing innovations, capturing the dynamic trajectory of autonomous systems. We spotlight cutting-edge research and technological breakthroughs shaping their evolution, forecasting future trends that will redefine their capabilities and applications. Through this comprehensive analysis, the chapter sets the stage for a deeper exploration of Autonomous Systems, laying the groundwork for the subsequent discussions and investigations presented in this thesis.

In the conclusion, the contributions made are contextualized against the backdrop of the introduced concepts, underscoring their relevance and impact. This integrative approach not only highlights the advancements and insights offered by this work but also situates them within the broader discourse on autonomous systems. By doing so, it ensures that the theoretical foundations and novel contributions are seamlessly connected, enhancing the overall coherence and depth of the research. This meticulous contextualization serves to underscore the significance of the thesis's contributions, illustrating their potential to influence future developments in the field of autonomous systems.

1.1 Autonomous Systems

Autonomous systems represent a significant leap forward in the evolution of technology, extending beyond traditional automation by incorporating advanced decision-making capabilities that allow these systems to operate independently, learn, and evolve in response to their environment. Automation, defined as the operation of equipment, processes, or systems through mechanical or electronic devices that replace human effort, forms the basis from which autonomy emerges. The hallmark of an autonomous system lies in its shift of decision-making power from central control to individual, trusted components capable of acting without external intervention based on granted permissions.

Distinguishing between automated and autonomous systems is crucial; the former operates under predefined rules flawlessly, whereas the latter possesses the capability for self-learning and evolution. An autonomous system must effectively perceive, judge, and understand its environment, state, and tasks. It reasons through situations to independently devise and select action plans to achieve set objectives. This advanced form of automation integrates intelligence, enhancing its capability to handle tasks beyond pre-programmed scenarios through self-management and guidance. The development of autonomous systems is closely tied to advancements in artificial intelligence (AI) and cognitive technologies, benefiting from the continuous evolution of information technology.

Autonomy is driven by information, or more precisely, by knowledge, enabling systems to autonomously complete the cycle of perception, judgment, decision-making, and action. This allows them to adapt to new situations and task variations with a certain level of fault tolerance. Autonomous systems excel in processing vast quantities of data rapidly, a task that poses significant challenges for human capabilities. They are designed to operate within complex and variable environments, executing a wider range of actions and tasks independently compared to their automated counterparts. These systems are characterized by their integration of multi-source sensors and software capable of complex task processing, enabling them to fulfill designated tasks and goals without or with limited external communication.

The concept of complete autonomy remains a goal, with researchers and technologists worldwide exploring the potential of autonomous systems in various fields. Examples include unmanned aerial vehicles (UAVs) and

unmanned ground vehicles (UGVs) highlighted in strategic roadmaps, aiming to enhance military capabilities through autonomous operations. These developments indicate the future direction of autonomous systems, underscoring their potential to revolutionize our interaction with technology by achieving higher levels of intelligence and mobility. Autonomy, therefore, represents the next stage in the evolution of automation, where systems not only perform tasks independently but also adapt, learn, and evolve to meet new challenges, continuously improving their performance in unknown environments.

1.1.1 History

The conceptual genesis of the term "Autonomous" can be traced back to a pivotal moment in 1946, when Delmar S. Harder, then Vice-President for Manufacturing at Ford Motor Company in the United States, introduced the term 'automation' into the lexicon [5]. This historical landmark not only coined a new term but also laid the foundational principles for the modern understanding of autonomous systems. In this context, the term 'Autonomous' is not merely a linguistic evolution but embodies a profound shift towards self-regulating and independent systems, marking a significant milestone in the technological and industrial narrative.

One of the earliest instances of autonomous systems was demonstrated in December 1926 in Milwaukee by Achen Motors with a modified vehicle known as the "Phantom Auto." This early AVs was based on the Linriccan Wonder, showcasing the potential for vehicles to operate without direct human control. In 1960, RCA Labs further advanced the concept of AVs by presenting a model that was significantly more sophisticated than its predecessors. This model incorporated advancements in electronics and control systems, indicating the potential for AVs to become a practical reality [6]. The efforts of RCA Labs marked a pivotal moment in the development of autonomous systems, demonstrating the feasibility of creating vehicles that could navigate and operate independently of human drivers.

During the 1980s and 1990s, there were significant advancements in the field of autonomous systems. This period witnessed the development and increasing adoption of computer and robotic technologies, laying the groundwork for modern autonomous systems. Specifically, the 1980s marked the onset of using artificial intelligence algorithms and expert systems in industrial and research applications. Progress in sensor technology and data processing enabled robots

to perform increasingly complex tasks with a degree of autonomy. These developments paved the way for innovations inAVs, advanced robotics, and virtual assistants that are prevalent today.

During the 1980s and 1990s, groundbreaking advancements emerged, including the development of pioneering systems such as the inaugural experimental AVs and sophisticated robots tailored for industrial and research applications. For instance, the Defense Advanced Research Projects Agency also called (DARPA) spearheaded initiatives aimed at fostering autonomous land vehicle technologies, culminating in the DARPA Grand Challenge in 2004 [7]. This period delineated a transformative era in technological progression, leaving an indelible mark on the ongoing evolution of contemporary autonomous systems. Moreover, this periods witnessed notable strides in Artificial Intelligence systems. One exemplar is the MYCIN expert system, deployed within the medical realm to diagnose infectious diseases and prescribe treatments. Similarly, Honda's ASIMO robot, unveiled in 2000 following decades of dedicated robotics research, epitomized a significant breakthrough in the mobility and interactive capabilities of humanoid robots [8]. Collectively, these advancements underscored the vast potential inherent in AI and robotics technologies, decisively shaping the trajectory of autonomous technology for the foreseeable future.

In today's landscape, Autonomous Systems Autonomous Systems (ASs) are rapidly evolving and increasingly intersecting with various sectors, significantly enhancing operational efficiency, safety, and fostering innovation. This technological convergence encompasses a wide array of applications, from drones utilized in precision agriculture for enhancing farming practices to AVs deployed in transportation, thereby improving traffic management and reducing emissions. In the realm of healthcare, the utilization of remote diagnostics and patient monitoring systems is advancing medical care accessibility.

Furthermore, the imperative for sustainability is reflected in optimized resource utilization and concerted efforts towards environmental conservation. However, alongside these advancements, there are ethical, legal, and societal considerations that underscore the necessity for comprehensive policy frameworks. These frameworks aim to ensure that the deployment of autonomous technologies benefits society inclusively, while also addressing pertinent issues such as privacy, security, and potential employment challenges.This narrative underscores the pivotal role played by autonomous

systems in shaping future societal landscapes. Notably, their potential to solve complex global issues is highlighted, showcasing their significance in driving positive change. Particularly noteworthy are the applications of autonomous systems in domains such as drones and metropolitan trains, where autonomy is increasingly becoming indispensable [9]. As society progresses, the natural evolution of autonomous systems continues to contribute to societal growth and advancement.

1.1.2 Architecture

Figure 1.1 depicts a schematic representation of the functional architecture of an autonomous system interacting within a dynamic environment. The autonomous system is encapsulated within a larger rectangle, signifying its operation as a cohesive unit. Within this system, the decision-making process is sub-divided into three interconnected modules: Perception, Planning, and Control.

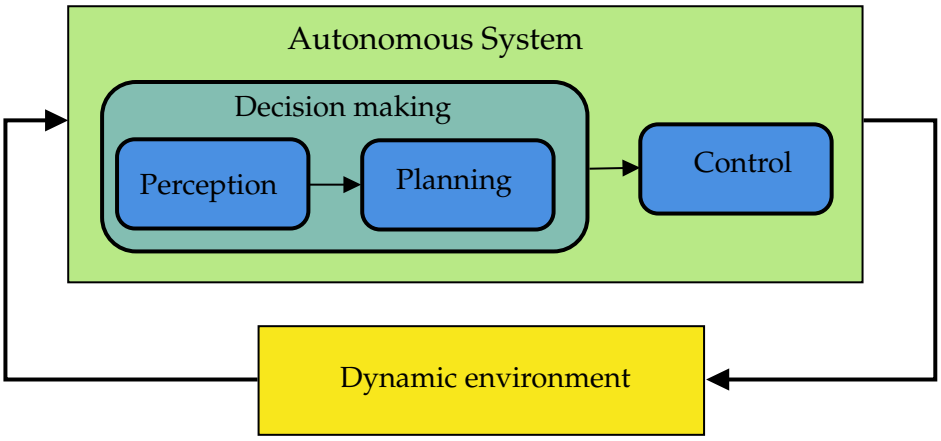


Figure 1.1: Framework Autonomous System.

write it in a formal language without iusing complex words, and maintining the same number of words and concept try to uniform this text to be more readable and compact: Perception, a fundamental aspect of AV technology, encompasses the intricate interplay between the vehicle's sensory inputs and its understanding of the surrounding environment. Through a symphony of proprioceptive, external, and exteroceptive sensors, AVs engage in a continuous dialogue with the world around them, shaping their perception of the road ahead. This perceptual framework, characterized by a diverse array of sensor technologies such as radar, cameras, LiDAR, and GPS, forms the bedrock upon which AVs navigate their operational landscapes.

The choice of sensors is a pivotal decision, with each type offering unique insights into the environment. Cameras, for instance, provide high-resolution visual data, capturing intricate details of the surroundings and facilitating robust object recognition. Radar, on the other hand, excels in detecting objects even in adverse weather conditions, offering a complementary perspective to visual data. LiDAR enhances spatial awareness, mapping out the environment with precision through laser-based technology. GPS, though not a direct sensory input, provides invaluable geospatial context, aiding in localization and route planning.

Yet, beyond the technical intricacies lies the crux of perception: how the AV interprets and synthesizes these disparate streams of sensor data into a cohesive understanding of its surroundings. Perception, in this context, is not merely the passive reception of sensory inputs but rather an active process of sense-making and contextualization. It involves discerning meaningful patterns from the sensor data, anticipating dynamic changes in the environment, and making informed decisions in real-time.

The perception systems of AVs, therefore, serve as the eyes and ears of these autonomous entities, shaping their interaction with the world and influencing their behavior on the road. It is through the lens of perception that AVs navigate complex urban environments, negotiate challenging road conditions, and ensure the safety of passengers and pedestrians alike. As technology advances and perception algorithms grow increasingly sophisticated, the future holds the promise of even greater perceptual acuity for AV, ushering in an era of safer, more efficient transportation systems.

The planning phase, as a pivotal component of the AV decision-making continuum, orchestrates a seamless transition from perception to action, synthesizing environmental cues into a cohesive course of action. At its core, this phase encapsulates a sophisticated interplay between machine intelligence and real-world dynamics, where AVs dynamically chart their trajectory amidst the ever-evolving landscape of their operational milieu.

Within this realm, the AV embarks on a deliberative journey, intricately weaving together inputs garnered from the perception phase with mission objectives and situational imperatives. Whether navigating bustling city streets, traversing expansive highways, or negotiating intricate railway networks, the AV's planning algorithm meticulously tailors its trajectory to align with operational exigencies and safety constraints.

Moreover, the adaptability and agility inherent in the planning process

imbue AVs with the capacity to respond nimbly to emergent scenarios, deftly recalibrating their path in real-time to circumvent obstacles and optimize efficiency. This dynamic recalibration, informed by a fusion of sensory inputs and predictive analytics, ensures that the AV remains resilient in the face of uncertainty, fostering a symbiotic relationship between perception and action.

It is imperative to underscore that the nuances of planning are not universal but rather bespoke to the unique characteristics of each AV archetype. Whether terrestrial, aerial, or rail-bound, the planning paradigm undergoes bespoke tailoring to harmonize with the intrinsic attributes and operational exigencies of the vehicle in question. This bespoke tailoring ensures that the planning process remains attuned to the idiosyncrasies of each domain, optimizing performance and fostering seamless integration within the broader transportation ecosystem.

In essence, the planning phase serves as the linchpin of AV autonomy, catalyzing the translation of perception into purposeful action. By orchestrating a symphony of cognitive processes and environmental awareness, this phase not only steers AVs towards their destination but also paves the way for a future where intelligent transportation systems harmoniously coalesce with the fabric of urban mobility.

Decision-making for autonomous systems involves processes through which these entities make safe and efficient decisions based on environmental information and the state of the system itself. In contexts of autonomous systems, such as AVs, decision-making is critical for safe navigation and interaction with other road users. This decision-making process encompasses a multitude of factors, including real-time sensor data, predictive analytics, and adherence to predefined rules and regulations.

Classical decision-making methods, such as rule-based systems, optimization algorithms, and probabilistic models, provide a structured approach to handling various scenarios encountered by autonomous systems. Rule-based systems rely on a set of predefined rules to guide decision-making, while optimization algorithms aim to find the best possible solution to a given problem, considering constraints and objectives. Probabilistic models, on the other hand, use statistical inference to estimate the likelihood of different outcomes, enabling decision-making under uncertainty.

In contrast, learning-based methods leverage artificial intelligence techniques to improve decision-making capabilities over time. By learning from large volumes of example data, these methods can adapt and evolve to handle complex

and dynamic environments more effectively. Statistical learning techniques, such as support vector machines and decision trees, analyze patterns in data to make predictions or classifications. Deep learning algorithms, which involve neural networks with multiple layers, excel at processing complex sensory inputs, such as images or speech, to make high-level decisions. Reinforcement learning, inspired by behavioral psychology, enables autonomous systems to learn through trial and error, optimizing decisions based on feedback from the environment.

These decision-making methods play a crucial role in enabling autonomous systems to operate autonomously and safely in diverse and unpredictable environments. By combining classical and learning-based approaches, autonomous systems can navigate complex scenarios, anticipate potential hazards, and interact seamlessly with their surroundings, ultimately enhancing safety and efficiency on the road.

Within the domain of AVs, the control sub-system assumes a critical role in the precise execution of planned trajectories. Its primary function is to guide the vehicle along designated paths as determined by the planning sub-system.

A control system is tasked with regulating or maintaining process conditions within a plant to align with desired values. This is achieved by adjusting certain process variables to manipulate the variables of interest, typically the output variables. In the context of AVs, the control system is responsible for generating commands for throttle, brake, and steering inputs, which are essential for directing the vehicle's motion parameters such as position, orientation, velocity, acceleration, and jerk.

It's important to clarify that the designation of input and output variables—often referred to as manipulated/process variables and controlled variables, respectively—is with respect to the plant rather than the controller itself, a distinction that can sometimes cause confusion.

The control system plays a pivotal role within the architecture of an autonomous vehicle. As the final component of the pipeline, it assumes the responsibility for physically maneuvering the vehicle. It is the control sub-system that ultimately dictates the behavior of the ego vehicle and its interactions with the surrounding environment.

While the control sub-system relies on inputs from the perception and planning sub-systems, it is equally valid to assert that the latter two are rendered ineffective if the controller fails to accurately track the prescribed trajectory. Thus, the control sub-system not only complements but also completes the holistic

framework of AV operation, ensuring seamless navigation and interaction within dynamic environments.

1.1.3 Future Trends

The future trends and directions of AVs are shaped by key insights from global expert discussions. The emphasis is on regulatory frameworks acting as catalysts for AV deployment, with a focus on safety, environmental impact, and societal benefits. Innovations in freight and public transport systems, such as drone delivery services and autonomous buses, are highlighted. Challenges like congestion, parking, and urban planning are acknowledged, with strategies for addressing these through technological advancements and policy adjustments. The development of common standards and data sharing protocols is vital for global adoption, alongside advancements in cybersecurity to protect emerging AV ecosystems. Overall, the future of AVs presents a complex interplay of technology, regulation, and societal acceptance, driving towards enhanced safety, efficiency, and environmental sustainability.

Part I

PERCEPTION

Literature review

Lane detection remains a cornerstone within the realm of advanced driver assistance systems, representing a critical element for intelligent autonomous systems and applications in smart vehicles. Notably, Lane Departure Warning Systems (LDWS) emerge as indispensable safety mechanisms, serving to alert drivers when their vehicles deviate from designated lanes on highways [10]. Over recent years, lane detection methodologies have undergone extensive scrutiny, leading to a classification of current studies into three primary categories: feature-based methods [11], [12], [13], model-based approaches [14], [15], and deep learning techniques [16], [17].

Feature-based methodologies typically entail the recognition of lanes through the analysis of lane marking attributes such as color and line edges. In this case, the input images are normally processed with contrast enhancement and edge detection techniques; after that, the lane markings in the image can be detected using a thresholding method, HT [18] or its variants (i.e., Randomized HT [19]). However, a notable drawback of feature-based methods lies in their dependency on clearly delineated lane marks, rendering them susceptible to disruptions such as weak lane markings and occlusions. To address these limitations, the concept of Inverse Projection Mapping (IPM) has been introduced [20]. By generating a bird's-eye view of the road surface, IPM circumvents issues associated with obscured or faint lane markings, although its efficacy relies heavily on the assumption of a perfectly flat road surface and precise camera calibration.

In model-based methods, lane markings are identified through the modeling of lanes using predetermined models such as straight-line, parabolic, or spline models. Various algorithms have been proposed, including combinations of Dynamic Programming (DP) and Hough Transform [14], as well as curve model fitting methods in conjunction with gradient enhancement and edge detection techniques [21]. However, the complexity of developing reliable models poses a significant challenge for model-based approaches.

Deep learning techniques, on the other hand, leverage deep learning algorithms, particularly Artificial Neural Networks (ANNs), to extract lane information. These methods, exemplified by the integration of Convolutional Neural Networks (CNNs) with algorithms like RANdom SAmple Consensus (RANSAC) [16] or Recurrent Neural Networks (RNNs) [17], demonstrate promising capabilities in noise reduction and robust lane detection, especially

in scenarios with challenging conditions such as bad lane markings or vehicle occlusion.

Despite their efficacy, the aforementioned state-of-the-art methods often entail high computational demands, particularly deep learning approaches. In response, this work proposes a novel Lane Detection algorithm based on Iterative Tree Search (ITS), designed specifically for low-cost hardware deployment. Characterized by its speed, computational efficiency, and low power consumption, the ITS-based approach represents a significant contribution to the field, offering a promising alternative to existing methodologies. Notably, to the best of our knowledge, there are no other algorithms for lane detection based on Iterative Tree Search, further highlighting the novelty and potential impact of this research endeavor.

Path following constitutes a significant application challenge within the domain of Unmanned Aerial Vehicles (UAVs), holding particular relevance in precision agriculture settings where robust path following algorithms are essential for maintaining high productivity rates and facilitating optimal plant growth [22]. Moreover, in civilian applications such as power line monitoring, path following encapsulates the essential task of navigating between multiple target regions requiring inspection [23]. Irrespective of the specific application domain, the successful completion of missions by drones hinges upon their ability to safely and accurately follow predefined paths.

Examining the path following problem reveals two distinct components: path detection and path following [24, 25]. In terms of path detection, the Hough transform and its advancements are widely regarded as prominent solutions in the literature s [26, 27, 28]. However, the computational demands associated with such transformations pose challenges, particularly for on-board implementation in aircraft with stringent battery and processing constraints. These challenges are further exacerbated in the context of Micro Aerial Vehicles (MAVs) where both sensor equipment and vehicle dimensions are minimal. Conversely, lightweight machine learning solutions have emerged as promising alternatives for path detection, albeit the time required for algorithm setup impedes their practical application in real-world scenarios [29, 30]. Hence, there is a growing interest in low computational intensity algorithms capable of providing path following references within predefined time constraints, even without prior knowledge of the surrounding environment.

Transitioning to the path following stage, much of the state-of-the-art

solutions [31, 32] rely on the use of nonlinear guidance laws [33], vector fields [34], and pure pursuit algorithms owing to their simplicity and ease of implementation [35]. While the choice of path planner is contingent upon specific application requirements, certain general considerations can be made. Nonlinear guidance laws, while simple to implement, exhibit degraded performance in scenarios characterized by rapid changes in target acceleration, leading to significant trajectory generation delays and instability [33]. Consequently, ensuring stability necessitates a comprehensive understanding of target velocity and acceleration dynamics. Conversely, while vector field solutions mitigate oscillation issues inherent in nonlinear guidance laws, they entail substantial computational overhead which is not suitable for embedded systems present on a UAV [34]. Meanwhile, the pure pursuit approach offers a viable solution for scenarios where tracking accuracy and computational efficiency are paramount. By dynamically adjusting the position of a look-ahead point based on predefined tracking criteria, pure pursuit algorithms aim to minimize the distance between the current position and the anticipated path, thereby facilitating precise path following [36, 37, 38].

Chapter 2

Enhanced Vision-Based UAV Path Following Strategy

In this chapter, we embark on an exploration of a novel algorithm designed for unmanned aerial vehicles (UAVs), focusing particularly on the path following and landing accuracy challenges that are pivotal in autonomous flight operations. The emergence of UAVs has revolutionized numerous sectors, including surveillance, delivery services, and environmental monitoring, by offering unprecedented flexibility and access to remote areas. However, the full potential of UAVs is contingent upon the advancement of robust navigation and control algorithms that can ensure precise maneuvering and operational reliability in diverse environments.

This research delves into the development and validation of an innovative algorithm that synergizes the pure pursuit path tracking methodology with advanced image processing techniques. This integration is tailored to enhance path detection and adherence capabilities, enabling UAVs to navigate and complete missions with heightened accuracy and efficiency. The motivation behind this algorithm springs from the need to address the computational constraints of micro aerial vehicles (MAVs), which necessitate lightweight yet effective solutions for real-time path following and target landing tasks.

The chapter sets the stage by outlining the theoretical underpinnings of the pure pursuit algorithm and its adaptation to the unique dynamics of UAV flight. It further discusses the incorporation of image processing for dynamic path detection, a critical component for ensuring the UAV's adaptability to varying operational scenarios. The mathematical models and simulation frameworks employed to evaluate the algorithm's performance are introduced, providing a

foundation for the subsequent analysis of numerical results.

A significant highlight of this research is its practical validation through participation in the IFAC2020 MathWorks Minidrone competition, a prestigious event that challenges participants to demonstrate the efficacy of their control algorithms in real-world scenarios. This competition not only serves as a rigorous testing ground for the algorithm but also as a benchmark for its success against international standards.

By offering an open-source implementation of the algorithm, this chapter contributes to the broader UAV research community, facilitating further exploration, adaptation, and enhancement of path following techniques. The introduction sets the tone for a detailed investigation into the algorithm's development, its operational merits, and its triumphant validation in a competitive arena, underscoring its potential to redefine UAV navigation and control paradigms.

2.1 Problem Description

The work presented here finds an application within the IFAC2020 MathWorks Minidrone competition [39], where the use of a model-based design approach is the aim of the specific contest. Path error and mission time are used as evaluation metrics for the algorithm. The whole process is the following: a quad-rotor UAV follows a pre-established red path by using its downward facing camera to get feedback from the environment. Images are updated according to the position and orientation of the vehicle simulated in the MATLAB Virtual Reality (VR) world. No prior knowledge on the path and the surrounding scenario is given. The drone takes off and starts its motion looking at the path, and the mission stops with the recognition of an end-marker. At that time, the drone lands staying within the delimited area. Figure 2.1 shows the considered scenario.

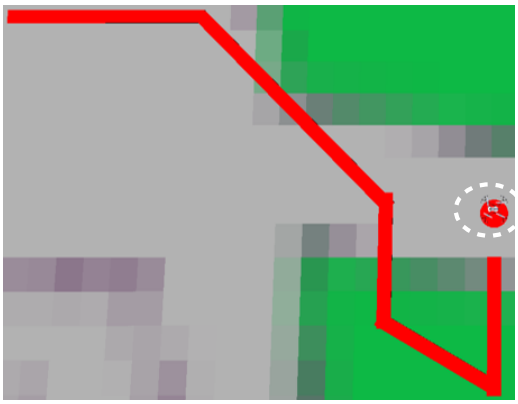


Figure 2.1: Snapshot extracted from the virtual scenario. A dashed circle is used to indicated the drone position.

2.2 Vision-Based Path Following Algorithm

The vision-based path following algorithm combines the advantages offered by the pure pursuit algorithm [40] with that of an easy image processing system to cope with the task. The algorithm starts selecting a target position ahead of the vehicle and that has to be reached, typically on a path. The framework is based on the following operations: (i) given the actual position $\mathbf{d} = (x_d, y_d)^\top \in \mathbb{R}^2$ where the UAV is located, a Virtual Target Point (VTP) is set over the track at $\mathbf{w} = (x_w, y_w)^\top \in \mathbb{R}^2$; then, (ii) the quad-rotor is commanded to reach the VTP along a straight line (essentially it is the pure pursuit algorithm with a curvature

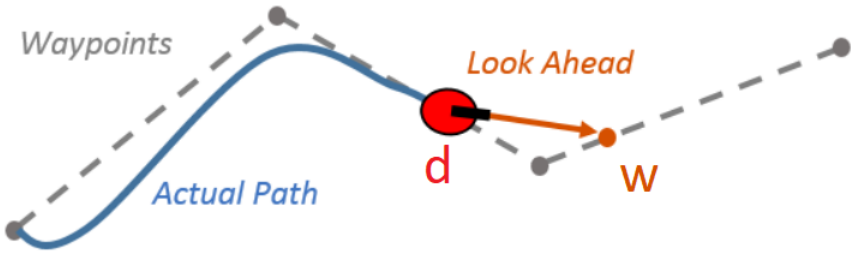


Figure 2.2: An illustrative example of the proposed vision-based path following algorithm works. The red point **d** represents the drone position, while the orange point **w** depicts the VTP.

of infinite radius) [40], i.e., moving the vehicle from its current position to the goal position[1]. In Figure 2.2 an illustrative example of how the algorithm works is depicted.

Contrary to the pure pursuit algorithm, the proposed approach exploits the intrinsic characteristics of the multi-rotor UAVs motion: differently from ground vehicles with steering wheels, drones can follow a path without modifying their heading. Such an assumption allows reducing the time to accomplish the task by removing the control of the heading from the purposes of the path follower.

In Figure 2.3 the whole control scheme architecture is reported. The algorithm is mainly divided into two parts: (i) the Image Processing System (IPS) deals with extracting the red path from the camera images, providing the errors along the x- and y-axis of the camera frame between the current drone position and the VTP point, and recognizing the End-Marker for the landing operations; while, (ii) the Path Planner (PP) figures out the path following problem by computing the new position **w** of the drone in the world frame[41, Sec. V] implementing an Image-Based Visual Servoing (IBVS) scheme. The control algorithm computes the commands u_T, u_φ, u_ϑ, and u_ψ that should be given to the drone in order to update its position and orientation in accordance to the PP references.

The overall mission is divided into four parts: *Take off*, *Following*, *End-Marker*, and *Landing*. A decision-making process has been implemented to achieve the competition objectives triggering the system from a state to another, as depicted in Figure 2.4. For each frame, the IPS accesses the system status and plan the next action (i.e., landing, following, etc.). The drone starts taking off from its initial position looking at the path. Once the vehicle reaches the hovering position, the IPS detects the path and the state machine enters in the *Following* state, hence

[1]The quad-rotor is assumed to fly at a fixed height along the entire mission.

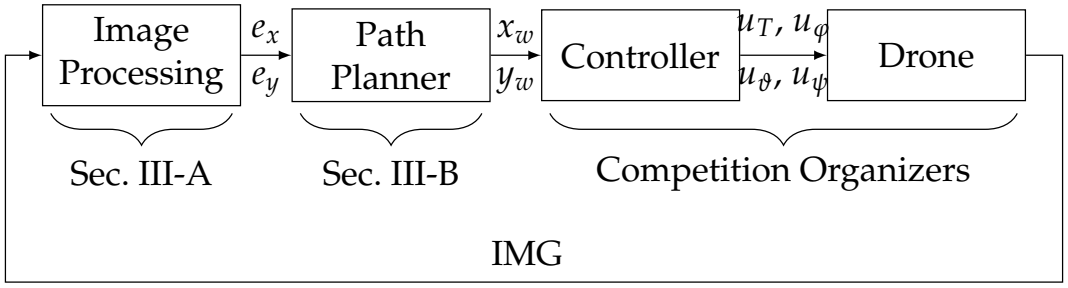


Figure 2.3: Control system architecture. From left to right: the image processing, path planner, controller, and drone blocks.

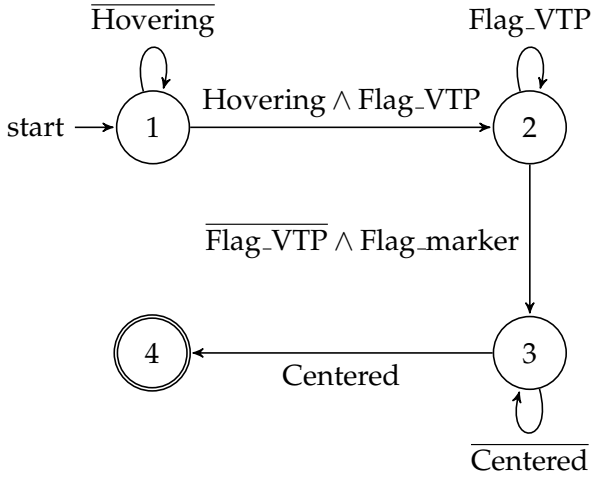


Figure 2.4: State machine implemented. State 1: *Take-off*. State 2: *Following*. State 3: *End-Marker*. State 4: *Landing*.

the path following starts. As soon as the IPS detects the End-Marker, the state machine exits from the *Following* state and goes into the *End-Marker* state. At this stage the mission stops, and the drone starts the landing. In the following subsections the implementation of the image processing system and path planner modules are detailed.

2.2.1 Image processing system

Starting from the camera frames, the IPS takes care of separating the features of the pre-established path from that of the environment.

The IPS receives frames of width W and height H from the camera sensor at each $T_{\text{IPS}} = 0.2\,\text{s}$, i.e., the camera sampling time. The image format is RGB with 8 bits for each color channel. The path is 0.01 m in width, while the landing marker

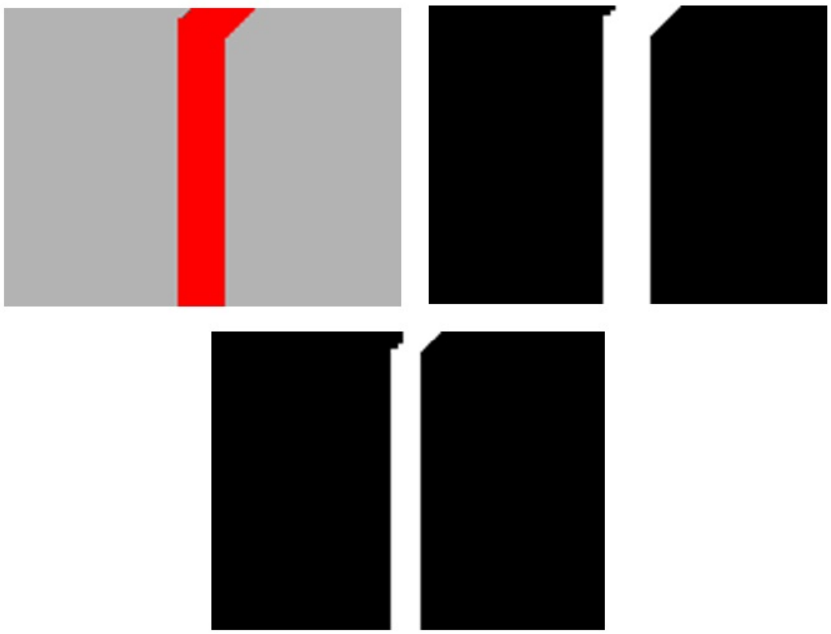


Figure 2.5: Original frame (upper left), converted and binarized frame (upper right), and eroded frame (lower).

is circular with a diameter of 0.02 m. The path color is red, and this information is taken into consideration in all the elaborations to filter out the background scenario. The procedure consists of the following steps: first, the RGB frame is converted into an intensity level frame representation as follows

$$F(n,m) = f_R(n,m) - \frac{f_G(n,m)}{GG} - \frac{f_B(n,m)}{GB}, \tag{2.1}$$

where the pair (n,m) represents the pixel located at row $n \in \{1,2,\ldots,H\}$ and column $m \in \{1,2,\ldots,W\}$ of the image frame and f_i, with $i \in \{R,G,B\}$, provides the intensity level representation of the corresponding red, green and blue channels. An heuristic approach was used to tune the GG, $GB \geq 1$ parameter values. These parameters help to detect the pixels belonging to the path. Further, a binarization process based on a K_T threshold value refines the process removing artifacts from the elaboration. The binarized frame can be described by the binary function $F_{\text{bin}}\colon (n,m) \to \{0,1\}$ whose output is one when the pixel belongs to the path and zero otherwise. Finally, an erosion operation is performed through a square kernel to shrink and regularize the binarized frame. In Figure 2.5 the overall process is reported for a single sample frame.

Then, the obtained reference path is used in a twofold way: (i) to identify a new VTP belonging to the track; (ii) to detect the landing marker. The two tasks are described in the pseudocode reported in Algorithm 1.

Looking at the algorithm, the first three functions (i.e., `channelConv`, `binarization`, and `erosion`) take care of extracting the path information from the frame. Then, the `detectTrack` and `detectMarker` functions deal with raising a flag when the path (`Flag_VTP`) or the End-Marker (`Flag_marker`) are detected. The path following algorithm starts with the IPS that computes the errors (e_x and e_y) between the drone position and the VTP point for the PP by using a circular arc mask centered in the drone Center of Mass (CoM)[2] with thickness $R_{\max} - R_{\min}$[3].

Algorithm 1 Image Processing System

IMG ← channelConv(IMG),
IMG ← binarization(IMG),
IMG ← erosion(IMG),
Flag_VTP ← detectTrack(IMG),
Flag_marker ← detectMarker(IMG)
if Flag_VTP **then**
  $x_{\text{VTP}}, y_{\text{VTP}} \leftarrow$ vtp(frame)
  $e_x \leftarrow x_{\text{VTP}} - x_{\text{CoM}}$
  $e_y \leftarrow y_{\text{VTP}} - y_{\text{CoM}}$
else
  if Flag_marker **then**
    $x_{\text{MARK}}, y_{\text{MARK}} \leftarrow$ cgMarker(frame)
    $e_x \leftarrow x_{\text{MARK}} - x_{\text{CoM}}$
    $e_y \leftarrow y_{\text{MARK}} - y_{\text{CoM}}$
return e_x, e_y, Flag_VTP, Flag_marker

In Figure 2.6, the arc mask considering the VTP position at time $\mathbf{t}_k$ is depicted, where $\mathbf{t}_k$ denotes the k-element of the time interval vector defined as $\mathbf{t} = (0, T_{\text{IPS}}, \ldots, NT_{\text{IPS}})^\top \in \mathbb{R}^{N+1}$, with $k \in \mathbb{N}_0$. The orientation angle $\vartheta = \text{arctan2}(x_{\text{VTP}}, y_{\text{VTP}})$ is calculated with respect to the frame coordinates, where the arctan2 function is the four-quadrant inverse of the tangent function. A portion Θ of the arc mask is established by taking into account the previous VTP's orientation. In particular, we set up two semi-arcs with width $\frac{\Theta}{2}$, namely Field of View (FoV), in counter-clockwise and clockwise directions from ϑ. Then,

[2]The assumption that the CoM being in the center of the reference frame, i.e., $x_{\text{CoM}} = \frac{H}{2}$ and $y_{\text{CoM}} = \frac{W}{2}$, is taken into consideration.

[3]$R_{\max}$ and $R_{\min}$ are the outer and inner radius, respectively.

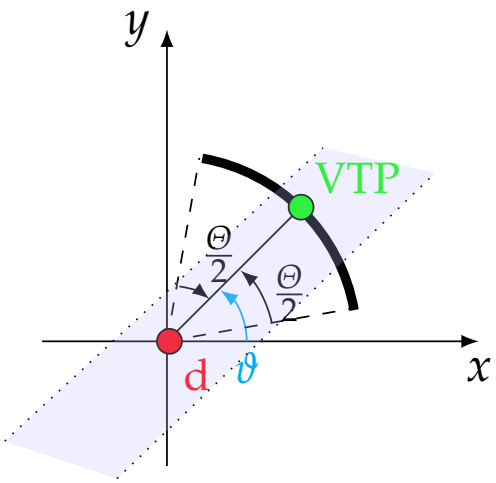


Figure 2.6: Arc mask. The drone position (red), the previous VTP (green), and the pre-established path to follow (purple) are reported.

Figure 2.7: Frame after the application of the Arc mask (left). Extracted pixels belonging to the path (right).

the arc mask is applied to the eroded image obtaining the VTP point at $\mathbf{t}_{k+1}$. The function VTP calculates x_{VTP}, y_{VTP}, and ϑ which represent the frame coordinates and angle orientation of the VTP at $\mathbf{t}_{k+1}$, respectively. Subsequently the corresponding errors with respect to the center of mass, i.e., e_x and e_y, are computed inside the frame coordinates. Finally, the `Flag_VTP` and the e_x and e_y values are provided as input to the PP at each T_{IPS}. Figure 2.7 shows the result of the entire process setup.

It is worth noticing that when the landing marker is detected and no other VTP point is found in the frame, the IPS triggers the state machine in the `End-Marker` state. Here, the new main task of the IPS is to obtain the position of the End-Marker within the frame coordinates. An additional erosion process is

Figure 2.8: Original (left) and eroded frames (right) of the End-Marker are reported.

performed by using a circular kernel, as depicted in Figure 2.8.

2.2.2 Path planner

The PP is designed to compute the position of the VTP point $\mathbf{w} = (x_w, y_w)^\top$ maintaining a constant altitude (z_H) while following the path. Roughly speaking, the PP computes the spatial coordinates x_w and y_w trying to reduce the errors, i.e., e_x and e_y, between the drone position and the VTP. These values are later used by the drone controller to tune the command signals u_T, u_φ, u_ϑ, and u_ψ, as described in Figure 2.3. The proposed path planner is based on Proportial-Integral (PI) control loops. As a common rule in cascade structure, the inner loop, i.e., the PP, is regulated at a rate faster than the outer loop, i.e., the IPS. In our case, the PP runs at 200 Hz ($T_{\text{PP}} = 5 \times 10^{-3}$s) while the IPS runs at 2 Hz ($T_{\text{IPS}} = 0.2$ s). These are a standard solution in the literature for quad-rotors control design [42].

As described in Sec. 2.2, the path following stops with the detection of the End-Marker. At that time, the IPS implements a toggle switch behavior raising the `Flag_marker` flag while holding low the `Flag_VTP` flag. This mutually separates the *Following* and *Landing* phases avoiding instability issues. The pseudocode of the proposed algorithm is reported in Algorithm 2 with parameter values detailed in Table 5.3.

In Subsection 2.3.1, we show how α can be set to control the velocity of vehicle along the entire mission. Therefore, the proposed Vision-Based Path Following algorithm makes it possible not only to generate the spatial coordinates x_w and y_w using a IBVS scheme but also to set the velocity during the entire mission.

Algorithm 2 Path Planner

e_x, e_y, Flag_VTP, Flag_marker
if Flag_VTP **then**
  $x_{k+1} \leftarrow x_k + \alpha e_x$
  $y_{k+1} \leftarrow y_k + \alpha e_y$
  $z_{k+1} \leftarrow z_H$
if Flag_marker **then**
  if $(e_x = 0 \wedge e_y = 0)$ **then**
    $x_{k+1} \leftarrow x_k$
    $y_{k+1} \leftarrow y_k$
    $z_{k+1} \leftarrow 0$
  else
    $x_{k+1} \leftarrow x_k + \beta e_x$
    $y_{k+1} \leftarrow y_k + \beta e_y$
    $z_{k+1} \leftarrow z_H$
$x_w \leftarrow x_{k+1}$, $y_w \leftarrow y_{k+1}$, $z_w \leftarrow z_{k+1}$
return x_w, y_w, z_w

2.3 Numerical Results

To demonstrate the validity and effectiveness of the proposed framework, numerical simulations have been carried out by using the 2019b release of MATLAB equipped with MathWorks Virtual Reality toolbox [43] and Parrot support package for Simulink [44]. The video available at [45] illustrates in a direct way how the system works, i.e., the ability of the quad-rotor UAV to follow the pre-established red path and to land on the End-Marker. In addition, the video shows the behavior of the IPS and PP that never lose the path during the entire mission.

In Figure 2.9 a comparison of the system performance by using various values of α is reported. As can be seen from the plots, the larger α is, the lower the mission time (T_s) is. On the other hand, the lower the mission time is, the greater the path error is. Looking at the zoom plot (see, Figure 2.9d) it is even clearer how the system performance degrades with increasing α value, and these are all the more evident as the path is angular. For the considered scenario, an heuristic approach was used to tune the α and β parameter values.

Figure 2.10 depicts the drone velocity v_x and v_y along the x- and y-axis, respectively, and the norm of the drone velocity v_D. As described in Sec. 2.2.2 and detailed in 2.3.1, the norm of the drone velocity remains approximately constant while following the path. The presence of spikes might be due to the coupling

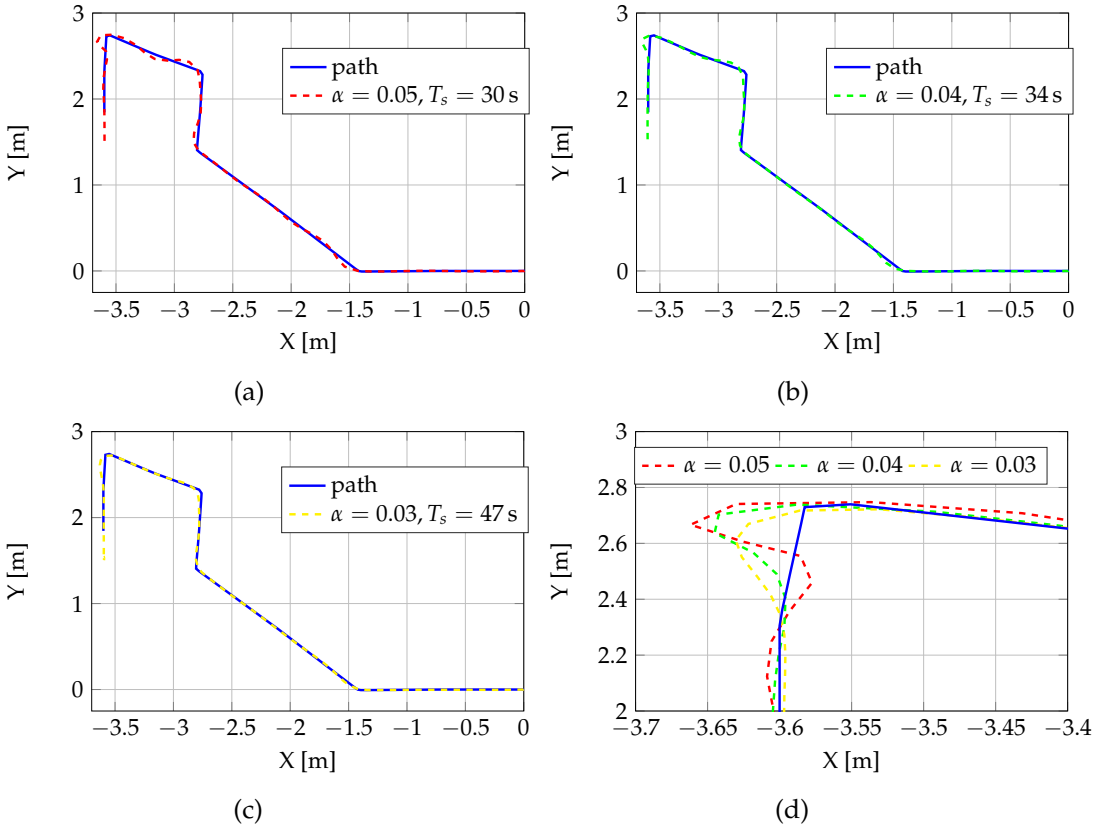


Figure 2.9: Trajectory plots. From left to right: the desired and the drone paths for various values of α are represented. The mission time T_s and a comparison between the considered α values are also reported.

effects of the drone xy dynamics even though x_w and y_w references have not been modified yet (see, Figure 2.9). Such coupling effects are probably caused by the asymmetric positioning of the rotors with respect to the principal axis and the effect of the discrete image pixelization.

2.3.1 UAV Velocity Control

Let us consider a continuous-time dynamical system $\mathcal{H}$ and its discrete time version $x_{k+1} = f(x_k, u_k)$, where $x_k, x_{k+1} \in X \subset \mathbb{R}^n$ are the current state and the next state of the system, respectively, $u \in U \subset \mathbb{R}^m$ is the control input. Let us consider the PP algorithm implementation detailed in Algorithm 2. Hence, the next state of the system x_{k+1} and y_{k+1} along the x- and y-axis can be written as

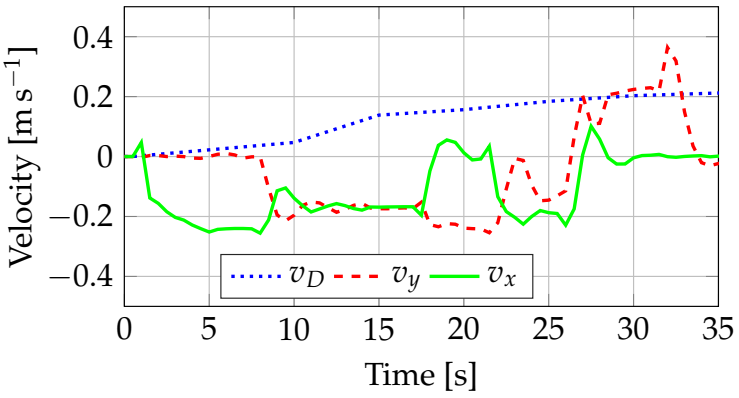


Figure 2.10: Velocity plot.

follows:

$$x_{k+1} = x_k + \alpha e_{x_k}, \qquad y_{k+1} = y_k + \alpha e_{y_k}, \tag{2.2}$$

respectively. After some simple algebra, we can write:

$$\frac{x_{k+1} - x_k}{T_{\text{PP}}} = \frac{\alpha e_{x_k}}{T_{\text{PP}}}, \qquad \frac{y_{k+1} - y_k}{T_{\text{PP}}} = \frac{\alpha e_{y_k}}{T_{\text{PP}}}, \tag{2.3}$$

and hence,

$$v_x \approx \frac{\alpha e_{x_k}}{T_{\text{PP}}} = \tilde{\alpha} e_{x_k}, \qquad v_y \approx \frac{\alpha e_{y_k}}{T_{\text{PP}}} = \tilde{\alpha} e_{y_k}, \tag{2.4}$$

with $\tilde{\alpha} = \frac{\alpha}{T_{\text{PP}}}$.

Knowing that e_{x_k} and e_{y_k} are by definition the projections over a circle along the x- and y-axis of the VTP with an angle ϑ_k, we can write

$$\begin{aligned} e_{x_k} &= \frac{R_{\max} + R_{\min}}{2} \sin \vartheta_k, \\ e_{y_k} &= \frac{R_{\max} + R_{\min}}{2} \cos \vartheta_k, \end{aligned} \tag{2.5}$$

and thus,

$$V_D = \sqrt{v_x^2 + v_y^2} \approx \tilde{\alpha} \frac{R_{\max} + R_{\min}}{2}. \tag{2.6}$$

Hence the parameter α controls the drone velocity.

2.4 Chapter Summary

The chapter provides a comprehensive analysis of a novel winner-prize algorithm designed for path following in UAVs, specifically within the context of the IFAC2020 MathWorks Minidrone competition. This algorithm integrates the pure pursuit algorithm with simple image processing for path detection and tracking, optimized for deployment on MAVs with limited computational capacity. Through MATLAB simulations and the MathWorks VR toolbox, the effectiveness of this approach is validated, emphasizing its practical applicability and open-source availability for replication and further study.

The numerical results section highlights the algorithm's performance through various simulations, demonstrating its capability to efficiently follow a prior established path and accurately land on an end-marker. Adjustments in the algorithm's parameters, such as α, show a trade-off between mission time and path error, providing insights into its adaptability and tuning for specific mission requirements. The velocity analysis further underscores the system's robustness, maintaining consistent velocity while managing the dynamics and discrete pixelization effects inherent in UAV operations.

This chapter culminates in affirming the algorithm's significance, not only through its empirical success in simulations but also by winning a prestigious competition. This accolade serves as a testament to its innovative design, operational efficiency, and potential impact on future UAV path following and autonomous navigation research. The open-source availability of the code further enhances its contribution to the field, encouraging ongoing development and application in various UAV operations.

Chapter 3

An Iterative Approach for Real-Time Lane Detection

In this chapter, we introduce an innovative lane detection approach utilizing an Iterative Tree Search (ITS) algorithm, tailored for real-time application in autonomous driving systems. Amidst the growing need for advanced driver-assistance systems (ADAS) and the evolution of autonomous vehicles, lane detection remains a critical challenge, demanding high accuracy and computational efficiency. The ITS algorithm, by operating directly at the pixel level without relying on traditional feature extraction methods or complex mathematical modeling, presents a novel solution that is both resource-efficient and adaptable to various driving conditions.

Our focus is on demonstrating the algorithm's superior performance, particularly in environments with fluctuating illumination and diverse road conditions. The methodology's uniqueness lies in its lower computational complexity, making it ideal for implementation on less powerful hardware platforms without compromising the detection accuracy or operational speed. This approach not only showcases the potential for significant advancements in real-time lane detection but also opens avenues for its integration into low-cost, low-power embedded systems, paving the way for broader applications in the automotive industry.

The experimental setup, detailed herein, leverages real-world driving scenarios to evaluate the algorithm's effectiveness across different lighting conditions, illustrating its robustness and reliability. Through comparative analysis with state-of-the-art algorithms, we highlight the ITS algorithm's competitive edge in performance, underscoring its feasibility for enhancing

road safety and the autonomy of driving systems. This chapter sets the stage for a comprehensive discussion on the development, implementation, and validation of this groundbreaking lane detection algorithm, aiming to contribute significantly to the fields of autonomous driving and machine perception.

3.1 Proposed Method

The proposed method consists of three main phases: frame acquisition, image preprocessing and Iterative Tree Search (ITS), as shown in Fig. 3.1. Images from the highway environment are acquired through a front-facing camera directed towards the road, the image preprocessing block reports the steps employed to enhance the image features by eliminating unwanted falsification, while the ITS block is the core of the novel lane detection approach proposed in this work.

Figure 3.1: Flow Diagram of the proposed method.

The following assumptions are made for the lane detection problem: the lane boundary lines are made of a light color, (e.g., white or yellow); the road has only an ego-lane and lane boundary lines are not covered from other traffic participants. In the following, the three phases reported in Fig. 3.1 will be discussed in detail.

3.1.1 Frame acquisition

The camera sensor sends Red Green Blue (RGB) format frames to the embedded system. Let $F_i \in \mathbb{R}^{m \times n}$ be the acquired frame channel, where $m \times n$ is the frame size and $i \in \{r, g, b\}$ represent the red, green and blue channel, respectively.

In Figure 3.2 a frame captured from the road scene is presented.

Figure 3.2: Original frame captured from the highway environment.

Figure 3.3: Grayscale conversion and median filtering.

3.1.2 Image Preprocessing

Usually, an image taken from the highway scenario includes objects that are not significant for lane detection purposes (e.g., trees, clouds, parts of sky, car hood) and these irrelevant objects decrease the overall performance of the algorithm, so in order to clean the original frame from unnecessary information, this image preprocessing step is mandatory. The first step of the image preprocessing consist of a conversion from RGB color model to intensity level frame. Intensity-level conversion is performed by the weighted sum of RGB frame as follows [46]:

$$I(u,v) = w_r F_r(u,v) + w_g F_g(u,v) + w_b F_b(u,v), \tag{3.1}$$

where $u \in \{1,\ldots,m\}$ and $v \in \{1,\ldots,n\}$ represent the row and column of the selected pixel and w_r, w_g, $w_b \in \mathbb{R}$ are the conversion weights of RGB channels. The result of the intensity level conversion can be seen in Figure 3.3. This conversion reduces the cardinality of the frame and henceforth the computational load placed on the hardware. Furthermore, in order to improve the performances of the lane detection system, a median filter [47] is used, a particular non-linear technique of digital filtering. The application of the filtering stage has two advantages: i) it allows to reduce the noise and ii) it preserves the edges from the captured frame. Typically, the lane boundary lines are located in the bottom region of a road frame. Taking into account this observation, an isosceles trapezoid Region Of Interest (ROI) operation to restrict the frame information to

the road area is performed. Furthermore, shrinking the frame dimensions means that the computational cost of the detection algorithm is further reduced. This procedure is a well known image preprocessing technique as reported in [48]. The major base, minor base and the height of the isosceles trapezoid ROI expressed in pixels are B_{ROI}, b_{ROI} and H_{ROI}, respectively. From now every operation will done on the ROI frame $I_{\text{ROI}} \in \mathbb{R}^{m_{\text{ROI}} \times n_{\text{ROI}}}$, the latter represents an appropriate rectangle matrix where the elements outside the isosceles trapezoid are considered zero.

In Figure 3.4 the outcome of the image preprocessing block is shown.

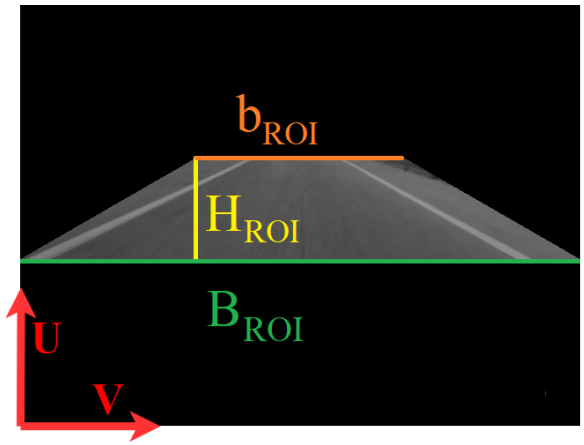


Figure 3.4: Frame after the image preprocessing phase. Frame coordinates (Red). Trapezoid major base (Green). Trapezoid minor base (Orange). Trapezoid height (Yellow).

3.1.3 ITS

The algorithm is in charge to find the pixels belonging to the lane boundary line and is based on an iterative tree search which allows to reduce the lane detection complexity. The concept of this algorithm derives from the idea that the image can be seen as a set of nodes, where these nodes are represented by the pixels. Lane boundaries are solid lines, uniform in color and different from the road color; an edge between the road background and lane boundary lines allows to distinguish the latter from the road in a frame. These structural properties are reformulated from a pixel property point of view.

To explain the ITS algorithm the terms from tree theory are inherited: Root Pixel (RP) node and Child Pixel (CP) node. In particular, the RP node and CP node contain the intensity level of the pixel. Since the ego-lane has only two lane boundary lines, the algorithm is applied in parallel for each of those.

The first step of the ITS algorithm is to find the first pixel belonging to the lane boundary line, i.e, RP node. The first RP node is searched among the pixels present in the first row of ROI as reported in Figure 3.5. Generally, the lane boundary line has a higher intensity level respect to the road; this allows to find the RP node as the pixel having the highest intensity level. In Figure 3.6 the intensity level profile regarding the first row of the ROI is shown, where two peaks are distinguishable that represents the two RP nodes for the two lane boundary lines.

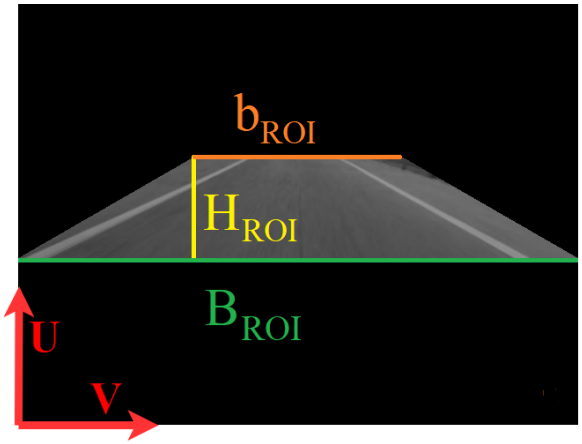


Figure 3.5: Finding the RP nodes. First pixel row ROI (Red). RP nodes (Blue).

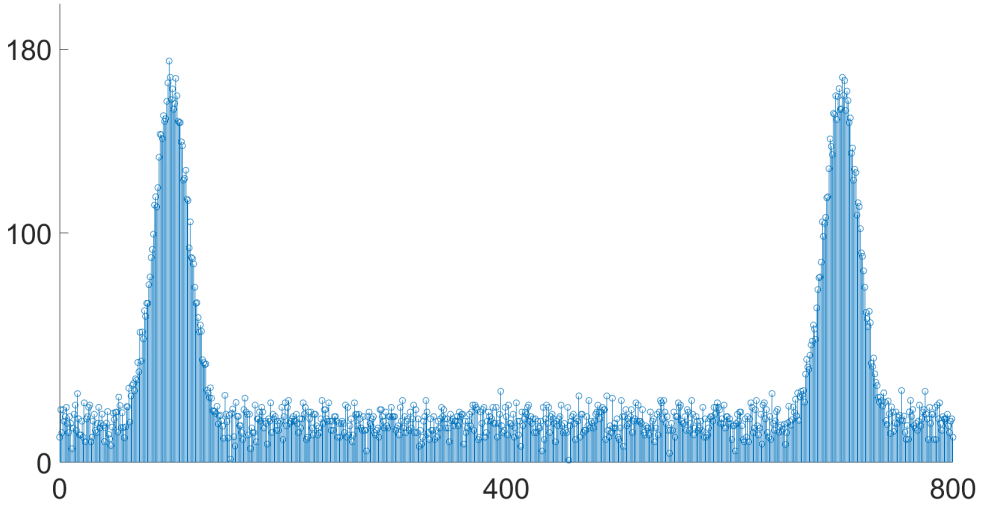


Figure 3.6: Intensity level first row ROI.

Once the first RP node is found, a number N_c of CP nodes are inspected to find another pixel belonging to the lane boundary line on the above row of I_{ROI}.

The choice is based on the following score function J which measures the score obtained from all the CP nodes:

$$J(u,v) = \alpha I_{\text{ROI}}(u,v) - \beta \mid v - v_r \mid + - \gamma \mid I_{\text{ROI}}(u,v) - I_{\text{ROI}}(u_r,v_r) \mid, \qquad (3.2)$$

where u, v (u_r, v_r) represent the row and column of CP node (RP node), respectively. The parameters α, β and γ are used to weight the contributes coming from the three addends. The first addend is the intensity level of the CP node, the second and third addends are the distance and the color dissimilarity between the CP node and root pixel node, respectively. The first term encourage the choice of brighter pixels; the second term penalizes the pixels that are farther from the root pixel, while the third term is added to encourage the uniformity of intensity between the RP and CP. Once the functional scores are calculated for each CP node, the one with highest score become the new RP node. Finally the pair (u_r, v_r) is saved into a memory buffer. The algorithm is iterated until the RP node row reaches the height H_{ROI} of the ROI. In Algorithm 3 the entire procedure is reported.

Algorithm 3 ITS

$u_r \leftarrow 1$
$v_r \leftarrow \text{FindRoot}(u_r)$
$LaneSet \leftarrow (u_r, v_r)$
while $u_r \leq H_{\text{ROI}}$ **do**
    for $v \in \left\{v_r - \frac{N_c}{2}, \ldots, v_r + \frac{N_c}{2}\right\}$ **do**
        score function in (5.5).
    $v_r = \text{argmax}_v \; J(u_r, v)$
    $LaneSet \leftarrow (u_r, v_r)$
    $u_r \leftarrow u_r + 1$
return *LaneSet*

A illustration of the proposed algorithm is presented in Figure 3.7. In Fig. 3.7a the RP node is represented with the yellow square. This pixel has been chosen as the pixel with the maximum intensity between the ones in the first row of ROI. The green squares represent the CP nodes, and for each one of them, the score function presented in (5.5) is evaluated. Then, the new RP node will be chosen among the CP nodes as the one that maximise the score function (Fig. 3.7b).

The ITS algorithm implemented has time complexity $O\big((N_c+1)H_{\text{ROI}}\big)$ that is two order lower than Hough Transform (HT) that is $O\big((mn)^4\big)$ [18], where m and n are the width and the height of the original frame. The minor time complexity

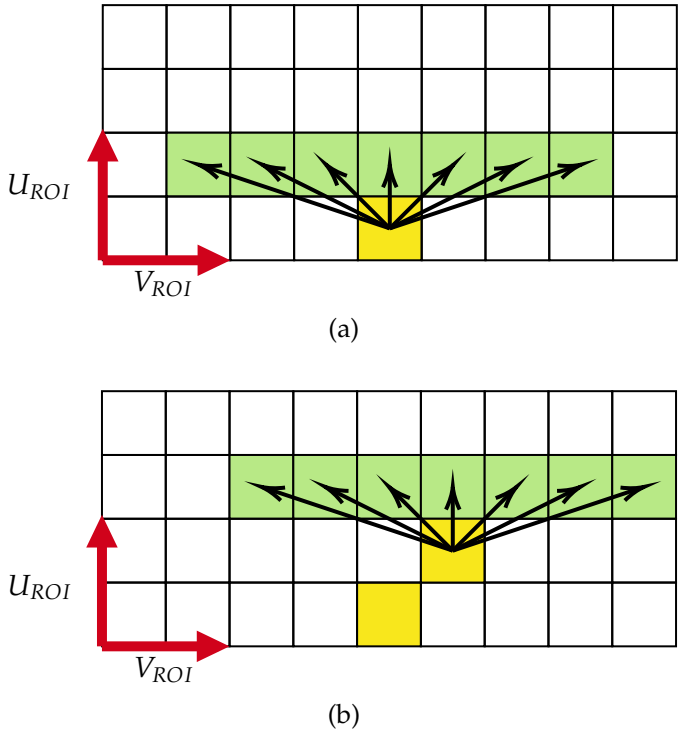


Figure 3.7: Illustration of the algorithm. First iteration (Fig. 3.7a). Second iteration (Fig. 3.7b). Green and Yellow colors represent CP node and RP node, respectively.

permits implementation on embedded systems. Furthermore, the parallelization of the algorithm is possible using a multicore CPU, Graphics Processing Unit or Field-Programmable Gate Array in order to further reduce the computation time.

3.2 Experimental Setup

To evaluate the timing performance of the proposed ITS algorithm, a series of real-world road sequences with different illumination conditions are collected. The data sets cover three kinds of road scenarios, such as: (i) night scenario, (ii) daytime scenario, and (iii) the fluctuating illumination scenario. The experimental setup used for the tests is shown in Figure 3.8. The main components used are two low-cost hardware devices, which will be described below.

The proposed algorithm is implemented using Python 3.6 and OpenCV library on a Nvidia Jetson Nano platform (Figure), an embedded system composed of quadcore Arm 57 CPU with a clock frequency of 1.43 GHz and 4 GB of RAM. A camera sensor Sony IMX219PQH5-C (Figure) is used and mounted

Figure 3.8: Hardware setup.

on the car hood. The camera is a CMOS active pixel type image sensor with a square pixel array and 8.08 Million of pixels. In addition to being a low-cost experimental setup, it also has a total power consumption of 5 W thus making the proposed system an appealing solution for low power applications.

3.2.1 Results

Furthermore, a comparison between computation times of the proposed algorithm and the other state of the art lane detection algorithm has been carried out. The ITS algorithm takes 30 ms to detect both lanes, this means it can achieve 33.3 Frame Per Second (FPS). In [49] and [13], the computation time of the algorithms are 20 ms and 200 ms, respectively; here more powerful hardware was used with a lower frame resolution thus reducing the overall computational effort. The proposed algorithm shows reasonable computation times with a less expensive hardware; furthermore, it allows to parallelize the detection of multiple lane boundaries on a multicore system.

Figure 3.9 presents results of applying the proposed algorithm to scenarios with different illumination conditions. In particular, three scenarios are reported to appreciate the effectiveness of the proposed method: clear sky, night road and different tunnel with multiple illuminations.

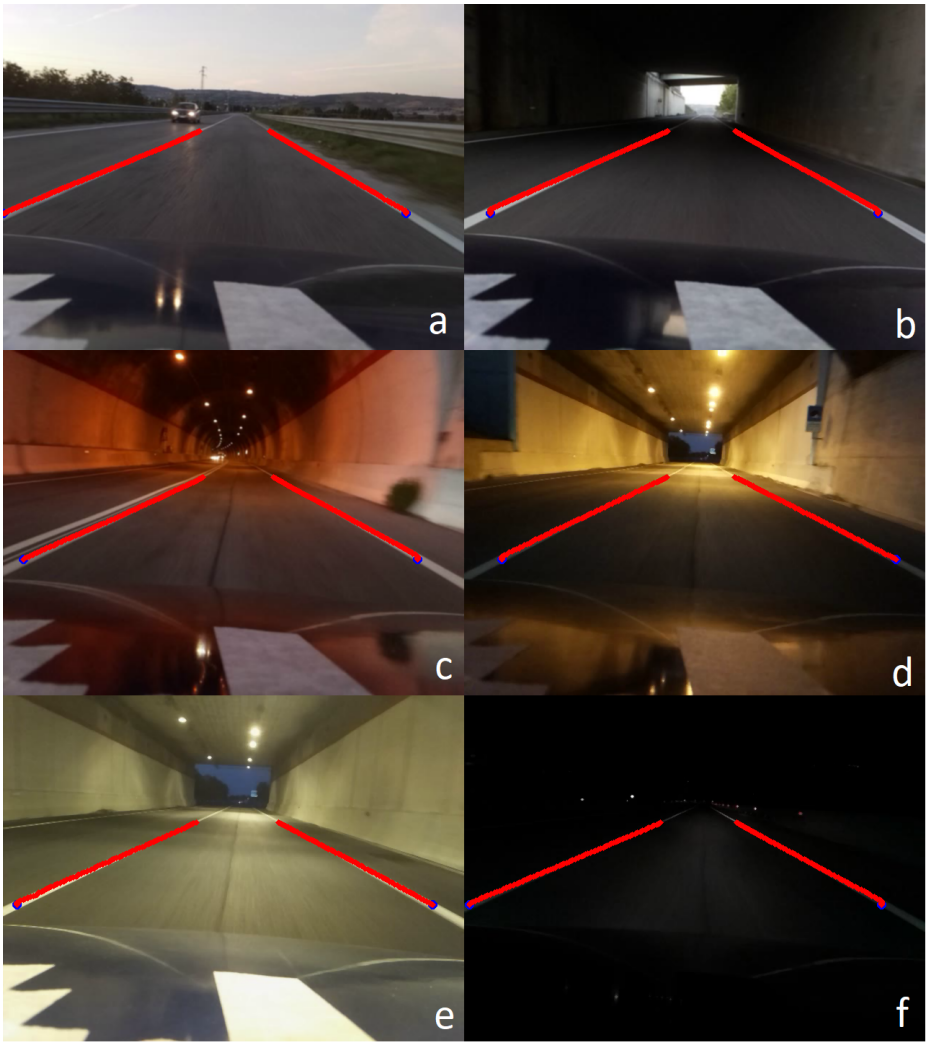


Figure 3.9: Results in different light scenarios. clear sky (a); under different tunnels with different luminosity (b)-(e); night road (f).

By observing the results presented in Fig. 3.9 it is possible to notice that the proposed algorithm manages to precisely track the road lane in all the conditions tested. When there is a change in brightness, for example when the car enters a tunnel, it is possible to notice a minor deviation between the effective position of the lane and the one tracked by the algorithm. Nevertheless, the line detection is still carried out. Good performances are also observed in the condition of scarce brightness, where the tracking is still precise. Generally speaking, by taking into account all scenarios, the proposed method manages to precisely detect a lane, both when there is good brightness and when there is not.

3.3 Chapter Summary

In concluding this chapter, we have demonstrated the effectiveness of the Iterative Tree Search (ITS) algorithm in the realm of autonomous driving, specifically addressing the critical aspect of lane detection. Through rigorous testing and evaluation under diverse conditions, the ITS algorithm has proven to be a robust and reliable solution, offering significant advantages over traditional methods in terms of computational efficiency and adaptability to varying environmental conditions.

The success of the ITS algorithm underscores the potential for significant advancements in real-time lane detection technologies, contributing to safer and more reliable autonomous driving systems. By optimizing for both accuracy and computational resourcefulness, the algorithm ensures its applicability across a wide range of automotive platforms, from high-end autonomous vehicles to more accessible driver-assistance systems.

Part II

LEARNING

Literature Review

Autonomous driving stands as one of the most promising prospects in today's automotive landscape [50]. Nevertheless, vehicles often navigate through intricate scenarios, where various actors with highly unpredictable behaviors coexist. To navigate such complex environments effectively, autonomous vehicles must process multiple facets of their surroundings, forecast environmental evolution, and devise suitable strategies or policies based on predefined objectives. Analogous to human decision-making processes, the predictive abilities and policy selection of autonomous vehicles should draw from their past learning experiences.

Reinforcement Learning (RL), a subfield of Machine Learning, presents a broad spectrum of computational approaches for goal-directed learning through interaction [51]. Particularly in recent years, when combined with Deep Learning, RL has found successful applications in numerous typical driving scenarios involving actors with unpredictable behaviors, such as other vehicles or pedestrians [52, 53]. However, safety considerations impose constraints on these applications, necessitating a careful balance between the inherent benefits of experience-driven learning methods and adherence to safety protocols [54, 55]. Notably, it becomes increasingly prudent to initially train RL algorithms offline via simulators before transitioning to online learning in the actual operational environment.

The unexpected crossing of pedestrians represents one of the most common and critical scenarios in urban areas. According to data from the National Center for Statistics and Analysis [56], approximately 6,000 pedestrians were fatally injured along roadways in the United States in 2017. Given the advancement of autonomous vehicles, the implementation of pedestrian collision avoidance systems assumes paramount importance. This work proposes an approach grounded in Deep Reinforcement Learning for autonomous vehicles navigating scenarios marked by unforeseen pedestrian crossing events. In addition to the imperative of pedestrian avoidance, the resulting agent must adhere to a predefined trajectory.

We adopt a continuous-time state/action space representation for the problem at hand, motivating the utilization of the Deep Deterministic Policy Gradient (DDPG) algorithm for agent training [57]. Both training and testing of the DDPG-based agent are conducted via numerical simulations (i.e., offline) to

ensure compliance with aforementioned safety constraints. The proposed approach offers several advantages inherent to RL-based methods, including the ability to handle complex systems or scenarios difficult to model, learning via simulators or real-world interaction, adaptability of learned policies to environmental changes, and minimal time required for control action selection from the learned policy [51, 58].

While various RL-based approaches have been applied to autonomous driving systems in literature, differences exist compared to our approach, such as the utilization of discrete action spaces that may not accurately represent real car control inputs, differences in problem formulation, and the adoption of online RL approaches [59, 60].

AV technologies represent a glimpse into the future of transportation, with many car manufacturers integrating new functionalities that constitute the foundation of Advanced Driver Assistance System (ADAS), ultimately paving the way for full autonomous driving capabilities [61]. However, amidst this progress, there are significant concerns regarding cybersecurity, which stands to become increasingly pivotal as Internet connectivity emerges as a critical enabling technology. The susceptibility of Control Area Network Control Area Network (CAN) based communication and internet connections exposes vehicles to potential malicious attacks targeting onboard control units.

While the computer science community has extensively addressed cybersecurity in AVs, there remains a notable absence of contributions from the control systems perspective [62]. In efforts to address this gap, recent research endeavors have implemented EKF based architectures to mitigate potential sensor attacks [63], [64]. By employing a residual cumulative sum detector to monitor deviations between predicted and measured states, these approaches offer a means of detecting cyber-attacks. However, the reliance on a single threshold for detection raises concerns regarding false alarms, and strategies for mitigating the impact of detected attacks remain unexplored.

Expanding the purview of cybersecurity analysis from a control systems perspective, significant attention has been directed towards scenarios wherein the plant under scrutiny is modeled as a Linear-Time-Invariant (LTI) system with corresponding control loops. Notably, in [65], the estimation and control of linear systems under cyber-attacks are examined, with the authors proposing an algorithm based on compressed sensing for reconstructing corrupted states. However, limitations emerge, as demonstrated by the impossibility of

reconstructing the state of an LTI system if more than half of the sensors are compromised.

Model Predictive Control (MPC) based solutions have also been explored to bolster the resilience of control loops in the face of cyber threats, as evidenced in [66] and [67] . These studies implement LTI-based solutions have also been explored to bolster the resilience of control loops in the face of cyber threats, as evidenced in [66] and [67] . controllers to enhance control loop robustness when the threatened plant is assumed to be linear and time-invariant. However, while assuming LTI dynamics may not always align with the complexities of AVs, it offers mathematical tractability, which proves advantageous for investigating the problem at hand.

Chapter 4

Reinforcement Learning for Collision Avoidance

As we delve into the complexities of autonomous vehicular navigation in urban settings, the challenge of ensuring pedestrian safety emerges as a critical concern. This chapter presents a comprehensive study on enhancing pedestrian collision avoidance through the application of Deep Reinforcement Learning (DRL), with a specific focus on the Deep Deterministic Policy Gradient (DDPG) algorithm. The integration of DRL in autonomous driving systems represents a paradigm shift towards more adaptive, intelligent, and safer navigation strategies that can dynamically respond to the unpredictability of urban environments. This chapter not only explores the theoretical underpinnings of DRL and its suitability for continuous state and action spaces but also provides a meticulous examination of the DDPG algorithm's architecture, highlighting its potential to revolutionize pedestrian safety mechanisms in autonomous vehicles.

The urgency for such advancements cannot be overstated, as pedestrian safety remains a paramount concern in the development of autonomous driving technologies. Traditional approaches often fall short in dynamically complex scenarios, where the ability to predict and react to pedestrian movements in real-time is crucial. The DDPG algorithm, with its capability to learn optimal policies over continuous action spaces, offers a promising solution to this problem, enabling autonomous vehicles to make nuanced decisions that significantly reduce the risk of collisions.

This work proposes an approach based on Deep Reinforcement Learning for autonomous vehicles operating in a scenario featured by unexpected pedestrian crossing events. Besides the capability of avoiding pedestrians, the resulting agent has to follow a given trajectory. We adopt a continuous time state/action space representation for the problem at hand, and this choice motivates the usage of the Deep Deterministic Policy Gradient (DDPG) algorithm to train the agent [68]. Both the training and the testing of the DDPG based agent are performed via numerical simulations (i.e., off-line) in order to meet the above mentioned safety constraints. The main advantages of using the proposed approach also resides in the typical benefits of applying RL based methods, such as the possibility of dealing with complex systems/scenarios difficult to be modelled, the possibility of learning via simulators or by interacting with the real environment, the adaptability of the learnt policy to the environment changes (thanks to RL intrinsic on-line continuous learning capabilities), the negligible time needed to select the control action from the learnt policy [51, 58]. In literature, we can find different RL based approaches applied to autonomous driving systems. However, some differences versus our approach can be noticed, such as the usage of discrete action spaces which does not represent accurately the real car control inputs, the problem formulation, and the adoption of on-line RL approaches [59, 60].

4.1 The proposed RL system for the pedestrian collision avoidance problem

This section describes all the elements of the RL system for the pedestrian collision avoidance problem, including the reward function definition. Both the agent architecture and the environment model are described, the latter needed for training and testing the DDPG based agent via numerical simulations.

We consider the intersection scenario shown in Fig. 4.1, where an autonomous vehicle has to follow a straight trajectory planned by a local path planner and to manage unexpected pedestrian crossings. A DDPG based agent is designed and trained to achieve such two competing objectives.

More specifically, such scenario is composed by a straight road (with length $r_l = 50\,\text{m}$ and width $c_l = 10\,\text{m}$) and a crosswalk area (perpendicular to the road direction and placed at $\frac{r_l}{2}$ with a height of $c_h = 2.5\,\text{m}$). The vehicle has to follow the reference trajectory depicted as dashed white line in Fig. 4.1. During any

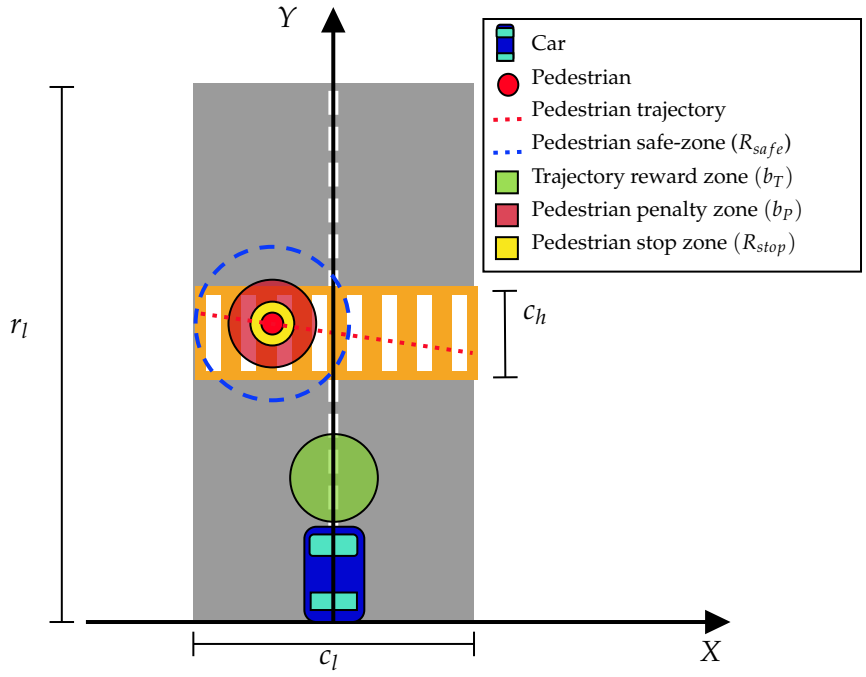


Figure 4.1: The intersection scenario for the pedestrian collision avoidance problem.

travelling episode along the road, a pedestrian can unexpectedly cross the road when the traffic light allows the vehicle to proceed.

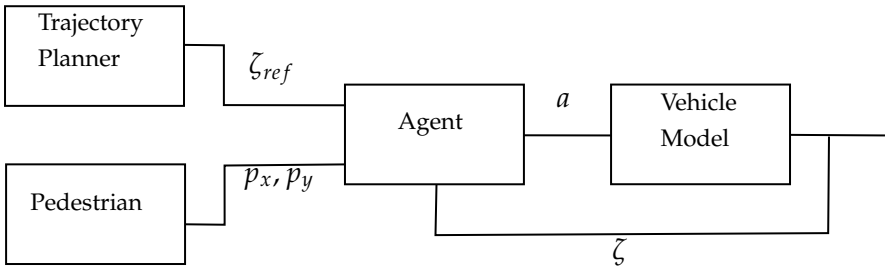


Figure 4.2: The agent-environment interaction for the pedestrian collision avoidance problem.

In such a situation, the trained DDPG based agent has to command the vehicle to avoid the inattentive pedestrian, and then taking again the reference trajectory. The overall RL system has been implemented in Matlab/Simulink. It consists of the DDPG based agent and its environment, that is to say, the vehicle itself, the pedestrian, and the trajectory planner (see Fig. 4.2). The agent interacts with its environment through the vehicle acceleration and steering commands. The stochastic nature of the proposed system is linked to the unexpected behaviour of the pedestrian, who can enter the crosswalk area at any time and by following

an unknown path within such area. In other words, while learning to follow the reference trajectory, the effect of the selected sequence of actions on the expected return G_t is subject to such unknown pedestrian behaviour.

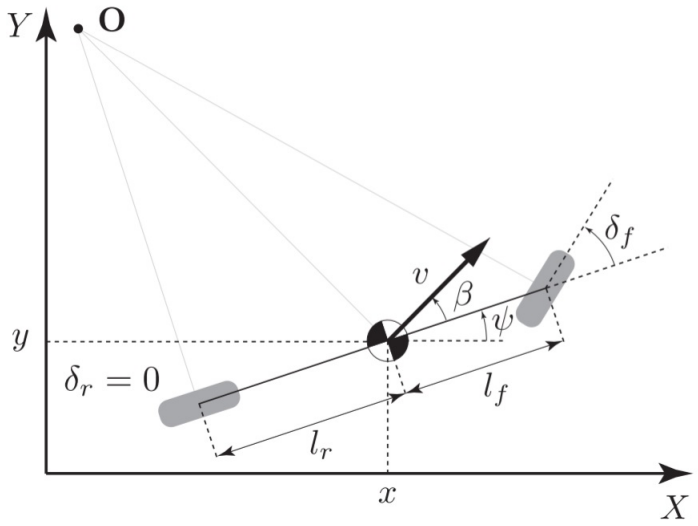


Figure 4.3: The vehicle model.

4.1.1 The vehicle dynamics model

Vehicles operating in an urban environment can be represented via a kinematic based model [69]. In this work, we adopt the model described in [70], which is shown in Fig. 4.3. In such model, the wheels do not slip at their contact point with the ground since both the velocity and the acceleration values are assumed to be low. The following set of nonlinear continuous time equations describes the selected model for the vehicles dynamics

$$\frac{dx(t)}{dt} = v(t)\cos(\psi(t) + \beta(t)), \tag{4.1a}$$

$$\frac{dy(t)}{dt} = v(t)\sin(\psi(t) + \beta(t)), \tag{4.1b}$$

$$\frac{d\psi(t)}{dt} = \frac{v(t)}{l_r}\sin(\beta(t)), \tag{4.1c}$$

$$\frac{dv(t)}{dt} = \mu(t), \tag{4.1d}$$

with [resume]

$$\beta(t) = \tan^{-1}\left(\frac{l_r}{l_f + l_r}\tan\left(\delta_f(t)\right)\right). \tag{4.2a}$$

In our work, for the sake of simplicity, we make explicit the time dependence of the state and action variables only when they are introduced for the first time. In the presented model, x and y are the vehicle mass-center coordinates in an inertial frame, while the heading angle and the speed of the vehicle mass-center are ψ and v. The distance from the vehicle mass-center to the front and rear axles are $l_f = 1.738\,\text{m}$ and $l_r = 1.105\,\text{m}$, respectively. Moreover, β represents the angle of the current mass-center velocity vector with respect to the longitudinal axis of the vehicle, while μ is the mass-center acceleration along such velocity vector. The control inputs are the front steering angles δ_f and the acceleration μ. In other words, we assume that the vehicle is a front wheel steering vehicle, hence $\delta_r \equiv 0$.

The vehicle dynamics model (4.1) can be expressed in a vector form as follows

$$\frac{d\zeta}{dt} = f(\zeta, u), \tag{4.3}$$

where $\zeta = [x, y, \psi, v]' \in R^4$ and $u = [\delta_f, \mu]' \in R^2$ represent the state and input vector, respectively (note that the symbol $'$ denotes the transpose operator).

4.1.2 The pedestrian position and the trajectory planner

The pedestrian position measurements are described by the following equations

$$\begin{cases} p_x(t) = p_{x_0} + v_{x_p} \cdot t + \eta_{\text{x}}, \\ p_y(t) = p_{y_0} + v_{y_p} \cdot t + \eta_{\text{y}}, \end{cases} \tag{4.4}$$

where

- p_{x_0} and p_{y_0} are the coordinates of the pedestrian initial position.
- v_{x_p} and v_{y_p} are the component of the pedestrian velocity vector and are assumed to be constant.
- $\eta_{\text{x}} \sim \mathcal{N}\left(0, \sigma^2_{\text{x}}\right)$ and $\eta_{\text{y}} \sim \mathcal{N}\left(0, \sigma^2_{\text{y}}\right)$ represent the Gaussian noise of the vehicle on-board devices (e.g., RADAR sensors), responsible for the acquisition of the current pedestrian position $p = [p_x, p_y]'$.

We set the values of the pedestrian initial and final position vectors, $p_0 = [p_{x_0}, p_{y_0}]'$ and $p_f = [p_{x_f}, p_{y_f}]'$, as follows

- p_{x_f} and p_{x_0} are chosen according to a Bernoulli random variable $b \sim \text{Bern}(0.5)$ as

$$\begin{aligned} p_{x_f} &= -(2b-1) \cdot \frac{c_l}{2}, \\ p_{x_0} &= (2b-1) \cdot \frac{c_l}{2}; \end{aligned} \tag{4.5}$$

- p_{y_f} and p_{y_0} are Gaussian random variables, namely $p_{y_f} \sim \mathcal{N}\left(\frac{r_l}{2}, \sigma_y^2\right)$, $p_{y_0} \sim \mathcal{N}\left(\frac{r_l}{2}, \sigma_y^2\right)$.

The variances $\sigma_{\text{x}}^2 = \sigma_{\text{y}}^2$ and σ_y^2 are set to $0.2\,\text{m}^2$ and $1\,\text{m}^2$, respectively. The pedestrian moves along a straight path within the crosswalk area at the constant velocity vector $v_p = [v_{x_p}, v_{y_p}]'$.

The local trajectory planner calculates a uniform rectilinear motion $\zeta_{\text{ref}}(t) = [x_{ref}(t), y_{\text{ref}}(t), \psi_{\text{ref}}(t), v_{\text{ref}}(t)]'$ that the vehicle has to follow

$$\begin{aligned} x_{\text{ref}}(t) &= 0, \\ y_{\text{ref}}(t) &= v_{\text{const}} \cdot \sin \psi_{\text{const}} \cdot t, \\ v_{\text{ref}}(t) &= v_{\text{const}}, \\ \psi_{\text{ref}}(t) &= \psi_{\text{const}}. \end{aligned} \tag{4.6}$$

In our training episodes, we set the reference cruise velocity v_{const} to 10 $\frac{\text{m}}{\text{sec}}$ and the reference heading direction ψ_{ref} to $\frac{\pi}{2}$ rad .

4.1.3 The proposed DDPG based agent architecture

The proposed DDPG based agent processes the following two error vectors

$$\begin{aligned} e_T(t) &= [x_{\text{ref}}(t) - x(t), y_{\text{ref}}(t) - y(t)]', \\ e_P(t) &= [p_x(t) - x(t), p_y(t) - y(t)]', \end{aligned} \tag{4.7}$$

where e_T and e_P are the error of the autonomous vehicle with respect to the reference trajectory and its relative distance to the pedestrian, respectively. Hence, the continuous time state and action vectors for the DDPG based agent

can be expressed as follows

$$\begin{aligned} s(t) =& [e_T(t), \frac{de_T(t)}{dt}, e_P(t), \frac{de_P(t)}{dt}]', \\ a(t) =& [\delta_f(t), \mu(t)]'. \end{aligned} \tag{4.8}$$

In this work, we use the agent architecture shown in Fig. B.1. The neural networks are composed of fully connected layers with $N_{\text{nn}} = 100$ number of neurons. The DDPG algorithm [68] is used to train the agent's neural networks. During each episode, the vehicle operates at the intersection scenario shown in Fig. 4.1, while the DDPG based agent updates the parameters of both the actor and the critic neural networks. At each step of the episode, a tuple $(s_t, a_t, r_{t+1}, s_{t+1})$ is stored into a circular buffer, which can hold up to $N_{\text{buff}} = 10000$ samples. The actor and critic parameters are updated by using a mini-batch of $N_{\text{batch}} = 256$ samples, randomly selected from the circular buffer.

4.1.4 Reward function shaping

The reward function selection was driven by the two competing objectives to be achieved by the agent: following a reference trajectory and avoiding unexpected pedestrian crossings. The proposed reward function is composed of five deterministic terms, and depends on all the components of the s and a vectors (4.8), with the exception of their time derivative elements. The tuning of their parameters was the result of a trial and error approach.

To encourage the agent to maintain the reference trajectory, we introduce the following exponential function as the first addend of the reward function (see Fig. 4.4)

$$r_T(s) = r_{\max} \cdot \ (e^{-\alpha_T \cdot \|e_T\|}), \tag{4.9}$$

where $r_{\max} = 15$ is the maximum desired reward in correspondence of the zero trajectory error ($\|e_T\| = 0$) and α_T is a constant gain defined as follows

$$\alpha_T = -\frac{\ln \frac{r*}{r_{\max}}}{b_T}, \tag{4.10}$$

where b_T corresponds to all the trajectory error values e_T with $r_T \leq r^*$ (in our case $r^* = 1$).

Fig. 4.4 shows the traversal section of the two-variable reward function graph (4.9), centered on the zero trajectory error. Note that the reward function

term (4.9) is symmetric with respect to the origin, and has its level sets centered on the zero error trajectory.

As shown in Fig. 4.4, two different values of b_T are considered

$$b_T = \begin{cases} b_{T_{\text{ped}}} & \text{if } \|e_P\| \leq R_{\text{safe}} \\ b_{T_{\text{traj}}} & \text{if } \|e_P\| > R_{\text{safe}}, \end{cases} \tag{4.11}$$

with $b_{T_{\text{ped}}} = 27.8\,\text{m}$, $b_{T_{\text{traj}}} = 1.35\,\text{m}$, and $R_{\text{safe}} = 5\,\text{m}$ is a radius defining a safe zone around pedestrian. This way, as soon as the vehicle enters the safe zone, being very close to the reference trajectory becomes less important, and the agent is forced to focus on the primary objective of avoiding the inattentive pedestrian.

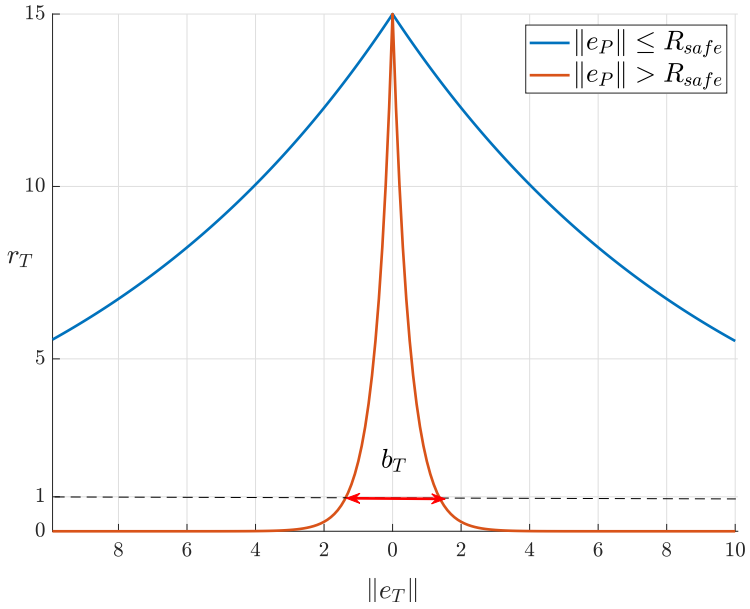


Figure 4.4: Trajectory reward function term.

A similar approach is used to define the reward function term for the pedestrian collision avoidance, In particular, we introduce the following exponential function (see Fig. 4.5)

$$r_P(s) = p_{\max}(e^{\alpha_P \cdot \|e_P\|}), \tag{4.12}$$

where $p_{\max} = -15$ is the maximum penalty in correspondence of the pedestrian

position and α_P is a constant defined as follows

$$\alpha_P = -\frac{\ln \frac{p*}{p_{\max}}}{b_P}. \tag{4.13}$$

Like the previous term, b_P corresponds to all the relative distance values e_P to the pedestrian with $r_P \leq p^*$ (in our case $p^* = 1$ and $b_P = 5.5\,\text{m}$). As shown in Fig. 4.5, the pedestrian reward term is only activated when $\|e_P\| \leq R_{\text{safe}}$. Both trajectory and pedestrian reward term parameters are carefully chosen in order to assure an appropriate reward function shape in the proximity of the pedestrian position as shown in Fig. 4.6, where the maximum reward values are placed at the border of the pedestrian safe zone.

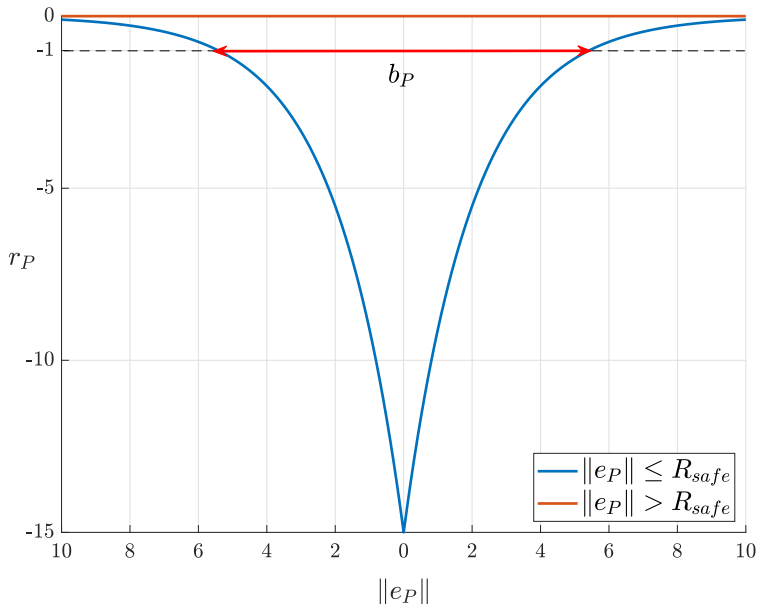


Figure 4.5: Pedestrian reward function term.

The third reward function term is linked to the selected actions

$$r_{\text{act}}(s) = -\alpha_a \cdot (\|a_t - a_{t-1}\|^2), \tag{4.14}$$

where $\alpha_a = 10$ is the weight parameter for the control action. It aims at endorsing smooth variations of the control actions between two consecutive time slots.

Two further reward terms are added to strengthen the learning process. In particular, the training episode is stopped whenever a predetermined distance from the reference trajectory $E_{\max}$ is reached or the pedestrian stopping zone is

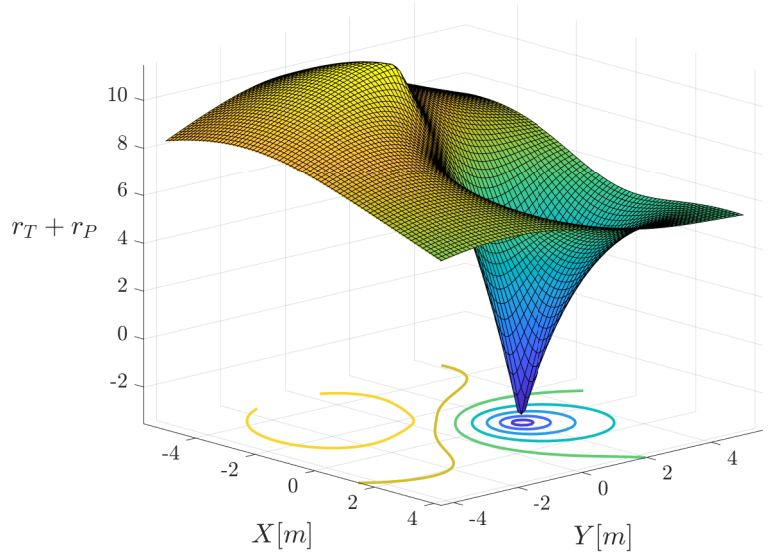


Figure 4.6: Combination of the trajectory and pedestrian reward function terms.

entered. The following two terms are defined

$$r_{\text{end}_P(s)} = \begin{cases} p_P & \text{if } e_P \leq R_{\text{stop}}, \\ 0, & \text{otherwise}, \end{cases} \tag{4.15}$$

$$r_{\text{end}_T}(s) = \begin{cases} p_T & \text{if } e_T > E_{\text{max}}, \\ 0, & \text{otherwise}. \end{cases} \tag{4.16}$$

We set the following values: $p_P = p_T = -50$, $R_{\text{stop}} = 1\,\text{m}$, and $E_{\text{max}} = 15\,\text{m}$. To sum up, the complete reward function is given by the following expression

$$r(s,a) = r_T(s) + r_P(s) + r_{\text{end}_P}(s) + r_{\text{end}_T}(s) + r_{\text{act}}(a). \tag{4.17}$$

4.2 Numerical simulations and results

In this section, we discuss the training process of the DDPG based agent previously described, and then we show the performance achieved by the resulting trained agent.

As expected, the training process was influenced by three main factors: (i) the architecture of the actor and the critic neural networks (along with the values assigned to their hyper-parameters); (ii) the DDPG algorithm configuration parameters (e.g., the size of the mini-batch N_{batch}); the reward function definition. For instance, a small mini-batch of samples did not allow exploring the state/action space in an efficient way. In such a case, both the quality and the speed of the overall learning process were affected. Such issues were also observed when we defined and shaped the reward function. Indeed, the reward function presented in the paragraph 4.1.4 was the result of an engineering effort supported by a trial and error approach.

The training process was carried out for $N_{\text{ep}} = 6500$ episodes. In addition, we set the sampling time T_s to 0.05 sec, the maximum time of each episode T_{sim} to 5 sec, and the discount factor γ to 0.99. The outcome of the training process is shown in Fig. 4.7, where the moving mean value and the moving standard deviation of the total reward computed over the last $N_{\text{av}} = 20$ episodes are plotted. At the beginning of the training phase, the agent learnt to follow quickly the reference trajectory up to the beginning of the pedestrian crosswalk area (thanks to the exploration capabilities of the DDPG algorithm [68]). After being stuck for a short period, the learning process continued, but more slowly. In this phase, we also observed increasing standard deviation values of the total reward (mainly due to the occurrence of pedestrian hits within the crosswalk area). At the end, the total reward mean value stabilised, while the related standard deviation decreased significantly. In particular, the agent learnt to avoid the pedestrian, and the residual standard deviation was only due to the distance of the vehicle from the reference trajectory. We also observed that the agent learnt to follow the reference trajectory beyond the pedestrian crosswalk area.

After the training phase, we tested our agent for $N_{\text{test}} = 10000$ episodes, in which we observed zero pedestrian hits and adequate trajectory tracking performance. Fig. 4.8 shows the distribution of the maximum trajectory tracking error, while Fig. 4.9 plots the distribution of the minimum vehicle-pedestrian distance.

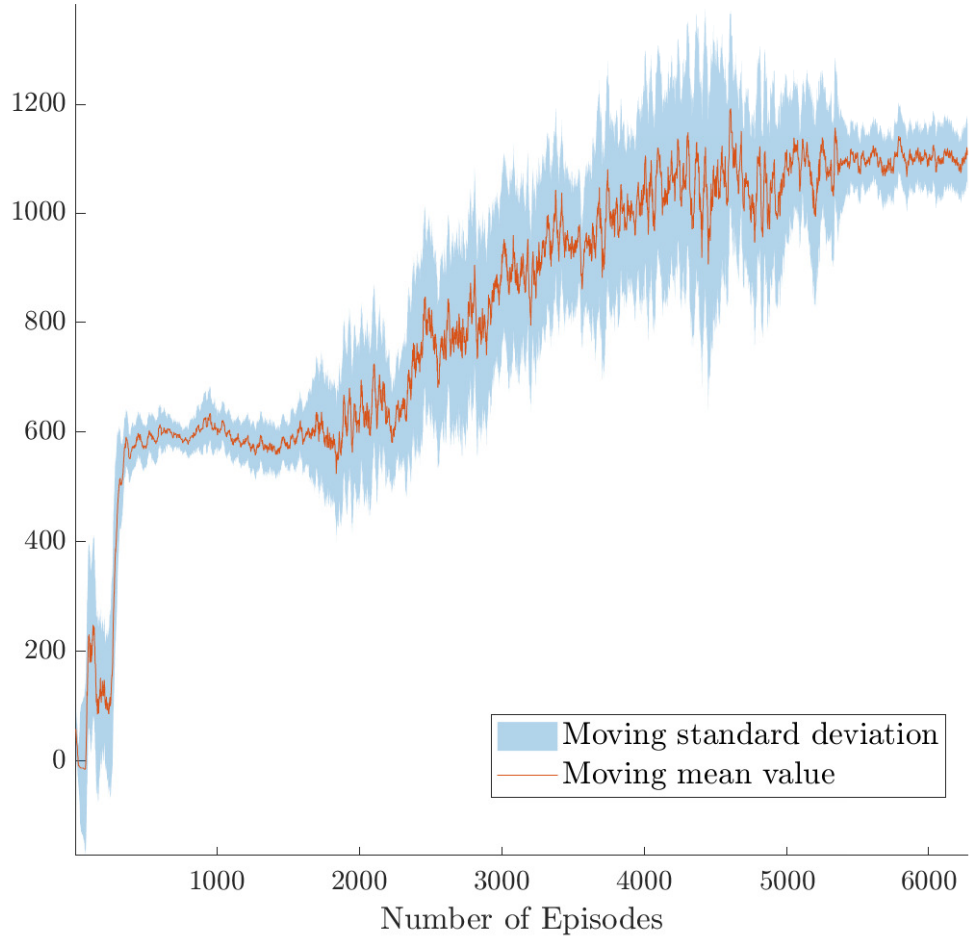


Figure 4.7: Moving mean value and standard deviation of the total reward during the training process.

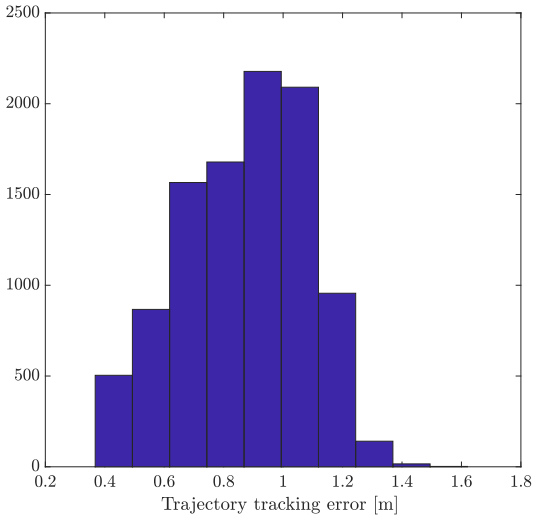


Figure 4.8: Maximum trajectory tracking error distribution.

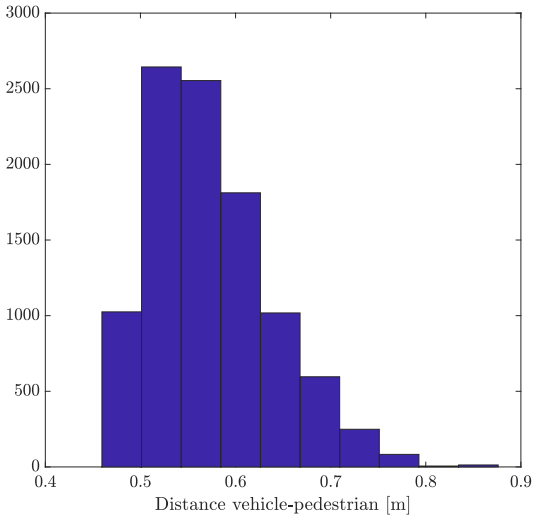


Figure 4.9: Minimum vehicle-pedestrian distance distribution.

4.3 Chapter Summary

In conclusion, the implementation of the Deep Deterministic Policy Gradient (DDPG) algorithm for pedestrian collision avoidance in autonomous vehicles has showcased substantial potential in enhancing urban navigation safety. Through detailed numerical analysis, the algorithm demonstrated a commendable capability in reducing collision risks with pedestrians, a critical aspect of autonomous vehicle deployment in densely populated areas. The numerical results underline the algorithm's effectiveness, showing marked improvements in decision-making accuracy under various simulated urban scenarios. However, it's the inherent adaptability and learning efficiency of the DDPG algorithm that stand out, suggesting that with ongoing refinement, its application could lead to even safer and more precise autonomous driving solutions. By leveraging the strengths and addressing the limitations identified in this study, the DDPG algorithm can significantly contribute to the evolution of autonomous driving technologies, ensuring a safer coexistence between autonomous vehicles and urban pedestrians.

Chapter 5

A Machine Learning-Control Approach for Cybersecurity

This chapter introduces a sophisticated defense and mitigation control scheme designed to protect autonomous vehicles during overtaking maneuvers from cyber-attacks, including Replay Attacks and Denial of Service attacks. The proposed Informative Model Predictive Scheme architecture aims to detect and mitigate these attacks, thereby preserving the safety and integrity of AV control systems. The scheme leverages Machine Learning for attack detection, employing a Constrained Support Vector Machine classifier informed by Nonlinear Model Predictive Control data to distinguish between normal operation and various attack scenarios effectively. This approach is innovative in its adaptability to time-varying or nonlinear dynamics and its ability to implement countermeasures dynamically, showcasing a novel direction in enhancing vehicular cybersecurity. Through simulation and analysis, this chapter seeks to underline the effectiveness of the I-MPS in ensuring the safe and reliable operation of autonomous vehicles in the face of increasingly sophisticated cyber threats.

5.1 Control Strategy Architecture and Problem Formulation

We consider the AV control architecture, depicted in Fig. 5.1, consisting on a trajectory planner which computes a high-level plan for the incoming overtaking, the proposed Informative Model Predictive Scheme (I-MPS) which is in charge of tracking the plan provided by trajectory planner, the sensory system which is responsible of providing relevant information to the on-board systems regarding the vehicle's velocity, position, acceleration and heading and an EKF which is in charge of filtering out the measurement noise from the measurements coming from the upstream sensory system. We further assume: (i) no model mismatch between the model used by the I-MPS and the mismatch for the vehicle's dynamics and (ii) that a hacker can inject Replay Attack (RA) or Denial of Service (DoS) attacks by accessing the vehicle's CAN gateway (e.g., through Bluetooth, radio data, telematics) [71].

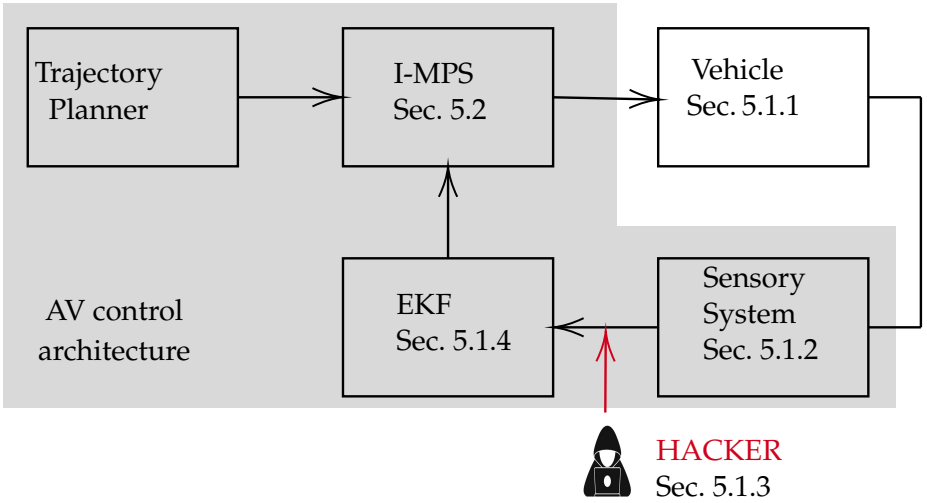


Figure 5.1: AV control architecture. The hacker corrupts the outputs of the sensory system.

5.1.1 Vehicle Model

Vehicles operating in an urban environment can be modeled via a kinematic bicycle based model [70], [72]. In such model, the wheels do not slip at their contact point with the ground since both the velocity and the acceleration values are assumed to be low. Furthermore, we assume that the vehicle is a front wheel steering vehicle.

Thus, the vehicle dynamics are given by [70]

$$\dot{x} = v\cos\left(\psi + \beta\right), \tag{5.1a}$$
$$\dot{y} = v\sin\left(\psi + \beta\right), \tag{5.1b}$$
$$\dot{\psi} = \frac{v}{l_r}\sin\beta, \tag{5.1c}$$
$$\dot{v} = a, \tag{5.1d}$$

where x and y are the coordinates of the vehicle mass-center in an inertial frame, ψ is the heading angle of the vehicle mass-center, v is the velocity of the vehicle mass-center, a is the mass-center acceleration, $\beta = \arctan\left(l_r \tan\delta_f / (l_f + l_r)\right)$ is the side slip angle of the vehicle, and l_f and l_r are the distances of the front and rear axles from the mass-center, respectively, δ_f and a the steering angle and acceleration, respectively.

In what follows, $\zeta = \begin{bmatrix} x & y & \psi & v \end{bmatrix}^\top \in \mathbb{R}^4$ and $u = \begin{bmatrix} \delta_f & a \end{bmatrix}^\top \in \mathbb{R}^2$ will be used to indicate the state and the input vectors of the system, respectively.

5.1.2 Sensory System

The state measurements supplied by the Sensory System module are subject to additive noise and are modeled as

$$\zeta_m = \zeta + \eta, \tag{5.2}$$

where $\zeta_m = \begin{bmatrix} x_m & y_m & \psi_m & v_m \end{bmatrix}^\top$ and $\eta = \begin{bmatrix} \eta_x & \eta_y & \eta_\psi & \eta_v \end{bmatrix}^\top$, with η_x, η_y, η_ψ and η_v being zero mean, white Gaussian noises. The state measurements vector ζ_m is the whole knowledge accessible by the hacker.

5.1.3 Hacker Model

In this subsection we define how a hacker can corrupt the measurements, as depicted in Fig. 5.1. For both attacks, t_a and T_d will indicate the time at which an attack starts and its duration, respectively.

5.1.3.0.1 Replay Attacks RAs consist of a recording and a replay phase. During the recording phase, the hacker stores the past measurements, which in the replay

phase are injected into the system. The RA model is given by [73]

$$\zeta_m^{\mathrm{RA}}(t) = \begin{cases} \zeta_m(t - T_d), & \text{if } t \in \left[t_a, t_a + T_d\right], \\ \zeta_m(t), & \text{otherwise.} \end{cases} \tag{5.3}$$

with the recording taking place in the time interval $\left[t_a - T_d, t_a\right]$.

5.1.3.0.2 Denial of Service Attacks In this case, the hacker interrupts the data stream between the Sensory System and the EKF. In real vehicles the Electronic Control Unit (ECU), where the Controller block is implemented, typically holds the last useful data in the local memory which is then used in place of the expected new ones when they are not available. The model for DoS attacks is [74]

$$\zeta_m^{\mathrm{DoS}}(t) = \begin{cases} \zeta_m(t)\big|_{t=t_a}, & \text{if } t \in \left[t_a, t_a + T_d\right], \\ \zeta_m(t), & \text{otherwise.} \end{cases} \tag{5.4}$$

That is, the effect of the DoS attack consists in resending the sensor measured at the time $t = t_a$ during the time interval $\left[t_a, t_a + T_d\right]$.

5.1.4 EKF

In this work, the EKF [75] provides the estimates $\hat{\zeta}$ of the vehicle's dynamics to the I-MPS block. Although the use of this estimator could improve the detectability of a cyber-attack [63], in the case under investigation, it is in charge only to provide the state estimation to the Nonlinear Model Predictive Control (NMPC).

5.2 I-MPS

Fig. 5.2 shows the proposed I-MPS. It includes three interconnected modules: the Detector, the Mitigator and an NMPC. However, the architecture and the deriving scheme are general enough so as to allow other types of Model Based Controller such as, e.g., LTI-MPC, Linear-Time-Varying Model Preditive Controller (LTV-MPC), Linear-Parameter-Varying Model Preditive Controller (LPV-MPC). A discrete-time setting is assumed with k being the discrete-time variable and T_s being the sampling time such that $t = kT_s$.

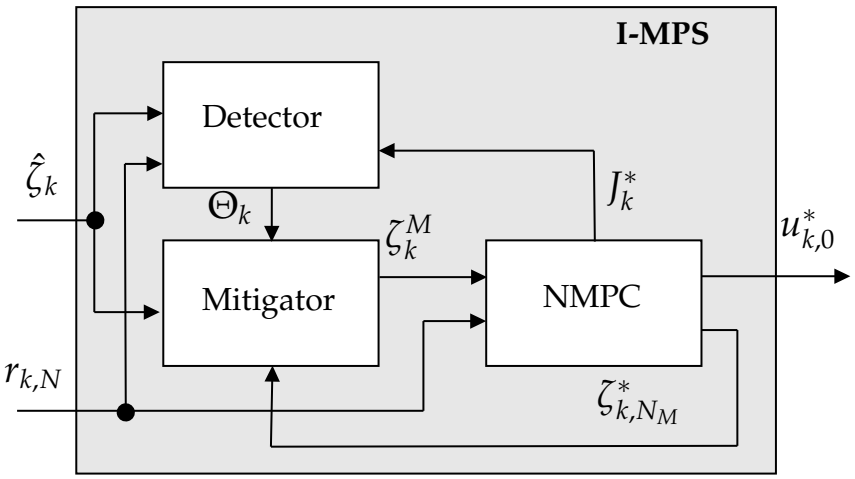


Figure 5.2: I-MPS architecture.

5.2.1 NMPC for Trajectory Tracking

The NMPC is solved at each k by minimizing the cost function

$$J_k = \sum_{i=k}^{k+N} (\zeta_i - r_i)^\top L(\zeta_i - r_i) + \sum_{i=k}^{k+N-1} (u_i - u_{i-1})^\top S(u_i - u_{i-1}) + u_i^\top R u_i, \tag{5.5}$$

across the prediction horizon N and where $\zeta_i = \begin{bmatrix} x_i & y_i & \psi_i & v_i \end{bmatrix}^\top$ is the i-th future state vector of the vehicle's dynamic model used by the NMPC, r_i is the i-th future reference provided by the trajectory planner through the sequence $r_{k,N} = \{r_k, \ldots, r_{k+N}\}$, u_i is the i-th NMPC future outputs, and L, S and R are suitable weighting matrices of appropriate dimensions by which the control input performances are taken into account.

The discrete-time optimization problem is solved by discretizing the continuous time model (5.1) assuming a piecewise constant control $u(\tau) = u_k$, with $\tau \in [t_k, t_{k+1}[$, and then using the multiple shooting method [76] which yields the constraint

$$\zeta_{k+1} - \Xi(\zeta_k, u_k) = 0, \tag{5.6}$$

where $\Xi(\zeta_k, u_k)$ is the 4-th order Runge-Kutta integrator for the integration of the system dynamics over the shooting interval. Furthermore, the initial state is

included as constraint into the optimization problem as follow

$$\zeta_k - \zeta_k^M = 0, \tag{5.7}$$

where ζ_k^M is the Mitigator output.

The physical limitations of the control input regarding the minimum and maximum values, $u_{\min}$ and $u_{\max}$, respectively, and slew-rates, $\dot{u}_{\min}$ and $\dot{u}_{\max}$, respectively, that the actuation system is capable of, can be taken into account as

$$u_{\min} \leq u_k \leq u_{\max}, \tag{5.8a}$$
$$\dot{u}_{\min} \leq \dot{u}_k \leq \dot{u}_{\max}. \tag{5.8b}$$

Ultimately, the NMPC computes

$$\begin{aligned} \zeta_{k,N}^*, u_{k,N-1}^* = & \underset{\zeta_{k,N}, u_{k,N-1}}{\arg\min} && (5.5) \\ & \text{s.t.} && (5.6) - (5.8), \end{aligned} \tag{5.9}$$

where $\zeta_{k,N} = \{\zeta_k, \ldots, \zeta_{k+N}\}$ and $u_{k,N-1} = \{u_k, \ldots, u_{k+N-1}\}$.

5.2.2 Mitigator

The Mitigator is triggered depending to the flag $\Theta_k \in \{0, 1\}$ (see Fig. 5.2): if $\Theta_k = 0$ no cyber-attack has been detected and the state estimate $\hat{\zeta}_k$ can be fed to the NMPC at the current iteration k; while, if $\Theta_k = 1$ a cyber-attack has been detected and a countermeasure is taken. Thus, the Mitigator implements

$$\zeta_k^M = \begin{cases} \hat{\zeta}_k & \text{if } \Theta_k = 0, \\ \bar{\zeta}_k & \text{if } \Theta_k = 1, \end{cases} \tag{5.10}$$

where $\hat{\zeta}_k$ is the state estimate at time-instant k and

$$\bar{\zeta}_k = \sum_{i=k-N_M}^{k-1} \frac{\zeta_{i,k}^*}{N_M}, \tag{5.11}$$

where $\zeta_{i,k}^*$ is the optimal state that was computed by the $(k-i)$-th NMPC iteration and considered at current time k, $N_M \leq N$ is the number of past samples used by

the Mitigator at current time k and $\bar{\zeta} = \begin{bmatrix} \bar{x} & \bar{y} & \bar{\psi} & \bar{v} \end{bmatrix}^\top$.

5.2.3 Detector

The Detector is a Cubic Support Vector Machine (CSVM) classifier trained against RA and DoS attacks. In particular, a multiclass classification problem is considered, with classes reported in Table 5.1.

Class	Cyber-attack	Number of instances
0	No attacks	14765
1	$\{x_m,\ y_m\}_{\text{DoS}}$	3135
2	$\{\psi_m, v_m\}_{\text{DoS}}$	3154
3	$\{x_m,\ y_m\}_{\text{DoS}} \cup \{\psi_m, v_m\}_{\text{DoS}}$	3126
4	$\{x_m,\ y_m\}_{\text{RA}}$	1812
5	$\{\psi_m, v_m\}_{\text{RA}}$	1708
6	$\{x_m,\ y_m\}_{\text{RA}} \cup \{\psi_m, v_m\}_{\text{RA}}$	1830

Table 5.1: Class definitions.

The class 0 identifies the case in which no cyber-attacks occur. The classes 1–3 and 4–6 regard RA and DoS attack onto either part or the entire attack space. In particular, the hacker is assumed to be able to access the set $\{x_m,\ y_m\}$ of the position measurements, the set $\{\psi_m,\ v_m\}$ of the heading and velocity measurements, or the set $\{x_m,\ y_m\} \cup \{\psi_m,\ v_m\}$. The particular choice of the classes has derived by the assumption that hacker affects the ECUs of the GPS and/or pose sensors. At each k, three features are used for detection: the first is the error

$$e_{\hat{\zeta},k} = \hat{\zeta}_k - r_k, \tag{5.12}$$

between the state estimates from the EKF and the references provided by the trajectory planner, the second is the NMPC cost function J_k in (5.5), the third is the vector

$$D_k^{\text{KL}} = \begin{bmatrix} D_k^{\text{KL}}(\bar{E}_x \| Q_{x,k}) \\ D_k^{\text{KL}}(\bar{E}_y \| Q_{y,k}) \\ D_k^{\text{KL}}(\bar{E}_\psi \| Q_{\psi,k}) \\ D_k^{\text{KL}}(\bar{E}_v \| Q_{v,k}) \end{bmatrix}, \tag{5.13}$$

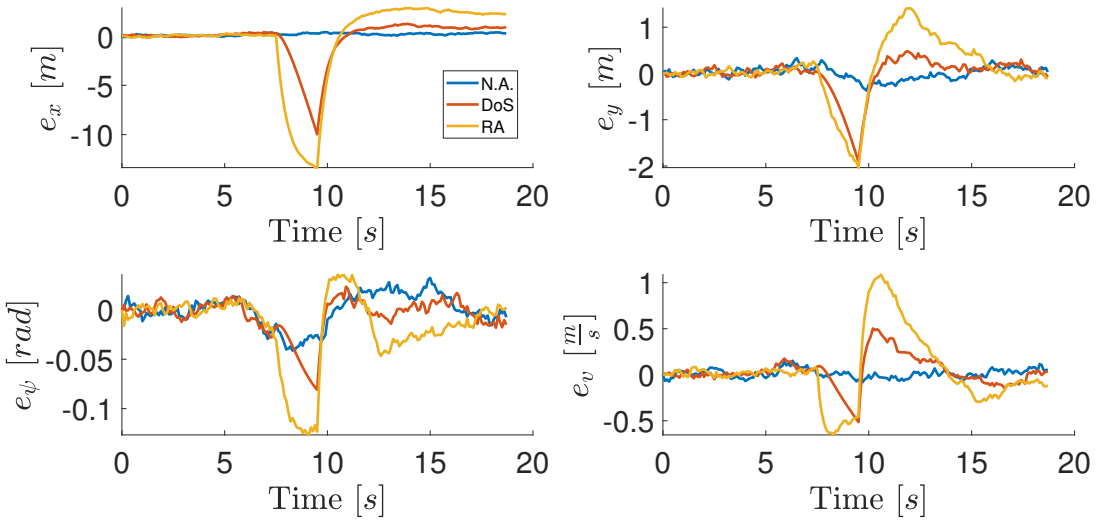


Figure 5.3: Errors e_ζ. Top-left: e_x. Top-right: e_y. Bottom-left: e_ψ. Bottom-right: e_v.

of the Kullback-Leibler (KL)[1] divergences between the estimated probability densities of $\bar{E}_x$, $\bar{E}_y$, $\bar{E}_\psi$ and $\bar{E}_v$ of $\bar{e}_x = \bar{x} - \hat{x}$, $\bar{e}_y = \bar{y} - \hat{y}$, $e_\psi = \bar{\psi} - \hat{\psi}$ and $\bar{e}_v = \bar{v} - \hat{v}$, respectively, under known conditions when no cyber-attack is taking place and the corresponding $Q_{x,k}$, $Q_{y,k}$, $Q_{\psi,k}$ and $Q_{v,k}$ estimated during the driving during the time window $\{k - N_M, \dots, k-1\}$ at the current time k. Here, the errors $\bar{e}_x$, $\bar{e}_y$, $\bar{e}_\psi$ and $\bar{e}_v$ are calculated the difference between $\bar{\zeta}$ and $\hat{\zeta}$ which represent the predicted state from the past and the state estimate, respectively.

5.2.3.1 Classifier Training

In order to build the attack dataset for the training of the detection model, for each class $N_{\text{sim}} = 10000$ simulations for the overtaking scenario have been carried out. Regarding the class 0, no attacks is injected all along the overtaking maneuver, whereas, regarding the other classes, the corresponding attack is injected. At the beginning of each attack, t_a and T_d are computed as realization of uniformly distributed random variables with supports $\left[t_a^-, t_a^+\right] \subset \mathbb{R}^+$ and $\left[T_d^-, T_d^+\right] \subset \mathbb{R}^+$, respectively. The features (5.5), (5.12) and (5.13) of the classes 1-6 are collected during the attack phase $t \in \left[t_a, t_a + T_d\right]$, while for the class 0 the same features

[1]The KL divergence between two probability densities $p(x)$ and $q(x)$ is defined as

$$D_{\text{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx.$$

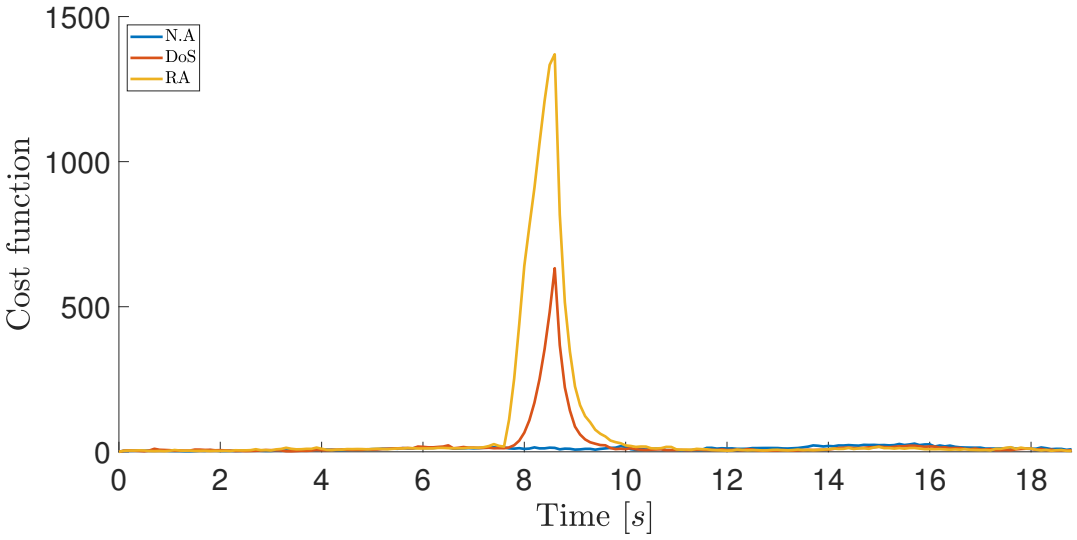


Figure 5.4: Functional cost J^*.

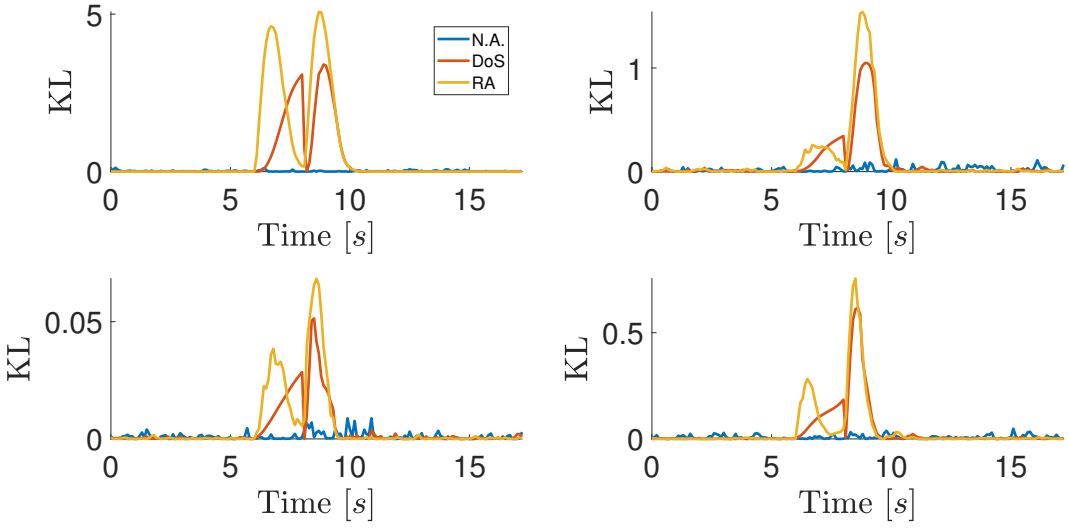


Figure 5.5: KL Divergences D^{KL}. Top-left: $D^{\mathrm{KL}}(\bar{E}_x \| Q_x)$. Top-right: $D^{\mathrm{KL}}(\bar{E}_y \| Q_y)$. Bottom-left: $D^{\mathrm{KL}}(\bar{E}_\psi \| Q_\psi)$. Bottom-right: $D^{\mathrm{KL}}(\bar{E}_v \| Q_v)$.

are collected during the whole time of the simulation.

The number of instances of each class during the training is reported in Table 5.1. In particular, the dataset is built such that in half of the instances there is an attack while no attacks are taking place in the rest.

The features e_ζ, J and D^{KL} during the training are reported respectively in Fig. 5.3–5.5, where the Yellow lines refer to RA, the Orange lines refer to DoS attacks and the Blue lines refer to no attacks (N.A.).

The confusion matrix of the CSVM Classifier is reported in Fig. 5.6, and Precision, Recall and F1 Score are reported in Table 5.2. In order to understand

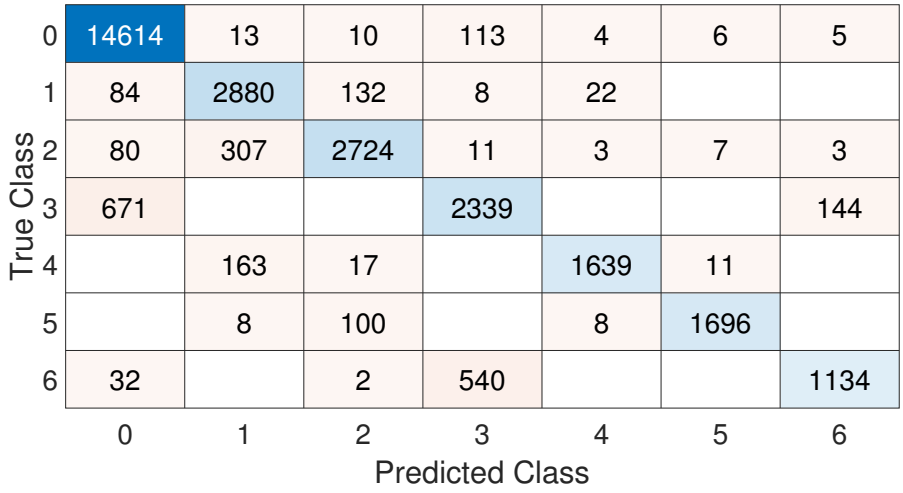


Figure 5.6: CSVM Confusion Matrix.

the effectiveness of the proposed approach we carried out a confrontation with the RA Detector proposed in [77] whose Precision, Recall and F1-Score result to be 81.6%, 35.1% and 49.0%, respectively. While the proposed approach achieves Precision of 98.9%, Recall of 94.4% and F1-Score of 96.6%. As it is possible to see, the obtained results show that the predictive features of the NMPC could obtain satisfactory Classifier metrics. Regarding the metrics shown in Table 5.2, we remark that classes 2 is less detectable compared to class 3 and class 5 is less detectable respect to class 6, due to the different impact that the corresponding type of attack has on the performances of the control architecture.

5.3 Simulations and results

In this section we present the simulation outcomes of the proposed scheme in comparison with those of a AV architecture where no mitigation is considered.

Classes	Precision	Recall	F1 Score
0	94.4 %	99.0 %	0.96
1	85.4 %	92.1 %	0.88
2	77.7 %	74.2 %	0.75
3	91.3 %	86.9 %	0.89
4	97.8 %	89.6 %	0.93
5	88.2 %	66.4 %	0.75
6	98.6 %	93.6 %	0.96

Table 5.2: Cubic SVM Classifier.

The software is implemented in MATLAB® and CasADi [78], and the simulation parameters are reported in Table 5.3. In Fig. 5.7-5.9, the simulations regarding

Parameters	**Value**
l_f	$1.738\,\mathrm{m}$
l_r	$1.105\,\mathrm{m}$
T_s	$0.1\,\mathrm{s}$
S	$\mathrm{diag}(100, 1000)$
R	$\mathrm{diag}(0.1, 0.1)$
L	$\mathrm{diag}(10, 10, 400, 200)$
$\dot{u}_{\max}$	$\begin{bmatrix} 0.6\,\mathrm{m/s^3} & 0.2\,\mathrm{rad/s} \end{bmatrix}^\top$
$\dot{u}_{\min}$	$\begin{bmatrix} -0.6\,\mathrm{m/s^3} & -0.2\,\mathrm{rad/s} \end{bmatrix}^\top$
$u_{\max}$	$\begin{bmatrix} 2.5\,\mathrm{m/s^2} & 0.7\,\mathrm{rad} \end{bmatrix}^\top$
$u_{\min}$	$\begin{bmatrix} -2.5\,\mathrm{m/s^2} & -0.7\,\mathrm{rad} \end{bmatrix}^\top$
H_p	30
N_M	15

Table 5.3: Parameter values.

the overtaking maneuver are reported in case no mitigation countermeasure is implemented. Cyan lines refer to the trajectory references, orange lines refer to the state measurements and Yellow lines refer to the state estimates. Only attacks on the entire attack space have been considered with $t_a = 9.5\,\mathrm{s}$ and $T_d = 2\,\mathrm{s}$ for RAs and $t_a = 7.5\,\mathrm{s}$ and $T_d = 2\,\mathrm{s}$ for DoS attacks. For different values of $t_a \in \left[t_a^-, t_a^+\right]$ the Detector behaviour is the same.

In Figs. 5.8, 5.9 it is possible to note that the error between the measured/actual x and y and the reference in RAs is greater than in DoS attacks, thus implying a higher control degradation. This happens because in DoS attacks only the state in the last sampling interval is available while in RAs a replayed state sequence is available that is related to a time window spanning several

samples back in time. Reintroduce these past state estimate sequence to the NMPC produces a considerable error in the calculation of the control inputs; in particular, the greater the replay time, the more evident the degradation of performance is.

The effectiveness of the I-MPS architecture is shown in Fig. 5.10 where the heading angle, the velocity and the position are reported when an RA with $t_a = 10\,\text{s}$ and $T_d = 2\,\text{s}$. As it is possible to notice, the implemented I-MPS is able to detect in time the cyber-attack and the mitigation action allows the AV to conduct an overtaking maneuver safely. Indeed, the safety of the maneuver is obtained by minimizing the position error with respect to the trajectory planned by the planner. Furthermore, the time to detect the attacks is of two time steps.

The effectiveness of proposed architecture is also highlighted in Fig. 5.11 which reports the trajectories in the $e_x - e_y$ plane for an AV during overtaking when both the proposed scheme is implemented and the proposed scheme is not implemented. An ellipsoidal safe region for the overtaking maneuver is also plotted (solid red line) and the trajectory position errors are reported for the aforementioned simulations. The safe region is computed as the maximum deviation with respect to the trajectory reference which allows the AV to perform the overtaking maneuver without damaging or hitting possible obstacles.

Cyan, Magenta, Green and Blue represent the simulations regarding No Attack, RA with I-MPS, DoS attack and RA without I-MPS, respectively. The novel I-MPS implemented allows to reduce the effect of cyber-attacks generated by a hacker. In cases of no detection and mitigation of RA and DoS attack, the errors of the trajectory result out of the safety region, which could cause damages and possible injuries to the driver.

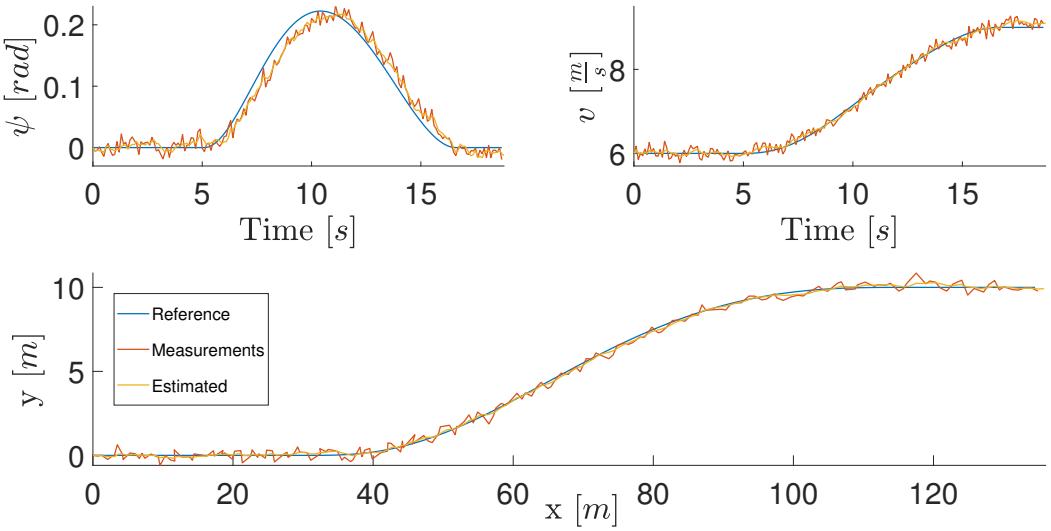


Figure 5.7: No attacks.

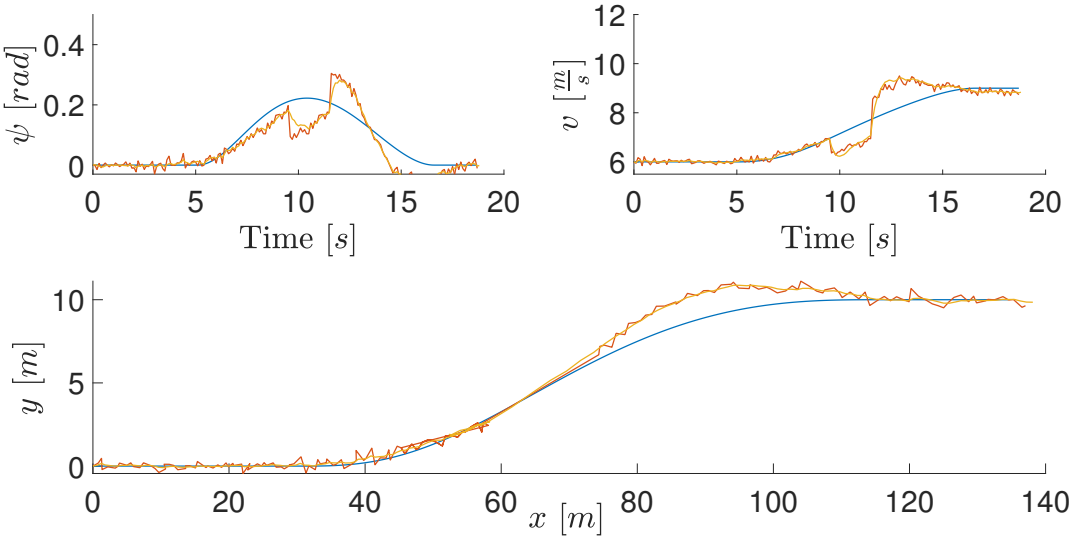


Figure 5.8: RA.

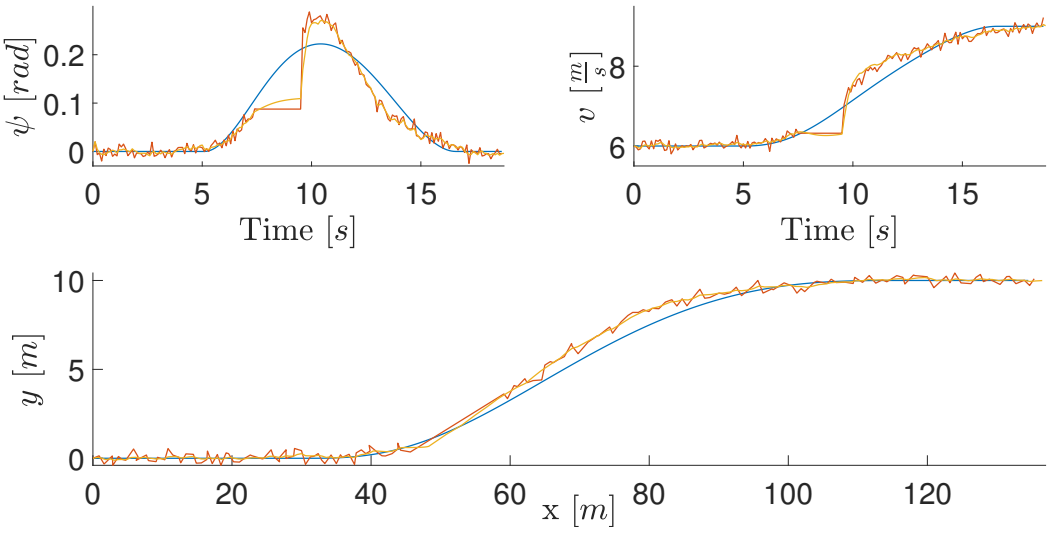


Figure 5.9: DoS attack.

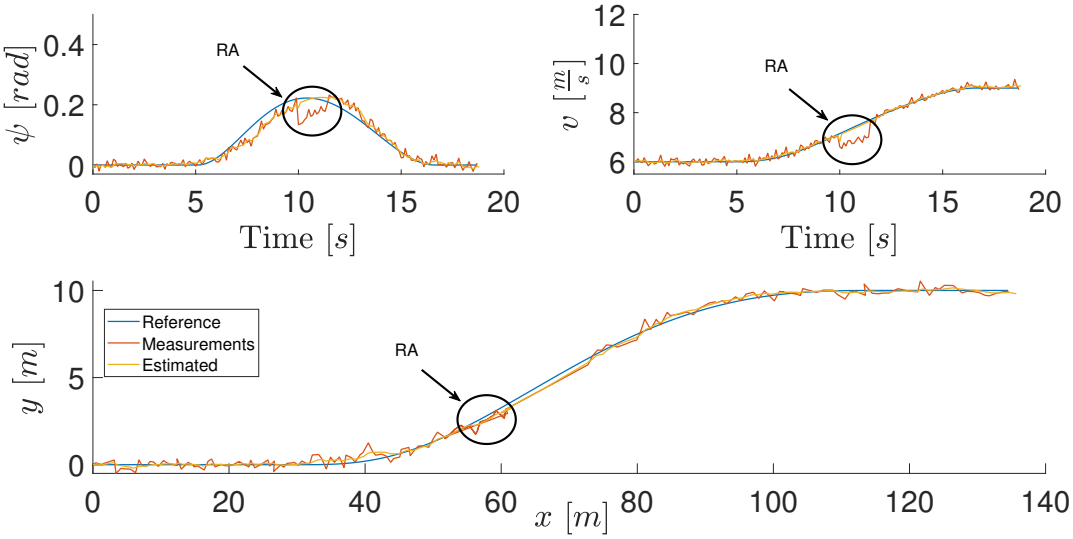


Figure 5.10: RA mitigation using I-MPS. Start time attack and duration are $t_a = 10\,\text{s}$ and $T_d = 2\,\text{s}$, respectively. Top-left: Heading angle. Top-right: Velocity. Bottom: Position.

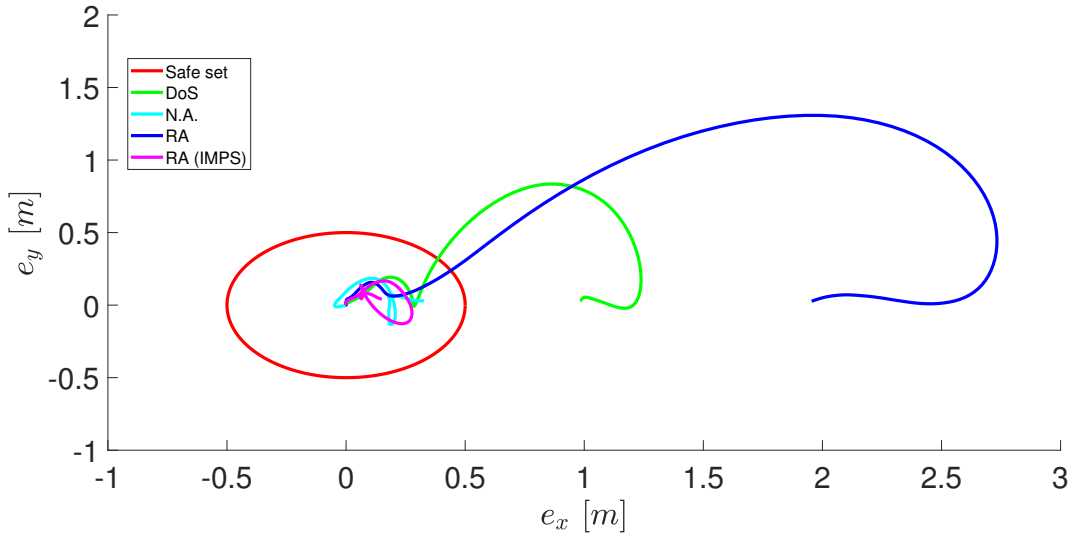


Figure 5.11: Safe region and error trajectories.

5.4 Chapter Summary

The conclusion of this chapter highlights the Informative Model Predictive Scheme algorithm's successful application in enhancing cybersecurity for autonomous vehicle systems, particularly during overtaking maneuvers vulnerable to cyber-attacks such as Replay Attacks and Denial of Service. Numerical results from simulations validate the efficacy of the I-MPS, demonstrating its potential in accurately detecting and mitigating cyber threats in real-time. These findings suggest a significant step forward in vehicular cybersecurity, illustrating the innovative synergy between machine learning algorithms and control systems in combatting cyberattacks. The results also indicate areas for further accuracy enhancement, reinforcing the I-MPS as a pioneering approach in the ongoing evolution of cybersecurity measures for autonomous vehicles.

Part III

COOPERATION

Literature Review

A fair amount of work exists on the topic of VC and in particular regarding the control techniques developed [79]. The main technologies mentioned include: consensus-based control [80], constraint following control [81, 82], sliding mode control [83, 84], MPC, and Machine Learning (ML)-based control. Emphasis will be given to the last two types, as they prove to be the most promising in this field; specifically, the latter is the type upon which the control system presented in this work is based.

The integration of ML into control algorithms has gained significant attention in the context of railway systems, particularly concerning virtual coupling [85]. In [86], the use of RL is proposed to obtain an optimal policy for IoT-based Virtually Coupled Train Sets (VCTS). The proposed approach combines RL and artificial potential field to achieve global optimal policy and increase efficiency. Simulation results demonstrate the effectiveness of the proposed RL-based cooperative control approach for IoT-based VCTS. The control strategy presented in [87] shows a decentralized VC using a RL approach based on the DDPG algorithm, with a focus on robustness and safety. The robustness is evaluated solely through Monte Carlo simulations, considering parameter uncertainties of the dynamical system. However, the approach does not provide formal analytical guarantees of robustness or safety. Additionally, the strategy does not account for communication impairments from communication channel, including time-varying delays in Vehicle-to-Vehicle (V2V) communication, packet losses, or switching topologies. While Monte Carlo simulations provide empirical insights, formal guarantees remain absent for these challenging scenarios. The bottleneck of this ML-based control type lies in the foundations of its theory. Currently, in the railway domain, it is not yet possible to certify an ML-based controller according to railway safety standards, therefore making such approaches not realisic.

The approaches most closely related to our solution are the works in [88] and [89]. In [88], a novel train control system utilizing virtual coupling is introduced. This system employs a decentralized model predictive control framework to optimize the control of both leading and following trains in a convoy. Comparative analyses, particularly against the moving block system, reveal improved performance, demonstrating the virtual coupling's efficacy in reducing headway and ensuring safe train separation. A linear MPC

optimizing goals like track spacing, velocity, and comfort is implemented in [89]. Constraints, including line velocity, collision avoidance, and traction/braking, are considered. In this work, although safety constraints are considered inside the controller, no certification of safety is demonstrated. Compared with these two contributions, our control scheme takes into account the intrinsic characteristics of the communication channel and assumes a heterogeneous platoon with parameters uncertainty. In both [88] and [89], the authors propose decentralized linear MPCs to realize the VC and assume the two trains can communicate in a reliable sampled framework. In our framework we also consider (nonlinear) MPCs but we do not focus on this aspect in details. Rather, in this work we propose a hybrid/switching control architecture able to robustly satisfy safety constraints and reducing unnecessary follower train brakings, also considering not reliable communication between trains. Our framework is able to cope with diverse (MPC based) controller choices, both for the leader and the follower.

Chapter 6

A Safe and Robust Control System Architecture for Virtual Coupling

The continuous increase in demand for railway transportation is pressing for finding new solutions to increase capacity accordingly. Nowadays, the capacity of high-speed railway lines is saturated, and solutions that allow for an increase are being studied. In this context, the implementation of technology such as VC is emerging as a promising prospect to address these challenges and improve the overall operational efficiency of the railway system. In this work, we introduce a control system architecture enabling the transition from European Railway Traffic Management System Level 3 to VC and managing VC operation. The proposed architecture addresses parameter uncertainty within train models and the variability in data communication, which is often unreliable and affected by delays and packet drops. We establish a robust control framework for safety and reliability despite these challenges. We incorporate a safety control barrier function to certify safety guarantees and prioritize operational safety. The architecture is designed with the possibility of implementing various control laws, safety rules and estimators block still guaranteeing performances. To validate our work, a railway simulation tool for VC scenarios is developed, in collaboration with RFI S.p.A, the Italian railway company, demonstrating its effectiveness and foundational role in advancing railway safety.

6.1 Introduction

The railway network is a complex and interconnected system that comprises a multitude of critical components, such as track circuits, switches, signals, and many others. All these elements work together to ensure the safe and efficient operation of the entire network. At the core of a railway infrastructure there are two systems: the Interlocking System (IS) and Radio Block Centre (RBC).

The IS plays a crucial role in guaranteeing the efficient and safe operation of the railway network by continuously overseeing and managing the status of all its elements. It ensures their proper functioning and communication. Through real-time monitoring of track circuits, switches, and various railway components, the IS has the capability to issue commands to signal switches and other essential elements, thereby maintaining optimal railway functionality.

The second important element of the railway network is the RBC which serves as the interface between the IS and the on-board trains. In Europe, it is based on European Railway Traffic Management System (ERTMS) a standardized and interoperable train control and command system for railways [90]. In its pursuit of ensuring the secure spacing of trains, the RBC leverages real-time insights into the current state of the railway network. This comprehensive understanding is derived from the data received both from the IS and the dynamic information regarding the positions and velocities of actively circulating trains. The goal of the RBC is to manage and optimize the safe distances between trains, thereby contributing to the overall safety and efficiency of railway operations. In this work, the RBC will be considered the actor which is in charge to initiate a VC operation between trains.

The current communication technology present on the railway infrastructure is Vehicle-to-Infrastructure (V2I), namely, the communication between the RBC and trains. The existing standard V2I is an integral component of the railway communication infrastructure, relying on the GSM-R network. Nowadays, the emergence of the 5G network-based railway communication technology, known as Future Railway Communication and Management System[91], is a viable V2V technology. It is, indeed, gradually gaining prominence due to its advanced capabilities in data transfer, reliability and low-latency characteristics, ensuring swift and real-time data exchange among trains within the network.

The growing popularity of trains as a mode of transportation presents a challenge of overburdening the railway system's capacity. As a result, enhancing

infrastructure utilization and increasing capacity are two crucial objectives that railways are currently striving to achieve. These goals are being tackled by Shift2Rail [92], an initiative focused on developing innovative technologies and solutions to enhance the efficiency, safety, and sustainability of the railway system.

In the following, we will introduce two ERTMS technology standards, upon which the work is based: ERTMS Level 3 (L3) and VC which are depicted in Figure 6.1.

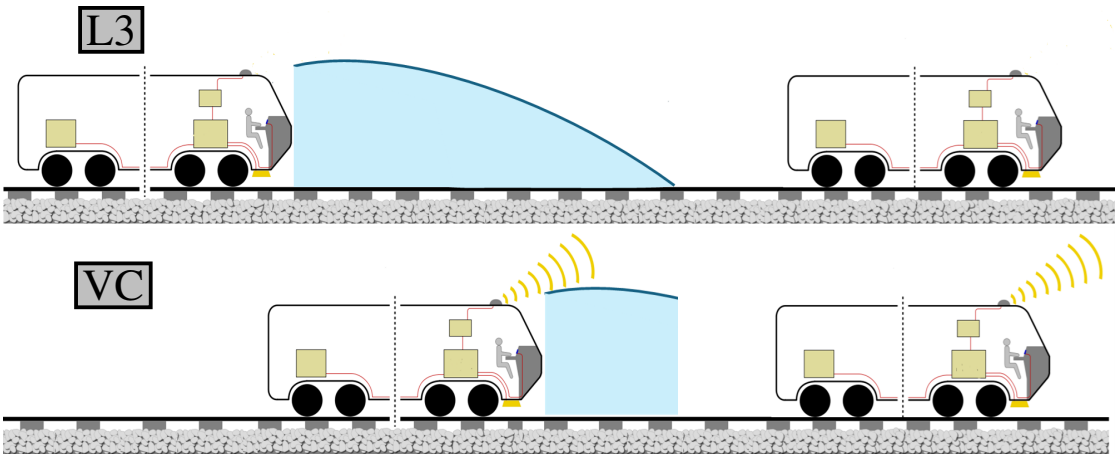


Figure 6.1: Comparison scenarios between L3 and VC. The blue area beneath the curve represents the absolute braking distance required to stop the train.

By transitioning from fixed to virtual block systems, L3 reduces dependence on trackside signals and infrastructure, relying instead on advanced onboard systems and continuous communication. This shift not only enhances line capacity but also reduces operational costs by optimizing train headways and minimizing the need for trackside maintenance; furthermore, it allows trains to move dynamically within a virtual continuously updated block which is contingent upon the position of the preceding train [93].

Nowadays, the technology for VC is still in the conception stage, and research is underway to determine the best way to implement it [94]. The control field is particularly interested in finding a solution that can provide the necessary velocity, reliability, and safety to support real-time communication between trains. In addition to these technical considerations, research in the control field is also underway to determine the best way to implement VC from an operational perspective [95]. This involves studying the impact of the new technology on

train operations and identifying the best practices for integrating the technology into the existing railway network [96].

In conclusion, VC presents a notable advantage in reducing railway delays, particularly in departure times. This innovative system holds the potential to decrease departure delays by optimizing train coordination and spacing. Promoting a more synchronized and efficient operation, VC emerges as a valuable solution to increase the capacity and enhance punctuality across the railway network.

6.1.1 Contribution

In this work, we introduce a control system architecture allowing the transition from L3 to VC and manage VC operations.

Specifically, the VC is a condition under which two trains are spaced not less than a safety distance d and, in principle, try to be spaced as close as possible to d. However, in the solution we propose the distance between the two trains may vary during the trains journey according to the operating conditions (but always keeping the prescribed minimum safety distance d).

The contributions of our work is articulated through the following key points:

1. Our architecture addresses the twofold challenge of parameter uncertainty within train system models and the throughput variability in data packet communication channel characteristics. We establish a robust control system framework that is specifically engineered to accommodate these variabilities, always ensuring safety and reliable performance.

2. For ensuring safety, we incorporate into the design of our control architecture a safety control barrier function. This allows to certify safety guarantees and prioritize operational safety.

3. The control architecture allows to implement very general control laws (we provide a switched controller but other choices are possible), guaranteeing not only safety, but also avoiding as much as possible follower emergency braking when no information are received from the leader.

4. To validate the applicability and safety of our control system, we have developed a specialized railway simulation tool for VC. This tool enables testing and evaluation of the control system across various scenarios. The demonstration of the system's effectiveness in these tests not only proves

its operational viability but also solidifies its foundational role in advancing railway safety.

The work is organized as follows: Section 6.2 provides an initial overview of the problem and the proposed control system architecture, the principal blocks implemented are reported. Section 6.3 introduces safe sets and control barrier functions, ensuring system safety through robust and safety-compliant controllers, which maintain the system within safe operational conditions despite uncertainties. Section 6.4 models the train's longitudinal dynamics defining the uncertain system parameters and ensuring safety with a robustification technique. Section 6.5 introduces braking and emergency controller definitions, used to ensure robust safety compliance for the train system. Section 6.6 provides a detailed description of the control system architecture for trains operating under VC. It thoroughly examines the primary components and their interactions, offering insights into the design and functionality of each block within the system. This section highlights the innovative approaches used to manage the complexities of VC, ensuring efficient and safe train operations. Section 6.7 presents the different operational scenario simulations and results. Finally, Section 6.8 includes the conclusions and future works.

6.2 Addressed problem and proposed control architecture

In this section we provide a preliminary description of the problem addressed in the work and the proposed control architecture. Specifically, we consider asynchronous communication between the leader and the follower over an unreliable communication channel. Leader sends packets about its planned trajectory (more precisely it will be a robust information about the planned trajectory). Such packets might be received with delay (or not received at all) by the follower.

The problem to be addressed is the one of controlling the trains according to a "virtual coupling" (VC) scheme. Specifically, the follower train should move in a coordinated way with the leader, keeping a safety distance of at least d (which is a provided control parameter). The aim of the VC is to reduce the spacing between the trains on the railway and so increase its capacity. Often, when referring to VC in the literature, the distance between the trains is imposed to be exactly d.

In our control architecture, instead, the inter-trains distance might vary (but never being below d) according to the operating condition. Implicitly, the control

architecture will control the inter-train distance guaranteeing safety and avoiding unnecessary train braking. This will be achieved through several control blocks simultaneously working. All these blocks run on the follower side.

Specifically:

- The Leader robust lower proxy (RLP) predictor block is responsible of predicting the evolution of a system (namely the RLP) that provides a lower bound of the true leader trajectory subjected to an hypothetical emergency braking (that is, the maximum allowed deceleration). This block will be reset at follower packets receptions with the updated state of the leader.

- The Safety control block is fed with the output of the leader RLP predictor and the actual state of the follower. It continuously monitor for the overall system safety via a suitable control barrier function. Such monitoring runs on predicted information even if no updates are received from the leader. If conditions are detected such that safety might be a risk, an emergency braking is imposed to the follower, overlapping any other possible control. The emergency braking is kept until safety conditions are restored (including new leader updates). The safety control block guarantees safety in a certified manner and robustly with respect to follower and leader parameters uncertainties and communication network unreliability.

- The Cruise virtual coupling control block manages the follower during the normal cruise (i.e., when no safety issues are raised). It implements a predictive controller over a receding horizon while keeping the position of the follower at a certain (time-varying) distance from the leader. This distance is such that, if no updates are received, the follower can still move with its planned velocity for a given tunable amount of time before possibly trigger an emergency braking. This feature allows to implicitly regulate the inter-train distance according to the characteristics of the communication channel, enlarging or squeezing this dwell time in presence of less or more reliable communication (respectively).

- The Delay estimator block continuously calculates the communication delay between the leader and follower by analyzing packet arrival times. This real-time estimation allows the control system to adjust reference trajectories based on current delay conditions, ensuring smooth coordination despite

channel variability. The block interacts with the Cruise virtual coupling block to maintain safety and minimize the impact of communication delays on overall performance.

It is worth mentioning that the proposed architecture does not prescribe a specific controller. Rather, it provides an environment enabling several possible controller choices, leaving the specific design to the developer. Therefore, while in this work we propose a switching controller over three possible operational modes (with more or less aggressive maneuvers according to the inter-train distance), several other choices are possible. Also, our framework seamless allows the implementation of any additional safety criteria or the integration of other modules/features, such as the online estimation of the channel quality and characteristic delay, so as to adjust in real-time the dwell time in the Cruise virtual coupling control block. Such block is presented in this work but not strictly necessary for safety and stability.

6.3 Background

6.3.1 Dynamical system and Barrier Function

Here we introduce the concept of safe set and barrier function. We keep our illustration rather informal and we only provide those concepts which will be useful in the rest of the work. For this reason we do not always follow a "standard" notation and cast this background for the specific problem in hand. Further details can be found in several papers and books in the literature, see for example [97] and references therein.

Let us consider a dynamical system described by

$$\dot{x}(t) = f(x(t), u(t)), \tag{6.1a}$$

$$\mathcal{C}(x(t), u(t)) \leq 0 \tag{6.1b}$$

$$x_0 = x(t_0). \tag{6.1c}$$

where $t \in \mathbb{R}_+$ is the time, $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}^m$ the control input, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ the (smooth) dynamical function and $\mathcal{C} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^l$ provides static constraints.

Let us consider a safe set $\mathcal{S} \subseteq \mathbb{R}^n$, that is a set of the state space where we aim to confine the state of the system.

Definition 1 (CBF). *Let us consider a dynamical system* (6.1) *and a safe set* $\mathcal{S} \subseteq \mathbb{R}^n$. *A control barrier function on* $\mathcal{S}$ *is any continuous function* $b(x) : \mathbb{R}^n \to \mathbb{R}$ *such that*

$$b(x) \leq 0 \iff x \in \mathcal{S}. \tag{6.2}$$

Notice that some authors in the literature reverse the sign of the inequality in the above definition; this, obviously, is only a convention that does not alter its meaning. Also, in the literature a further hypothesis on $b(x)$ is being continuously differentiable. In our work we do not ask for this requirement and we will consider $b(x)$ continuous only.

Control barrier function are particularly useful in control theory when there is the need of certifying system safety. Specifically, we give the following definition.

Definition 2. *Let us consider a dynamical system* (6.1) *and an initial condition* x_0 *such that* $b(x_0) \leq 0$ *and a control input* $u(t)$, *with* $t \in [t_0, +\infty)$.

The trajectory $x(t) = \phi(t, t_0, x_0, u)$ *is said to be safe if* $b(x(t)) \leq 0$ *for all* $t \in [t_0, +\infty)$. *If* $x(t)$ *is safe,* $u(t)$ *is said a safe control trajectory.*

Definition 3. *Let us consider a feedback controller* $u = K(x)$ *for system* (6.1) *and, with some abuse of notation, let* $\phi(t, t_0; x_0, K(\cdot))$ *denote the closed-loop state* $x(t)$. *It is said a safe controller for set* $\mathcal{S}$ *if for any* x_0 *such that* $b(x_0) \leq 0$, *the trajectory* $x(t) = \phi(t, t_0, x_0, K(\cdot))$ *is safe.*

Roughly speaking, the above definition means that $K(\cdot)$ is safe if, for any safe initial condition, it keeps the system in the safe set forward in time.

A further (stronger) definition for a feedback controller is provided in what follows.

Definition 4. *Let us consider a feedback controller* $u = K(x)$ *for system* (6.1) *and let* $x(t) = \phi(t, t_0, x_0, K(\cdot))$ *be the system solution. The controller is said safety compliant for set* $\mathcal{S}$ *if*

i. $K(\cdot)$ *is safe;*

ii. for any x_0 *such that* $b(x_0) > 0$, *the function* $b(x(t))$ *with* $t \in [t_0, \bar{t}]$ *is monotonically decreasing (not necessarily strictly) for any* $\bar{t}$ *such that* $b(x(\bar{t})) > 0$.

Roughly speaking, the above definition qualifies any controller able to keep safety when the system starts in the safe set and able not to "increase" the unsafety (that is, the value of $b(x(\cdot))$ for any unsafe initial condition.

With respect to Definition 4, let us call $\mathcal{T} \subseteq [t_0, +\infty)$ the set of the time instants where $b(x(t))$ is strictly monotone, that is $b(x(t_1)) > b(x(t_2))$ for any $t_1, t_2 \in \mathcal{T}$ with $t_1 < t_2$. The following result holds.

Theorem 6.3.1. *Let us consider a smooth dynamical system* (6.1)*, a smooth feedback controller $K(x)$ and a continuous control barrier function $b(x)$ defined for the safe set $\mathcal{S}$. Suppose that the controller is safety compliant and the trajectory $x(t) = \phi(t, t_0, x_0, K(\cdot))$ is bounded for any valid initial condition $x_0 \in \mathbb{R}^n$ (that is, satisfying constraints* (6.1b)*). Also, suppose $\mathcal{T}$ unbounded. Then, the limit set $\omega(x_0) \subseteq \mathcal{S}$ for all x_0, [98].*

Proof. We need to prove the results only for those $x_0 \notin \mathcal{S}$, since for $x_0 \in \mathcal{S}$ the result trivially comes from Definition 4. First of all, consider a sequence of time instants $t_k \in \mathcal{T}$, with $\lim_{k\to+\infty} t_k = +\infty$ (this can be done since $\mathcal{T}$ is supposed to be unbounded). The proof will be conducted with a contradiction argument. Let us suppose $\lim_{k\to+\infty} b(x(t_k)) = \bar{b}$, with $\bar{b} > 0$. Let us call $b^{-1}(\bar{b}) : \{x \in \mathbb{R}^n : b(x) = \bar{b}\}$. We obviously have $b^{-1}(\bar{b}) \cap \mathcal{S} = \emptyset$. Let us consider the limit set $\omega(x_0)$ (note that such set is not empty since $x(t)$ is bounded, see [98]). We trivially have $\omega(x_0) \subseteq b^{-1}(\bar{b})$.

Let us consider a point $\tilde{x} \in \omega(x_0)$, a finite time span $\tau > 0$ and an index sequence h such that $t < t_h \leq t + \tau$. By the smoothness of $f(\cdot)$ and $K(\cdot)$, the function $\phi(t + \tau, t, \tilde{x}, K(\cdot))$ is continuous with respect to the variable $\tilde{x}$ for any initial time t and evolution length τ. Calling $\tilde{b} = b(\phi(t + \tau, t, \tilde{x}, K(\cdot)))$, we have $\tilde{b} < \bar{b}$. Let us choose a $\varepsilon < \bar{b} - \tilde{b}$. Then, by continuity we have that there exists a $\delta > 0$ such that any $\hat{x}$ such that $\|\tilde{x} - \hat{x}\| < \delta$ implies

$$\|b(\phi(t + \tau, t, \tilde{x}, K(\cdot))) - b(\phi(t + \tau, t, \hat{x}, K(\cdot)))\| < \varepsilon.$$

A direct consequence of the above inequality is that $b(\phi(t + \tau, t, \hat{x}, K(\cdot)) < \tilde{b}$ which implies that $\tilde{x}$ cannot belong to $\omega(x_0)$. Since this reasoning can be conducted for any point in $\omega(x_0)$, we have that the latter set is empty, leading to a contradiction. The reasoning can be repeated for any $\tilde{b} > 0$ implying that $\omega(x_0) \in \mathcal{S}$.

□

Let us now consider a variation of system (6.1), namely

$$\dot{x}(t) = f(x(t), u(t), p), \tag{6.3a}$$
$$\mathcal{C}(x(t), u(t), p) \leq 0, \tag{6.3b}$$
$$x_0 = x(t_0), \tag{6.3c}$$
$$u = K(x), \tag{6.3d}$$

where we explicitly pointed out the dependency of the system on a parameter vector $p \in \mathcal{P} \subset \mathbb{R}^q$, with $\mathcal{P}$ bounded set.

Obviously, system (6.3) solution depends on the value of p, that is $x(t) = \phi(t, t_0, x_0, K(\cdot), p)$. It could be the case (as in our manuscript) that the value of parameter p is not know. For this reason, we provide the following definitions.

Definition 5. *Let us consider a feedback controller $u = K(x)$ for system* (6.3)*. The controller is said robustly safe if, for any x_0 such that $b(x_0) \leq 0$, the trajectory $x(t) = \phi(t, t_0, x_0, K(\cdot), p)$ is safe for any $p \in \mathcal{P}$.*

Definition 6. *Let us consider a feedback controller $u = K(x)$ for system* (6.3)*. The controller is said robustly safety compliant if it is safety compliant for any $p \in \mathcal{P}$.*

The following theorem holds.

Theorem 6.3.2. *Let us consider the dynamical system* (6.3) *smooth for any $p \in \mathcal{P}$, a smooth feedback controller $K(\cdot)$ and a continuous control barrier function $b(\cdot)$ defined for the safe set $\mathcal{S}$. Suppose that the controller is robustly safety compliant and the trajectory $x(t) = \phi(t, t_0, x_0, K(\cdot), p)$ is bounded for any $p \in \mathcal{P}$ and any valid initial condition $x_0 \in \mathbb{R}^n$. Then, the limit set $\omega(x_0, p) \subseteq \mathcal{S}$ for all x_0.*

Proof. The proof is a direct consequence of Theorem 6.3.1 applied at any dynamical system model for any parameter value p. □

Finally, we provide the following definition.

Definition 7. *Let us consider a (possibly parameter dependent) dynamical system in the general form* (6.3) *and suppose $\mathcal{S}' \subseteq \mathbb{R}^n$ is its safe set. Suppose also to consider another set $\mathcal{S}$ such that $\mathcal{S} \subseteq \mathcal{S}'$ and a CBF $b(x)$ on $\mathcal{S}$ (that is, considering $\mathcal{S}$ as safe set). We will say that $b(x)$ protects $\mathcal{S}'$.*

Lemma 6.3.3. Let us consider a dynamical system (6.3) and sets $\mathcal{S} \subseteq \mathcal{S}' \subseteq \mathbb{R}^n$, with $\mathcal{S}'$ safe set for the system. Consider a CBF $b(x)$ protecting $\mathcal{S}'$ and a feedback controller $u = K(x)$. The following points hold:

i. If $K(\cdot)$ is a safe controller for $\mathcal{S}$, it is also a safe controller for $\mathcal{S}'$;

ii. If $K(\cdot)$ is a safety compliant controller for $\mathcal{S}$, it is also a safety compliant controller for $\mathcal{S}'$.

Proof. The proof trivially derives from the inclusion relation between set $\mathcal{S}$ and $\mathcal{S}'$. □

6.3.2 Setting

In this work, similarly to what done in [88], we consider two consecutive trains on the same line, the leader (denoted with superscript L) and the follower (denoted with superscript F) (the extension to more than two trains is not addressed in the work, although the proposed framework allows for that, which is left as a future work).

Both leader and follower have a dynamical model (whose details are provided later) summarized as

$$\dot{x}^i(t) = f^i(x^i(t), u^i(t), p^i), \tag{6.4a}$$
$$\mathcal{C}^i(x^i(t), u^i(t), p^i) \leq 0, \tag{6.4b}$$
$$x_0^i = x^i(t_0), \tag{6.4c}$$

with $x^i \in \mathbb{R}^{n_i}$, $u^i \in \mathbb{R}^{m_i}$, $p^i \in \mathcal{P}^i \subset \mathbb{R}^{q_i}$, $\mathcal{C}^i(\cdot) \in \mathbb{R}^{l_i}$, with $i \in \{\mathrm{L}, \mathrm{F}\}$.

The controllers of the two trains have generic expressions K^{L} and K^{F}.

Notice that, via stacking $x = [{x^{\mathrm{L}}}^T, {x^{\mathrm{F}}}^T]^T$, $u = [{u^{\mathrm{L}}}^T, {u^{\mathrm{F}}}^T]^T$, $p = [{p^{\mathrm{L}}}^T, {p^{\mathrm{F}}}^T]^T$, $x_0 = [{x_0^{\mathrm{L}}}^T, {x_0^{\mathrm{F}}}^T]^T$, $f = [{f^{\mathrm{L}}}^T, {f^{\mathrm{F}}}^T]^T$, $\mathcal{C} = [{\mathcal{C}^{\mathrm{L}}}^T, {\mathcal{C}^{\mathrm{F}}}^T]^T$ and setting $n = n_{\mathrm{L}} + n_{\mathrm{F}}$, $m = m_{\mathrm{L}} + m_{\mathrm{F}}$, $l = l_{\mathrm{L}} + l_{\mathrm{F}}$ and $q = q_{\mathrm{L}} + q_{\mathrm{F}}$ the overall dynamical system is formally represented by (6.3), with controller $K = [{K^{\mathrm{L}}}^T, {K^{\mathrm{F}}}^T]^T$.

Both controller K^{L} and K^{F} will be applied over a receding horizon (more details about these controllers will be provided later). The overall horizon for the leader is called H.

6.3.3 Channel Communication

In this study, we consider the train communication under ERTMS technology, which utilizes an event-triggered transmission protocol. This approach entails the transmission of signals or messages in response to specific operational events or conditions. We specifically consider the challenges posed by communication

delays and packet losses within the V2V context. In this regard, here we abstract how the communication happens on the physical layer and its implementation mechanisms (either via a direct communication between leader and follower or via the railway infrastructure) and focus on all its control related aspects.

Specifically, the leader transmits its information, from the current value up to a horizon H. Transmissions happen at instants $\{t_{\kappa^{\mathrm{L}}}\}_{\kappa^{\mathrm{L}}=0}^{+\infty}$ via sending the tuple $\big(t_{\kappa^{\mathrm{L}}}, x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}}), \underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})\big)$, where $t_{\kappa^{\mathrm{L}}}$ is the trigger instant, $x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})$ the state of the leader and $\underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}}) = \underline{x}^{\mathrm{L}}(t)_{t\in[t_{\kappa^{\mathrm{L}}}, t_{\kappa^{\mathrm{L}}}+H]}$ the whole robustly predicted state of the leader (due to its own control trajectory) up to horizon H. More details on such robust state estimation trajectory and in which sense it can be considered robust will be provided later.

The leader transmission instants $\{t_{\kappa^{\mathrm{L}}}\}_{\kappa^{\mathrm{L}}=0}^{+\infty}$ might be generated periodically or asynchronously (due to some event-triggered logic of the leader controller). In the current ERTMS, transmissions happen synchronously on a fixed period base. Therefore, for validating our algorithm we used in Section 6.7 a fixed sampling time transmission. However, the whole framework we are going to develop can perfectly work with non periodic transmissions.

The follower receives the leader data packets at instants $\{t_{\kappa^{\mathrm{LF}}}\}_{\kappa^{\mathrm{LF}}=0}^{+\infty}$.

Notice that, due to packet losses and delays in the communication or stochastic processing time, leader packets are not received synchronously by the follower (or not received at all), in other words

$$\{t_{\kappa^{\mathrm{L}}}\}_{\kappa^{\mathrm{L}}=0}^{+\infty} \neq \{t_{\kappa^{\mathrm{LF}}}\}_{\kappa^{\mathrm{LF}}=0}^{+\infty}.$$

The following functions are defined to represent the last trigger time for the leader and follower, respectively, enabling a precise analysis of data transmission timing:

$$\ell_{\kappa^{\mathrm{L}}}(t) = \max_{t_{\kappa^{\mathrm{L}}}\leq t} \{t_{\kappa^{\mathrm{L}}}\}_{\kappa^{\mathrm{L}}=0}^{+\infty},$$
$$\ell_{\kappa^{\mathrm{LF}}}(t) = \max_{t_{\kappa^{\mathrm{LF}}}\leq t} \{t_{\kappa^{\mathrm{LF}}}\}_{\kappa^{\mathrm{LF}}=0}^{+\infty}.$$

Suppose the leader transmits a packet at time $t_{\kappa^{\mathrm{L}}}$, and let $\tau(t_{\kappa^{\mathrm{L}}}) \in \mathbb{R}^+ \cup \{+\infty\}$ denote the elapsed time of follower reception of such packet, that is the follower receives the packet at $t_{\kappa^{\mathrm{L}}} + \tau(t_{\kappa^{\mathrm{L}}})$.

The sequence of $\{\tau(t_{\kappa^{\mathrm{L}}})\}_{\kappa^{\mathrm{L}}=0}^{+\infty}$ can be modeled as a random process as will be proposed in Section 6.6.3. Nevertheless, as it will be clearer later in the

manuscript, our framework does not need a specific transmission packets delay model.

6.4 Train Modeling

The model employed in this manuscript relies on the principles of longitudinal train dynamics. It treats the train as a singular point mass with one degree of freedom. Additionally, it incorporates aspects such as the propulsion and braking system, the effects of rolling and bearing resistances, air input, the influence of aerodynamic drag, as well as the consideration of grade and curving resistances [99]:

$$\begin{aligned} \dot{x}_1^i(t) &= x_2^i(t), \\ \dot{x}_2^i(t) &= \frac{1}{M^i}(-A^i - B^i x_2^i(t) - C^i (x_2^i(t))^2) - \mathrm{F}_e^i(t) + \frac{u^i(t)}{M^i}. \end{aligned} \tag{6.5}$$

We utilize the symbols $x_1^i(t)$ and $x_2^i(t)$ to represent the i-th train's position and velocity, respectively, with $i \in \{L, F\}$. By convention, we consider a reference frame at the head of each train (therefore the position is referred to as the train head). Also, we consider positive positions, with the origin at the beginning of the railway. Implicitly, both position and velocity will only be nonnegative (the trains cannot move backward during normal operations). The constraints $x_1^i(t) \geq 0$ and $x_2^i(t) \geq 0$ will be therefore implicitly included in compact term (6.4b).

The state vector is compactly written as $x^i(t) = (x_1^i(t), x_2^i(t))^T$. The variable $u^i(t)$ is the control driving or braking force; F_e^i denotes the external force originating from the track; M^i denotes the mass parameter, while A^i encompasses both rolling resistance and bearing resistance; B^i is a coefficient related to the flange friction, and C^i represents the aerodynamic coefficient. In this model, $\mathrm{F}_e^i(t)$ is the i-th external force

$$\mathrm{F}_e^i(t) = g\sigma(x_1^i(t)) + \frac{\gamma}{\rho(x_1^i(t))},$$

where $\gamma = 6 \cdot 10^6$ is a constant parameter. It encompasses two distinct terms: the first one is the gravity force resulting from the track's slope $\sigma(x_1^i(t))$ at point $x_1^i(t)$, with g representing the gravitational acceleration; the second term designates the curving resistance, with $\rho(x_1^i(t))$ representing the curve's radius.

Regarding the constraints on the state and input of the model, they are

presented as follows

$$u^i(t) \in \left[-M^i a^i_{\text{br}},\ M^i a^i_{\text{dr}}\right], \tag{6.6a}$$

$$u^i(t) \cdot x^i_2(t) \in \left[-P^i_{\text{br}},\ P^i_{\text{dr}}\right], \tag{6.6b}$$

$$x^i_2(t) \in \left[0,\ \min\{V^{\max,i}, V^{\text{line}}(x^i_1(t))\}\right], \tag{6.6c}$$

$$x^i_1(t) \in \mathbb{R}_+. \tag{6.6d}$$

All the above parameters are nonnegative and are constructional characteristics unique to each individual train and the railway. Specifically, a^i_{br} and a^i_{dr} correspond to the maximum braking and acceleration admitted, P^i_{br} and P^i_{dr} represent the minimum and maximum mechanical power, and $V^{\max,i}$ and V^{line} denotes the maximum attainable velocity for the train and railway velocity limit at position x^i_1, respectively.

6.4.1 Robust Modeling

In the context of railway control, safety is of utmost importance and must be guaranteed even with parametric uncertainties.

In this work, the following model parameters are supposed uncertain in a given range

$$\begin{gathered} a^i_{\text{br}} \in [\underline{a}^i_{\text{br}}, \overline{a}^i_{\text{br}}],\ a^i_{\text{dr}} \in [\underline{a}^i_{\text{dr}}, \overline{a}^i_{\text{dr}}],\ P^i_{\text{br}} \in [\underline{P}^i_{\text{br}}, \overline{P}^i_{\text{br}}], \\ P^i_{\text{dr}} \in [\underline{P}^i_{\text{dr}}, \overline{P}^i_{\text{dr}}],\ C^i \in [\underline{C}^i, \overline{C}^i], M^i \in [\underline{M}^i, \overline{M}^i], \\ A^i \in [\underline{A}^i, \overline{A}^i],\ B^i \in [\underline{B}^i, \overline{B}^i]. \end{gathered} \tag{6.7}$$

Compactly, we call $\mathcal{P}^i$ the Cartesian product of the above intervals. With such a choice, and taking into account (6.5) and (6.6), the train model can be compactly written as (6.4).

6.4.2 Robust Proxies

To derive a control architecture robust against any possible parameter choice of the two trains, the notions of Robust Lower Proxy (RLP) and Robust Upper Proxy (RUP) are introduced as simple (yet effective) robustification technique.

Definition 8 (RLP). *A RLP for the system* (6.4) *is any dynamical system of the form*

$$\dot{\underline{x}}^i(t) = \underline{f}^i(\underline{x}^i(t), u^i(t)), \tag{6.8}$$

with $\underline{x} \in \mathbb{R}^{n_i}$ *and* $u \in \mathbb{R}^{m_i}$ *and such that its dynamical flow* $\underline{\phi}^i\left(t, t_0, x_0^i, u^i\right)$ *satisfies*

$$\underline{\phi}^i\left(t, t_0, x_0^i, u^i\right) \leq \phi^i\left(t, t_0, x_0^i, u^i, p^i\right), \tag{6.9}$$

for all $x_0 \in \mathbb{R}^{n_i}$*, for all* $u_i \in \mathbb{R}^{m_i}$*, for all* $t \in [t_0, +\infty)$ *and for all* $p^i \in \mathcal{P}^i$ *and where the inequality is meant component-wise.*

Similarly, we provide the following definition.

Definition 9 (RUP). *A RUP for the system* (6.4) *is any dynamical system of the form*

$$\dot{\overline{x}}^i(t) = \overline{f}^i(\overline{x}^i(t), u^i(t)), \tag{6.10}$$

with $\overline{x} \in \mathbb{R}^{n_i}$ *and* $u \in \mathbb{R}^{m_i}$ *and such that its dynamical flow* $\overline{\phi}^i\left(t, t_0, x_0^i, u^i\right)$ *satisfies*

$$\overline{\phi}^i\left(t, t_0, x_0^i, u^i\right) \geq \phi^i\left(t, t_0, x_0^i, u^i, p^i\right) \tag{6.11}$$

for all $x_0 \in \mathbb{R}^{n_i}$*, for all* $u_i \in \mathbb{R}^{m_i}$*, for all* $t \in [t_0, +\infty)$ *and for all* $p^i \in \mathcal{P}^i$ *and where the inequality is meant component-wise.*

As evident from the two definitions, there might exist several proxies for (6.4). In this work, we will consider the following piece-wise smooth systems.

The following system has been chosen as RLP:

$$\begin{cases} \dot{\underline{x}}_1^i(t) = \underline{x}_2^i(t), & \text{if } u^i(t) < 0, \\ \dot{\underline{x}}_2^i(t) = \frac{1}{\underline{M}^i}\left(-\overline{A}^i - \overline{B}^i \underline{x}_2^i(t) - \overline{C}^i\left(\underline{x}_2^i(t_0)\right)^2\right) & \\ \qquad\qquad -\overline{F}_e^i(\underline{x}_1^i(t_0)) + \frac{u^i(t)}{\underline{M}^i}, & \\ \text{with } \overline{F}_e^i(\underline{x}_1^i(t_0)) = g\sigma_{\sup}\left(\underline{x}_1^i(t_0)\right) + \frac{\gamma}{\rho_{\inf}\left(\underline{x}_1^i(t_0)\right)}, & \\ \dot{\underline{x}}_1^i(t) = \underline{x}_2^i(t), & \text{if } u^i(t) \geq 0, \\ \dot{\underline{x}}_2^i(t) = \frac{1}{\underline{M}^i}\left(-\overline{A}^i - \overline{B}^i \underline{x}_2^i(t) - \overline{C}^i\left(\underline{x}_2^i(t)\right)^2\right) & \\ \qquad\qquad -\overline{F}_e^i(\underline{x}_1^i(t_0)) + \frac{u^i(t)}{\overline{M}^i}, & \\ \sigma_{\sup}\left(\underline{x}_1^i(t_0)\right) = \max_{s\in[\underline{x}_1^i(t_0), s^{H,i}]} \sigma(s), & \\ \rho_{\inf}\left(\underline{x}_1^i(t_0)\right) = \min_{s\in[\underline{x}_1^i(t_0), s^{H,i}]} \rho(s). & \end{cases} \tag{6.12}$$

Above, t_0 is a time instant upon which we consider the proxy time evolution and $s^{H,i}$ is a far enough ahead train position. Details on t_0 and $s^{H,i}$ will be provided later in the work.

Similarly, we consider the following piece-wise system as RUP:

$$\begin{cases} \dot{\overline{x}}_1^i(t) = \overline{x}_2^i(t), & \text{if } u^i(t) < 0, \\ \dot{\overline{x}}_2^i(t) = -\frac{\underline{A}^i}{\overline{M}^i} - \underline{F}_e^i(\overline{x}_1^i(t_0)) + \frac{u^i(t)}{\overline{M}^i}, & \\ \text{with } \underline{F}_e^i(\overline{x}_1^i(t_0)) = g\sigma_{\inf}\left(\overline{x}_1^i(t_0)\right) + \frac{\gamma}{\rho_{\sup}\left(\overline{x}_1^i t_0\right)}, & \\ \dot{\overline{x}}_1^i(t) = \overline{x}_2^i(t), & \text{if } u^i(t) \geq 0, \\ \dot{\overline{x}}_2^i(t) = \frac{1}{\overline{M}^i}\left(-\underline{A}^i - \underline{B}^i \overline{x}_2^i(t) - \underline{C}^i\left(\overline{x}_2^i(t)\right)^2\right) & \\ -\underline{F}_e^i(\overline{x}_1^i(t_0)) + \frac{u^i(t)}{\underline{M}^i}, & \\ \sigma_{\inf}\left(\overline{x}_1^i(t_0)\right) = \min_{s\in[\overline{x}_1^i(t_0), s^{H,i}]} \sigma(s), & \\ \rho_{\sup}\left(\overline{x}_1^i(t_0)\right) = \max_{s\in[\overline{x}_1^i(t_0), s^{H,i}]} \rho(s), & \end{cases} \tag{6.13}$$

where again as before t_0 and $s^{H,i}$ are an initial time and a position ahead to be determined later.

The following lemma states the validity of our choice.

Lemma 6.4.1. Systems (6.12) and (6.13) are valid RLP and RUP for the train dynamics (6.5)-(6.7).

Proof. The proof is immediate since it suffices to consider the following differential equations

$$\dot{\tilde{x}}^i = \overline{f}\left(\tilde{x}^i, u^i\right) - f\left(\tilde{x}^i, u^i, p^i\right),$$
$$\dot{\hat{x}}^i = f\left(\hat{x}^i, u^i\right) - \underline{f}\left(\hat{x}^i, u^i, p^i\right).$$

Since all component terms are non-negative, this ensures the positivity of the systems.

□

6.5 Safety Guarantee

One of the objectives of the train control system we propose in this work is to ensure that the two trains maintain a safety separation distance. In other words, the goal is to guarantee that the distance between the leader and follower never falls below the predefined safety threshold d. Remembering that the overall state is $x = [x^{\mathrm{L}T}, x^{\mathrm{F}T}]^T$, this is formally modeled through the safe set

$$\mathcal{S}'(x) = \{x : x_1^{\mathrm{L}} - x_1^{\mathrm{F}} \geq L^{\mathrm{L}} + d\}, \tag{6.14}$$

where L^{L} is the leader's length.

In order to ensure that the system solution always stays in $\mathcal{S}'(x)$, we consider for this work another safe set $\mathcal{S}$ such that $\mathcal{S} \subseteq \mathcal{S}'$ and a protecting CBF for $\mathcal{S}'(x)$, as for Definition 7.

To do so, let us first define $\underline{\alpha}^i\left(x_2^i\right) = \min\left\{\underline{a}_{\mathrm{br}}^i, \frac{P_{\mathrm{br}}^i}{\overline{M}^i x_2^i}\right\}$. Also, considering an $\varepsilon < 1$ (the latter is a tuning parameter that can be chosen slightly less than one to reduce conservatism), let us compute a "large" deceleration of the follower

$$\underline{e}^{\mathrm{L}}\left(x_2^{\mathrm{L}}\right) = -\frac{1}{\underline{M}^{\mathrm{L}}}\left(-\overline{A}^{\mathrm{L}} - \overline{B}^{\mathrm{L}}(x_2^{\mathrm{L}}) - \overline{C}^{\mathrm{L}}\left(x_2^{\mathrm{L}}\right)^2\right) + \overline{F}_e^{\mathrm{L}}(x_2^{\mathrm{L}}) + \overline{a}_{\mathrm{br}}^{\mathrm{L}},$$

and a "small" deceleration of the follower

$$\overline{e}^{\mathrm{F}}(x^{\mathrm{F}}) = \frac{\underline{A}^{\mathrm{F}}}{\overline{M}^{\mathrm{F}}} + \underline{F}_e^{\mathrm{F}}(x_1^{\mathrm{F}}) + \varepsilon\underline{\alpha}^{\mathrm{F}}(x_2^{\mathrm{F}}).$$

We define the functions

$$b'(x) = L^{\mathrm{L}} + d - (x_1^{\mathrm{L}} - x_1^{\mathrm{F}}) - \frac{1}{2}\left(\frac{{x_2^{\mathrm{L}}}^2}{\underline{e}^{\mathrm{L}}(x_2^{\mathrm{L}})} - \frac{{x_2^{\mathrm{F}}}^2}{\overline{e}^{\mathrm{F}}(x_2^{\mathrm{F}})}\right), \tag{6.15a}$$

$$b''(x) = L^{\mathrm{L}} + d - (x_1^{\mathrm{L}} - x_1^{\mathrm{F}}), \tag{6.15b}$$

$$b(x) = \max\{b'(x), b''(x)\}. \tag{6.15c}$$

The safe set $\mathcal{S}$ corresponds to those states resulting in $b(x) \leq 0$, that is $\mathcal{S} = \{x : b(x) \leq 0\}$ as defined in (6.2). The following lemma holds.

Lemma 6.5.1. Function $b(x)$ in (6.15c) is a CBF protecting the set $\mathcal{S}'$ in (6.14).

Proof. The set $\mathcal{S}'$ corresponds to those states where $b''(x) \leq 0$. Due to (6.15c), $b(x) \leq 0 \Rightarrow b''(x) \leq 0$. □

Before showing further properties of the CBF $b(x)$, we first provide some definitions.

Let us first call $\mathcal{U}^i(x^i)$ the robust set of all the admissible control inputs at state x^i for system (6.4), that is

$$\mathcal{U}^i(x^i) = \left\{u^i \in \mathbb{R}^{m_i} : \text{(6.4b), (6.6a), (6.6b) are satisfied } \forall p^i \in \mathcal{P}^i\right\}. \tag{6.16}$$

Together with the above, we call the robust set of all admissible braking control input at state x^i for system (6.4) the set

$$\underline{\mathcal{U}}^i(x^i) = \left\{u^i \in \mathcal{U}^i(x^i) : u^i \leq 0\right\}. \tag{6.17}$$

We are ready to provide the following definition.

Definition 10 (Braking Controller)**.** *We define a braking controller* $u^i = K_{\mathrm{B}}^i(x^i)$ *any feedback function such that*

- $u^i \in \underline{\mathcal{U}}^i(x^i)$, *if* $x_2^i > 0$;
- $u^i = 0$, *if* $x_2^i = 0$.

Roughly speaking, a braking controller is any controller providing a nonpositive control input whenever the velocity is greater than zero and a null control input in case of null velocity.[1]

[1]The case of null control input for null velocity could, in principle, be omitted since it is implicitly deducible by the positive constraint $x_2^i \geq 0$ of the system. Nevertheless, we prefer to keep it for the sake of clarity.

Among the braking controllers, the emergency controller is a peculiar case, where the controller attains the highest allowed braking action.

Definition 11 (Emergency Controller)**.** *We define the emergency controller* $u^i(t) = K^i_{\mathrm{E}}(x^i)$ *a braking controller with*

- $u^i = \min \underline{\mathcal{U}}^i(x^i)$, *if* $x^i_2 > 0$;
- $u^i = 0$, *if* $x^i_2 = 0$.

We are now ready for the following result.

Theorem 6.5.2. *Let us consider the trains system of form* (6.4) *with* $i \in \{\mathrm{L}, \mathrm{F}\}$ *in their stack form* (6.3)*. Let us consider any valid (that is, outputting values satisfying the system constraints) leader controller* K^{L}*. Let us consider that* $x^{\mathrm{F}}_2(t_0) > 0$ *(the follower is not standing at the initial time* t_0*). Then, the controller* $K = [{K^{\mathrm{L}}}^T, {K^{\mathrm{F}}_E}^T]^T$ *is robustly safety compliant for* $\mathcal{S}$ *(and therefore also for* $\mathcal{S}'$ *according to Lemma 6.3.3).*

Proof. To prove the result, let us first consider two special choices for the leader and follower controller, namely:

$$\hat{K}^{\mathrm{L}}(x^{\mathrm{L}}) = \begin{cases} \overline{a}^{\mathrm{L}}_{\mathrm{br}}, & \text{if } x^{\mathrm{L}}_2 > 0, \\ 0, & \text{if } x^{\mathrm{L}}_2 = 0, \end{cases}$$

$$\hat{K}^{\mathrm{F}}(x^{\mathrm{F}}) = \begin{cases} \varepsilon \underline{a}^{\mathrm{F}}(x^{\mathrm{F}}_2), & \text{if } x^{\mathrm{F}}_2 > 0, \\ 0, & \text{if } x^{\mathrm{F}}_2 = 0, \end{cases}$$

with, as said, $\varepsilon < 1$ (a good choice is slightly less than one).

Now, let us suppose to apply from the initial time t_0 controllers $\hat{K}^{\mathrm{L}}$ and $\hat{K}^{\mathrm{F}}$ at the leader's RLP and at the follower's RUP, respectively.

Also, let us suppose for the moment that at the considered initial time t_0 the leader is not standing, i.e. $x^{\mathrm{L}}_2(t_0) > 0$ (this hypothesis will be removed later).

Notice that the assumption $x^{\mathrm{F}}_2(t_0) > 0$ in the theorem statement and the technical one $x^{\mathrm{L}}_2(t_0) > 0$ we just added temporarily are only meant to allow these two controllers to provide non zero input, since, in case a train is still, the set $\mathcal{U}^i(x^i) = \{0\}$ is the only available controller satisfying dynamics (6.5)–(6.7).

With such a choice, it is immediate to verify that the leader RLP dynamics are

$$\dot{\underline{x}}_1^{\mathrm{L}} = \underline{x}_2^{\mathrm{L}} \tag{6.18a}$$

$$\dot{\underline{x}}_2^{\mathrm{L}} = -\underline{e}^{\mathrm{L}}\left(x_2^{\mathrm{L}}(t_0)\right), \tag{6.18b}$$

$$\underline{x}^{\mathrm{L}}(t_0) = x^{\mathrm{L}}(t_0), \tag{6.18c}$$

while the follower RUP dynamics are

$$\dot{\overline{x}}_1^{\mathrm{F}} = \overline{x}_2^{\mathrm{F}} \tag{6.19a}$$

$$\dot{\overline{x}}_2^{\mathrm{F}} = -\overline{e}^{\mathrm{F}}\left(x^{\mathrm{F}}(t_0)\right), \tag{6.19b}$$

$$\overline{x}^{\mathrm{F}}(t_0) = x^{\mathrm{F}}(t_0). \tag{6.19c}$$

Let us define $t_{\mathrm{st}}^{\mathrm{L},p}(t_0|\hat{K}^{\mathrm{L}}) = \min\{t \in [t_0, +\infty) : \underline{x}_2^{\mathrm{L}}(t) = 0\}$ the stopping time of the leader RLP subjected to controller $\hat{K}^{\mathrm{L}}$ applied from time t_0 and, similarly, let us define $t_{\mathrm{st}}^{\mathrm{F},p}(t_0|\hat{K}^{\mathrm{F}}) = \min\{t \in [t_0, +\infty) : \overline{x}_2^{\mathrm{F}}(t) = 0\}$ the stopping time of the follower RUP subjected to $\hat{K}^{\mathrm{F}}$ from t_0.

From the dynamics (6.18) and (6.19), we have

$$t_{\mathrm{st}}^{\mathrm{L},p}(t_0|\hat{K}^{\mathrm{L}}) = \frac{x_2^{\mathrm{L}}(t_0)}{\underline{e}^{\mathrm{L}}\left(x_2^{\mathrm{L}}(t_0)\right)} + t_0 \tag{6.20}$$

at position

$$\underline{x}_1^{\mathrm{L}}(t_{\mathrm{st}}^{\mathrm{L},p}) = x_1^{\mathrm{L}}(t_0) + \frac{1}{2}\frac{x_2^{\mathrm{L}}(t_0)^2}{\underline{e}^{\mathrm{L}}\left(x_2^{\mathrm{L}}(t_0)\right)}, \tag{6.21}$$

while the follower's RUP stops at time $t_{\mathrm{st}}^{\mathrm{F},p}(t_0|\hat{K}^{\mathrm{F}}) = \frac{x_2^{\mathrm{F}}(t_0)}{\overline{e}^{\mathrm{F}}\left(x^{\mathrm{F}}(t_0)\right)} + t_0$ at position

$$\overline{x}_1^{\mathrm{F}}(t_{\mathrm{st}}^{\mathrm{F},p}) = x_1^{\mathrm{F}}(t_0) + \frac{1}{2}\frac{x_2^{\mathrm{F}}(t_0)^2}{\overline{e}^{\mathrm{F}}\left(x^{\mathrm{F}}(t_0)\right)}. \tag{6.22}$$

The position difference of the two trains proxies when they are both stopped is, therefore,

$$\begin{aligned}\underline{x}_1^{\mathrm{L}}(t_{\mathrm{st}}^{\mathrm{L},p}) - \overline{x}_1^{\mathrm{F}}(t_{\mathrm{st}}^{\mathrm{F},p}) =& x_1^{\mathrm{L}}(t_0) - x_1^{\mathrm{F}}(t_0) \\ &+ \frac{1}{2}\left(\frac{x_2^{\mathrm{L}}(t_0)^2}{\underline{e}^{\mathrm{L}}\left(x_2^{\mathrm{L}}(t_0)\right)} - \frac{x_2^{\mathrm{F}}(t_0)^2}{\overline{e}^{\mathrm{F}}\left(x_2^{\mathrm{F}}(t_0)\right)}\right).\end{aligned} \tag{6.23}$$

It is worth noticing that when the proxies are stopped, being $\mathcal{U}^{\mathrm{L}}(\underline{x}_1^{\mathrm{L}}(t_{\mathrm{st}}^{\mathrm{L},p})) = \{0\}$ and $\mathcal{U}^{\mathrm{F}}(\overline{x}_1^{\mathrm{F}}(t_{\mathrm{st}}^{\mathrm{F},p})) = \{0\}$, we have that $\underline{x}_1^{\mathrm{L}}(t) = \underline{x}_1^{\mathrm{L}}(t_{\mathrm{st}}^{\mathrm{L},p})$ at any $t \geq t_{\mathrm{st}}^{\mathrm{L},p}$ and, similarly, $\overline{x}_1^{\mathrm{F}}(t) = \overline{x}_1^{\mathrm{F}}(t_{\mathrm{st}}^{\mathrm{F},p})$ at any time $t \geq t_{\mathrm{st}}^{\mathrm{F},p}$.

When comparing (6.23) with $b'(x)$ in (6.15a), it is immediate to notice that

$$b'([\underline{x}^{\mathrm{L}^T}(t_0), \overline{x}^{\mathrm{F}^T}(t_0)]^T) = L^{\mathrm{L}} + d - \left(\underline{x}_1^{\mathrm{L}}(t_{\mathrm{st}}^{\mathrm{L},p}) - \overline{x}_1^{\mathrm{F}}(t_{\mathrm{st}}^{\mathrm{F},p})\right). \tag{6.24}$$

From the above equation we have that, when evaluated at $x = [\underline{x}^{\mathrm{L}^T}(t_0), \overline{x}^{\mathrm{F}^T}(t_0)]^T$, $b'(x)$ can be interpreted as the difference between the minimum required inter-trains distance $L^{\mathrm{L}} + d$ and the proxy leader-follower distance achieved when considering that, at a generic initial time instant t_0, the two proxies engage controllers $\hat{K}^{\mathrm{L}}$ and $\hat{K}^{\mathrm{F}}$, respectively, until they stop. Notice also that, in view of this interpretation, the following equality holds[2]

$$b'(\underline{x}^{\mathrm{L}}(t), \overline{x}^{\mathrm{F}}(t)) = b'(\underline{x}^{\mathrm{L}}(t_0), \overline{x}^{\mathrm{F}}(t_0)), \tag{6.25}$$

at any time instant $t > t_0$ with $\underline{x}^{\mathrm{L}}(t) = \underline{\phi}^{\mathrm{L}}(t, t_0, x^{\mathrm{L}}(t_0), \hat{K}^{\mathrm{L}})$ and $\overline{x}^{\mathrm{F}}(t) = \overline{\phi}^{\mathrm{F}}(t, t_0, x^{\mathrm{F}}(t_0), \hat{K}^{\mathrm{F}})$, that is evaluated on the dynamical flow of the two proxies subjected to the controllers $\hat{K}^{\mathrm{L}}$ and $\hat{K}^{\mathrm{F}}$.

For shorting the notation, let us rewrite $\underline{x}^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}})$ to denote the RLP trajectory of the leader under controller $\hat{K}^{\mathrm{L}}$ and initial condition $x^{\mathrm{L}}(t_0)$ and, similarly $\overline{x}^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})$ to denote the RUP trajectory of the follower under controller $\hat{K}^{\mathrm{F}}$ and initial condition $x^{\mathrm{F}}(t_0)$.

Analogously (and still considering the initial conditions $x^{\mathrm{L}}(t_0)$ and $x^{\mathrm{F}}(t_0)$), let us denote with $x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}})$ and $x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})$ the trajectory of the leader and the follower under the controllers $\hat{K}^{\mathrm{L}}$ and $\hat{K}^{\mathrm{F}}$, respectively. By the properties of RLP and RUP (Definition 8 and Definition 9) the following inequalities hold

$$x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}) \geq \underline{x}^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), \tag{6.26a}$$

$$x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}}) \leq \overline{x}^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}}), \tag{6.26b}$$

for any $t > t_0$.

Notice also that, so far, we considered the technical assumption that the initial velocity of the leader is not null, so as to apply the controller $\hat{K}^{\mathrm{L}}$. Nevertheless, inequality (6.26a) still holds (with the equality relation) also in case the leader has

[2]To ease the notation we will write $b'(\underline{x}^{\mathrm{L}}(t), \overline{x}^{\mathrm{F}}(t))$ instead of $b'([\underline{x}^{\mathrm{L}^T}(t), \overline{x}^{\mathrm{F}^T}(t)]^T)$ (and similarly mutatis mutandis for the other terms on which $b(\cdot)$ is evaluated).

null initial velocity. In that case, being the null input the only allowed, we have $\hat{K}^{\mathrm{L}}(\underline{x}_L) = 0$.

From (6.26a) and (6.26b) we have

$$b'(x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})]) \leq b'(\underline{x}^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), \overline{x}^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})), \qquad \forall t > t_0. \tag{6.27}$$

Let $t^F_{\mathrm{st}}(t_0|\hat{K}^{\mathrm{F}}) = \min\{t \in [t_0, +\infty) : x^{\mathrm{F}}_2 = 0\}$ represent the stopping time of the follower under the control $\hat{K}^{\mathrm{F}}$ applied from time t_0. Additionally, noting that $t^F_{\mathrm{st}} \leq t^{\mathrm{F},p}_{\mathrm{st}}$, the following inequality holds for the above formula

$$b'(x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})) < b'(\underline{x}^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), \overline{x}^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})), \ \forall t_0 \geq t \geq t^F_{\mathrm{st}}.$$

Combining (6.27) with (6.25), we obtain

$$b'(x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})) \leq b'(\underline{x}^{\mathrm{L}}(t_0), \overline{x}^{\mathrm{F}}(t_0))), \qquad \forall t > t_0. \tag{6.28}$$

Being t and t_0 two generic instants, inequality (6.28) shows that $b'(x)$ is monotonically decreasing on the trains system under the control provided by $\hat{K}^{\mathrm{L}}$ and $\hat{K}^{\mathrm{F}}$.

To show that the special controller $K = [{K^{\mathrm{L}}}^T, {K^{\mathrm{F}}_E}^T]^T$ is robustly safety compliant we need to analyze the evolution of $b(x)$ along the system trajectory also when $b(x)$ attains its maximum on $b''(x)$. To do so, let us consider the case that $b(\underline{x}^{\mathrm{L}}(t_0), \overline{x}^{\mathrm{F}}(t_0)) = b''(\underline{x}^{\mathrm{L}}(t_0), \overline{x}^{\mathrm{F}}(t_0))$ which, equivalently, reads as

$$b'(\underline{x}^{\mathrm{L}}(t_0), \overline{x}^{\mathrm{F}}(t_0)) < b''(\underline{x}^{\mathrm{L}}(t_0), \overline{x}^{\mathrm{F}}(t_0)). \tag{6.29}$$

Let us again consider the evolution of the two proxies $\underline{x}^{\mathrm{L}}_1(t^{\mathrm{L},p}_{\mathrm{st}})$ and $\overline{x}^{\mathrm{F}}_1(t^{\mathrm{F},p}_{\mathrm{st}})$. By considering (6.23) and (6.24), from (6.29) it follows that

$$\underline{x}^{\mathrm{L}}_1(t^{\mathrm{L},p}_{\mathrm{st}}) - \overline{x}^{\mathrm{F}}_1(t^{\mathrm{F},p}_{\mathrm{st}}) > x^{\mathrm{L}}_1(t_0) - x^{\mathrm{F}}_1(t_0). \tag{6.30}$$

Therefore, from the above equation, the case (6.29) yields an inter proxies distance when both are stopped greater than the initial one. Furthermore, by considering the position error dynamics, from (6.18) and (6.19) we have

$$\begin{aligned} &\dot{\underline{x}}^{\mathrm{L}}_1(t) - \dot{\overline{x}}^{\mathrm{F}}_1(t) = \\ &x^{\mathrm{L}}_2(t_0) - x^{\mathrm{F}}_2(t_0) - \left(\underline{e}^{\mathrm{L}}\left(x^{\mathrm{L}}_2(t_0)\right) - \overline{e}^{\mathrm{F}}\left(x^{\mathrm{F}}_2(t_0)\right)\right)(t - t_0). \end{aligned} \tag{6.31}$$

It is immediate to observe from (6.31) that the proxies distance is monotone and, by considering (6.30) we have that $\dot{\underline{x}}_1^{\mathrm{L}}(t) - \dot{\overline{x}}_1^{\mathrm{F}}(t) > 0$ for any $t < \max\{t_{\mathrm{st}}^{\mathrm{L},p}, t_{\mathrm{st}}^{\mathrm{F},p}\}$, while remaining constant otherwise. Therefore, it is monotone increasing. This leads to

$$b''(x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})) < b''(\underline{x}^{\mathrm{L}}(t_0), \overline{x}^{\mathrm{F}}(t_0)). \tag{6.32}$$

Combining (6.28) and (6.32) we prove the monotonicity of $b(x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}}))$ and, hence, that the special controller we considered it robustly safety compliant.

To prove that the controller $K = [{K^{\mathrm{L}}}^T, {K_E^{\mathrm{F}}}^T]^T$ is robustly safe compliant, let us denote with $x^{\mathrm{L}}(t|K^{\mathrm{L}})$ the trajectory of the leader under any controller K^{L} and with $x^{\mathrm{F}}(t|K_{\mathrm{E}}^{\mathrm{F}})$ the trajectory of the follower under the emergency controller $K_{\mathrm{E}}^{\mathrm{F}}$. By again considering the properties of RLP and RUP and the definition of emergency controller (Definition 11) the following inequalities hold

$$x^{\mathrm{L}}(t|K^{\mathrm{L}}) \geq x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), \tag{6.33}$$

$$x^{\mathrm{F}}(t|K_{\mathrm{E}}^{\mathrm{F}}) \leq x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}}), \tag{6.34}$$

for any $t > t_0$.

Hence, $x^{\mathrm{L}}(t|K^{\mathrm{L}}) - x^{\mathrm{F}}(t|K_{\mathrm{E}}^{\mathrm{F}}) \geq x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}) - x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})$ and, therefore,

$$b([x^{\mathrm{L}}(t|K^{\mathrm{L}}), x^{\mathrm{F}}(t|K_{\mathrm{E}}^{\mathrm{F}})]^T) \leq b([x^{\mathrm{L}}(t|\hat{K}^{\mathrm{L}}), x^{\mathrm{F}}(t|\hat{K}^{\mathrm{F}})]^T) \qquad \forall t > t_0.$$

□

6.6 Control system

In this section, we introduce the four blocks of the proposed control architecture which are depicted in 6.2: the Leader RLP predictor, the Safety control, the Cruise virtual coupling and the Delay estimator.

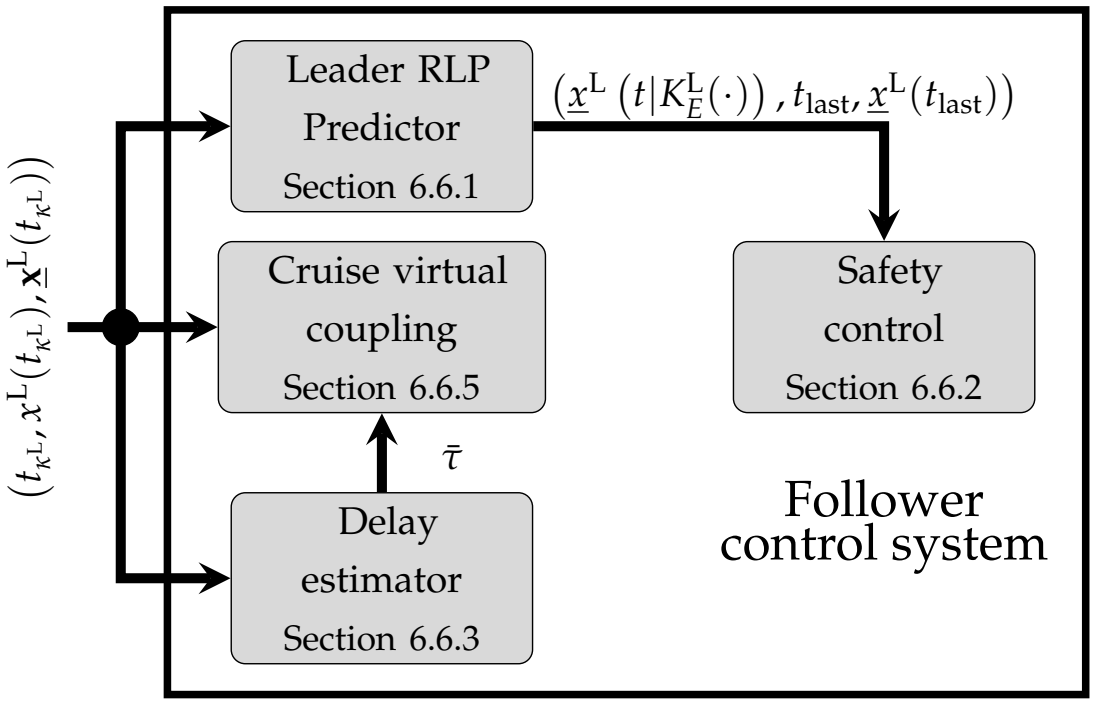


Figure 6.2: Follower control system architecture for VC operation.

The Leader RLP predictor block is crucial for the follower train, as it processes data packets from the leader, received at irregular intervals. It continuously updates the follower's predictions of the leader's state, starting from the last received packet and projecting forward using the leader's dynamical model, supposing (for unknown future control inputs) the activation of an emergency brake. The overall safety is managed by the Safety control block which continuously monitors the state of the follower and the predicted leader RLP state. The block uses specific safety conditions to determine when to enforce the emergency controller on the follower train. When safety conditions are not met the emergency controller is activated to ensure the follower train maintains a safe distance from the leader. The Cruise virtual coupling block synchronizes the follower train with the leader, maintaining a safe distance using the predicted trajectory data. This block adjusts the follower's control strategy to match the leader's state, avoids unnecessary braking by keeping an appropriate gap based on communication delays, and employs a switched controller architecture to manage performances under varying conditions. This ensures efficient and safe operation of the follower train in response to the leader's movements. The last block, the Delay estimator, is responsible for estimating the communication delay, which is expected to vary along the route. This estimator continuously

monitors and analyzes the communication signals to accurately predict the latency, allowing the control system to adjust accordingly. We wish to emphasize that this block is not necessary, and a fixed delay estimation could be assumed (at the expense, in general, of a more conservative inter-trains gap). The architecture we are going to develop is indeed very general and can be particularized for several design control choices including different possible delay estimators, different follower controllers, different safety rules. As it will be clearer later, the control system architecture we propose guarantees safety, robustness and some important performances on the emergency braking, leaving degrees of freedom to such control choices.

The following subsections will introduce the aforementioned control system blocks in detail.

6.6.1 Leader RLP predictor

As already reported in Section 6.3.3, the follower receives at instants $\{t_{\kappa^{\mathrm{LF}}}\}_{\kappa^{\mathrm{LF}}=0}^{+\infty}$ the asynchronous data from the leader (transmission time, state at the transmission time and a robust prediction of the state trajectory).

Since the follower is not aware of what actual control input the leader is adopting in the time between consecutive message receptions, the follower does not know of the real leader state. It is worth mentioning that among the control input options available to the leader, the emergency control is the most important to be considered. Indeed, right after a communication at time $\ell_{\kappa^{\mathrm{L}}}(t)$, the leader might start an emergency braking which could lead, if not adequately addressed, to safety hazards.

To avoid this, Algorithm 4 is run by the follower and executes a Leader RLP predictor in presence of possible emergency braking. Such prediction block will then be used in the next section for the safety control.

For consistency, we assume Algorithm 4 to be started the first time a follower receives its first packet, with initial conditions $t_{\mathrm{last}} = -\infty$, $\underline{x}^{\mathrm{L}}(t_{\mathrm{last}}) = +\infty$.

As it is possible to see, Algorithm 4 is continuously run in background, integrating the leader RLP under an emergency controller, resetting the initial time and initial condition on an event triggered basis at the most recent received values. The outputs of the algorithm are the predicted leader RLP state subjected to the emergency controller and here denoted as $\underline{x}^{\mathrm{L}}(t|K_{\mathrm{E}}^{\mathrm{L}})$ at the current time t, and the time of the most recent information from the leader, together with its state, respectively t_{last}, and $\underline{x}^{\mathrm{L}}(t_{\mathrm{last}})$. These two latter information, together with

Algorithm 4 Leader RLP predictor. Outputs: $\underline{x}^{\mathrm{L}}(t|K_{\mathrm{E}}^{\mathrm{L}})$, t_{last}, $\underline{x}^{\mathrm{L}}(t_{\mathrm{last}})$.

loop
  Listen to possible transmission of information from the leader;
  if A packet $\left(t_{\kappa^{\mathrm{L}}}, x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}}), \underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})\right)$ is received with $t_{\kappa^{\mathrm{L}}} > t_{\mathrm{last}}$ **then**
    $t_{\mathrm{last}} \leftarrow t_{\kappa^{\mathrm{L}}}$
    $\underline{x}^{\mathrm{L}}(t_{\mathrm{last}}) \leftarrow x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})$
  Integrate the dynamical model $\dot{\underline{x}}^{\mathrm{L}}(t) = \underline{f}^{\mathrm{L}}(\underline{x}^{\mathrm{L}}(t), K_{\mathrm{E}}^{\mathrm{L}}(\underline{x}^{\mathrm{L}}(t)))$ from the initial time t_{last} and initial condition $\underline{x}^{\mathrm{L}}(t_{\mathrm{last}})$ up to current time (the resulting trajectory will be denoted with $\underline{x}^{\mathrm{L}}(t|K_{\mathrm{E}}^{\mathrm{L}})$);

$\underline{x}^{\mathrm{L}}(t|K_{\mathrm{E}}^{\mathrm{L}})$, might be useful for some safety enforcing condition as described in the next section.

Notice that Algorithm 4 does not use the robust leader predicted trajectory estimation $\underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})$. Notice also that so far we did not describe yet how $\underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})$ is computed. This will be done later since $\underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})$ will be exploited in the Cruise virtual coupling control block.

6.6.2 Safety control block

This block allows to constantly keep the system safe by monitoring the follower state $x^{\mathrm{F}}(t)$ and the leader RLP prediction from the corresponding block outputs described in Section 6.6.1. This block receives in input, together with $x^{\mathrm{F}}(t)$, the t_{last} and $\underline{x}^{\mathrm{L}}(t|K_{\mathrm{E}}^{\mathrm{L}})$ from the RLP predictor block. To ease the notation, we denote in this subsection $\underline{x}^{\mathrm{L}}(t|K_{\mathrm{E}}^{\mathrm{L}})$ as $\underline{x}^{\mathrm{L}}(t)$, omitting the fact that the trjectory of the RLP in Algorithm 4 is generated supposing an emergency braking.

Specifically, considering $c(x^{\mathrm{F}}(t), \underline{x}^{\mathrm{L}}(t), t_{\mathrm{last}}, \underline{x}^{\mathrm{L}}(t_{\mathrm{last}}))$ any possible safety rule (further details on this will be provided later), the Safety control block is implemented by Algorithm 5 which continuously runs in background.

Algorithm 5 Safety control

loop
  Continuously monitor $b(\underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t))$ and $c(x^{\mathrm{F}}(t), \underline{x}^{\mathrm{L}}(t), t_{\mathrm{last}}, \underline{x}^{\mathrm{L}}(t_{\mathrm{last}}))$, with $b(x)$ given in (6.15c);
  if $b(\underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t)) \geq 0$ **or** $c(x^{\mathrm{F}}(t), \underline{x}^{\mathrm{L}}(t), t_{\mathrm{last}}, \underline{x}^{\mathrm{L}}(t_{\mathrm{last}})) \geq 0$ **then**
    Enforce the emergency controller for the follower, that is $K_{\mathrm{E}}^{\mathrm{F}}$;
  else
    Do not enforce/stop enforcing the follower emergency controller $K_{\mathrm{E}}^{\mathrm{F}}$;

In practice, Algorithm 5 continuously monitor the train system safety

condition and, in case it is violated (or at the boundary of its violation) enforce the follower emergency controller. When no safety violation is detected, the emergency controller is turned off.

This is formally stated in the next result.

Theorem 6.6.1. *Consider the trains system of form* (6.4) *with* $i \in \{\mathrm{L}, \mathrm{F}\}$ *and the safe set* $\mathcal{S}(x) = \{x : b(x) \leq 0\}$. *If the system starts in a safe state, that is* $[x^{\mathrm{L}^T}(t_0), x^{\mathrm{F}^T}(t_0)]^T \in \mathcal{S}$, *then the Safety control block of Algorithm 5 keeps the system safe forward in time, that is* $[x^{\mathrm{L}^T}(t), x^{\mathrm{F}^T}(t)]^T \in \mathcal{S}$ *for any* $t \geq t_0$.

Proof. By hypothesis the system starts in the safe set. Being $\underline{x}^{\mathrm{L}}(t)$ and $x^{\mathrm{F}}(t)$ continuous time curves and taking into account that $x^{\mathrm{L}}(t) \geq \underline{x}^{\mathrm{L}}(t)$, safety is preserved until $b(\underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t)) = 0$ and, therefore, $b(x^{\mathrm{L}}(t), x^{\mathrm{F}}(t)^T) \leq b(\underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t))$. Since when $b(\underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t)) = 0$ controller $K_{\mathrm{E}}^{\mathrm{F}}$ is activated, by Theorem 6.5.2 safety is kept. □

Remark. *As it is possible to notice, the activation of* $K_{\mathrm{E}}^{\mathrm{F}}$ *due to the triggers generated by the violation of condition* $c(x^{\mathrm{F}}(t), \underline{x}^{\mathrm{L}}(t), t_{\mathrm{last}}, \underline{x}^{\mathrm{L}}(t_{\mathrm{last}})) \geq 0$ *does not play any role in the proof of Theorem 6.6.1. For this reason, rule* $c(x^{\mathrm{F}}(t), \underline{x}^{\mathrm{L}}(t), t_{\mathrm{last}}, \underline{x}^{\mathrm{L}}(t_{\mathrm{last}}))$ *can be omitted by trivially setting* $c(x^{\mathrm{F}}(t), \underline{x}^{\mathrm{L}}(t), t_{\mathrm{last}}, \underline{x}^{\mathrm{L}}(t_{\mathrm{last}})) = -\infty$ *without harming the system safety. Therefore, any other choice of* $c(\cdot)$ *is allowed. The reason why we decided to report rule* $c(\cdot)$ *is motivated by the peculiar application we are considering. In train control, extra/redundant safety conditions are often adopted, even though they do not look necessary from a strict mathematical analysis. Specifically, being this work developed under the collaboration with personnel from the Italian railway company, the condition*

$$c(x^{\mathrm{F}}(t), \underline{x}^{\mathrm{L}}(t), t_{\mathrm{last}}, \underline{x}^{\mathrm{L}}(t_{\mathrm{last}})) = t - t_{\mathrm{last}} - T^{\max}, \tag{6.35}$$

is adopted, where $T^{\max}$ *is a prescribed maximum allowed time to wait for a new transmission update. As said, despite not necessary for keeping the system safe, the railway operator prefers to stop the follower train if no updates on the leader position are received after* $T^{\max}$. *Therefore, in this work we will consider* (6.35).

A good choice for $T^{\max}$ is having it bigger than the statistically relevant delay experienced by the communication channel. Its adoption turns to be useful in setting $s^{H,i}$ in (6.12) and (6.13), with the easy choice $s^{H,i} = x_1^i(t_0) + V_{\max}^i T^{\max}$.

Algorithm 5 implicitly provides a sequence of events in which the emergency braking is activated (and a sequence in which it is deactivated). We denote with $\{t_{\kappa^{\mathrm{E}}}\}_{\kappa^{\mathrm{E}}=0}^{+\infty}$ the sequence of time instants of emergency braking activation.

6.6.3 Delay estimator block

Given the variability in channel communication along railway lines, it is advantageous to introduce a block that estimates the communication delay between the leader and follower. As said, the follower receives data asynchronously. The sequence of delays $\{\tau(t_{\kappa^{\mathrm{L}}})\}_{\kappa^{\mathrm{L}}=0}^{+\infty}$ (see Section 6.3.3) is influenced by various factors, e.g. stochastic properties of channel communication and hardware processing time. To this end, let τ denote a realization of the random variable T, and consider a sample of observed values stored sequentially in a finite stack, defined as $\Lambda = \{\tau_1, \tau_2, \ldots, \tau_{N_\Lambda}\}$, which represents a sequence of N_Λ independent realizations of T. We denote the empirical cumulative distribution function based on this sample by $\hat{F}_\Lambda(\tau)$.

Given $\hat{F}_\Lambda(\tau)$ and a chosen probability threshold $\bar{p} \in [0,1]$ for receiving a packet within a specified time, the delay estimate $\bar{\tau}$ can be calculated to represent the expected packet delay. Algorithm 6 outlines the steps involved in this estimation process.

Algorithm 6 Delay estimator. Output: $\bar{\tau}$.

1: **loop**
2: Listen to possible transmission of information from the leader;
3: **if** A packet $\left(t_{\kappa^{\mathrm{L}}}, x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}}), \underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})\right)$ is received with $t_{\kappa^{\mathrm{L}}} > t_{\text{last}}$ **then**
4: Store $\tau(t_{\kappa^{\mathrm{L}}})$ in Λ
5: Compute $\bar{\tau} = \inf\{\tau : \hat{F}_\Lambda(\tau) \geq \bar{p}\}$

The estimate is updated each time a new packet arrives from the leader. In this case, the estimator calculates $\bar{\tau}$ over the time interval $t \in [t_{\kappa^{\mathrm{L}}-N_\Lambda}, t_{\kappa^{\mathrm{L}}}]$.

As mentioned previously, this block may be omitted since the proposed architecture ensures both safety and a minimum response time before emergency braking is triggered. However, the inclusion of a delay estimator enables dynamic estimation of communication delays, reducing the occurrence of unnecessary braking events. Additionally, we highlight that the flexibility of the proposed framework allows for various alternative approaches to the delay estimator.

In Section 6.7, we will demonstrate the benefits of incorporating this block within the control architecture through simulations.

6.6.4 Cruise virtual coupling block: fixed controller

It is worth to observe that the safety condition in line 3 of Algorithm 5 might be restored either because the follower, while braking, is increasing its distance from $\underline{x}^{\mathrm{L}}(t)$ and/or because a new update of the leader position is received, thus resulting in new initialization update of the Leader RLP predictor (line 7 of Algorithm 4.) This latter case is probably the most interesting in practice, since in normal (desirable) conditions, the leader train proceeds at constant or smoothly variable velocity without incurring in aggressive braking. Similarly, the desired virtual coupling condition is that the follower proceeds at nearly the same velocity of the leader at a certain (obviously safe) distance from it.

The Safety control block is designed to always preserve safety via enforcing the follower to brake in the worst possible condition, that is supposing that from the last transmission the leader starts a maximum braking. Since, as said, this condition in practice will be rare, a good choice for the follower distance from the leader during VC cruise is such that it could proceed at the same velocity the leader has planned for a given amount of time $\bar{\tau}$ before the condition of line 3 of Algorithm 5 is met. In this way, at least a time $\bar{\tau}$ is wait before a follower emergency braking occurs. With the hypothesis of having the statistical description of the communication delay process $\{\tau(t_{\kappa^{\mathrm{L}}})\}_{\kappa^{\mathrm{L}}=0}^{+\infty}$ reported in Section 6.6.3, choosing a $\bar{\tau}$ as in line 5 of Algorithm 6 ensures that $100\bar{p}$ percent of the times a leader state update is received before triggering any follower braking (which will proceed roughly maintaining the leader velocity).

In this section we formally describe a control block able to guarantee the behaviour described before and implement VC between the two trains. To do so, let us first recall that the follower receives asynchronous packets $\left(t_{\kappa^{\mathrm{L}}}, x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}}), \underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})\right)$ with $\underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})$, as said, robust leader' state trajectory estimation up to horizon H. Specifically, the latter is computed via any valid MPC embedded on the leader over its nominal model. The computed control $\mathbf{u}^{\mathrm{L}}(t)$ for the time interval $[t_{\kappa^{\mathrm{L}}}, t_{\kappa^{\mathrm{L}}} + H)$ is then applied open loop to the leader RLP (initialized at state $x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})$) to compute $\underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})$, which results in a robust estimated leader trajectory. In the framework we are going to propose, we consider the choice

$$\bar{\tau} < T^{\max} \ll H. \tag{6.36}$$

Let us consider again controller $\hat{K}^{\mathrm{L}}$ as defined in the proof of Theorem 6.5.2 and $t_{\mathrm{st}}^{\mathrm{L},p}(t|\hat{K}^{\mathrm{L}})$ as defined in (6.20) (here for a generic time t). For brevity, from now

on we will omit the dependency on the controller $\hat{K}^{\mathrm{L}}$ and we will write $t_{\mathrm{st}}^{\mathrm{L},p}(t)$. We define

$$v_{\bar{\tau}}^{\mathrm{L}}\left(t,\underline{x}^{\mathrm{L}}\right)=\begin{cases}0, & \text{if } t_{\mathrm{st}}^{\mathrm{L},p}(t)\leq t+\bar{\tau},\\ \underline{x}_2^{\mathrm{L}}-\underline{e}^{\mathrm{L}}\left(\underline{x}_2^{\mathrm{L}}\right)\bar{\tau}, & \text{if } t_{\mathrm{st}}^{\mathrm{L},p}(t)>t+\bar{\tau},\end{cases} \tag{6.37}$$

and

$$s_{\bar{\tau}}^{\mathrm{L}}\left(t,\underline{x}^{\mathrm{L}}\right)= \begin{cases}\underline{x}_1^{\mathrm{L}}+\underline{x}_2^{\mathrm{L}}\left(t_{\mathrm{st}}^{\mathrm{L},p}(t)-t\right) & \text{if } t_{\mathrm{st}}^{\mathrm{L},p}(t)\leq t+\bar{\tau},\\ \quad -\frac{1}{2}\underline{e}^{\mathrm{L}}\left(\underline{x}_2^{\mathrm{L}}\right)\left(t_{\mathrm{st}}^{\mathrm{L},p}(t)-t\right)^2, & \\ \underline{x}_1^{\mathrm{L}}+\underline{x}_2^{\mathrm{L}}\bar{\tau}-\frac{1}{2}\underline{e}^{\mathrm{L}}\left(\underline{x}_2^{\mathrm{L}}\right)\bar{\tau}^2, & \text{if } t_{\mathrm{st}}^{\mathrm{L},p}(t)>t+\bar{\tau}.\end{cases} \tag{6.38}$$

Quantities (6.37) and (6.38) represent the velocity and the position of the leader RLP at time $t+\bar{\tau}$ under a braking with controller $\hat{K}^{\mathrm{L}}$ starting at time t from the leader state $x^{\mathrm{L}}(t)$ and a time duration $\bar{\tau}$.

We also define $V_{\bar{\tau}}^{\mathrm{line},*}(t,x_1^{\mathrm{F}},\underline{x}^{\mathrm{L}})=\min_{s\in\left[x_1^{\mathrm{F}},s_{\bar{\tau}}^{\mathrm{L}}\left(t,\underline{x}^{\mathrm{L}}\right)\right]}V^{\mathrm{line}}(s)$ and

$$v_{\bar{\tau}}^{\max,\mathrm{F}}\left(t,x^{\mathrm{F}},\underline{x}^{\mathrm{L}}\right)=\min\left\{x_2^{\mathrm{F}}+\overline{a}_{\mathrm{dr}}^{\mathrm{F}}\Delta T,V^{\max,\mathrm{F}},V_{\bar{\tau}}^{\mathrm{line},*}(t,x_1^{\mathrm{F}},\underline{x}^{\mathrm{L}})\right\}, \tag{6.39}$$

where $\Delta T=t_{\mathrm{last}}+H-\bar{\tau}$, where we remember t_{last} being an output of Algoritm 4.

The term (6.39) is an upper bound on the maximum velocity the follower can assume along the railway interval $\left[x_1^{\mathrm{F}},s_{\bar{\tau}}^{\mathrm{L}}\left(t,\underline{x}^{\mathrm{L}}\right)\right]$.

Furthermore, since the state trajectory of the leader RLP is available up to time $t_{\mathrm{last}}+H$, the follower can compute the leader's average velocity

$$v_{\mathrm{ave}}^{\mathrm{L}}(t)=\int_t^{t+\bar{\tau}}\underline{x}_2^{\mathrm{L}}(r)\,dr. \tag{6.40}$$

In view of the (6.37)-(6.40) we can define

$$\begin{aligned}\tilde{z}_{\bar{\tau}}^{\mathrm{F},1}(t,\underline{x}^{\mathrm{L}}(t),x^{\mathrm{F}}(t))=-\Bigg[&L^d+d-\left(s_{\bar{\tau}}^{\mathrm{L}}(t,\underline{x}^{\mathrm{L}}(t))-v_{\mathrm{ave}}^{\mathrm{L}}(t)\bar{\tau}\right)\\ &-\frac{1}{2}\left(\frac{{v_{\bar{\tau}}^{\mathrm{L}}}^2(t,\underline{x}^{\mathrm{L}}(t))}{\underline{e}^{\mathrm{L}}(\underline{x}_2^{\mathrm{L}}(t))}-\frac{{v_{\bar{\tau}}^{\max,\mathrm{F}}}^2\left(t,x^{\mathrm{F}},\underline{x}^{\mathrm{L}}\right)}{\overline{e}^{\mathrm{F}}(x^{\mathrm{F}}(t))}\right)\Bigg],\end{aligned}$$

$$\tilde{z}_{\bar{\tau}}^{\mathrm{F},2}(t,\underline{x}^{\mathrm{L}}(t))=-\left[L^d+d-\left(s_{\bar{\tau}}^{\mathrm{L}}(t,\underline{x}^{\mathrm{L}}(t))-v_{\mathrm{ave}}^{\mathrm{L}}(t)\bar{\tau}\right)\right],$$

$$\tilde{z}^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t)) = \min\left\{\tilde{z}^{\mathrm{F},1}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t)), \tilde{z}^{\mathrm{F},2}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t))\right\} \tag{6.41}$$

and

$$\tilde{z}^{\mathrm{FL}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t)) = \underline{x}^{\mathrm{L}} - \tilde{z}^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t)). \tag{6.42}$$

The above (6.42) is therefore exploited in what follow to define

$$z^{\mathrm{FL}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_0) = \begin{cases} \max_{t' \in [t_0, t]} \tilde{z}^{\mathrm{FL}}_{\bar{\tau}}(t', \underline{x}^{\mathrm{L}}(t'), x^{\mathrm{F}}(t')) & \text{if } t - t_0 \leq \bar{\tau}, \\ \max_{t' \in [t-\bar{\tau}, t]} \tilde{z}^{\mathrm{FL}}_{\tau}(t', \underline{x}^{\mathrm{L}}(t'), x^{\mathrm{F}}(t')) & \text{if } t - t_0 > \bar{\tau}, \end{cases} \tag{6.43}$$

where t_0 is a generic time that will be assigned later.

Finally, we define

$$z^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}_1(t), t_0) = \underline{x}^{\mathrm{L}}_1(t) - z^{\mathrm{FL}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_0). \tag{6.44}$$

The above formulas allow to run Algorithm 7 which provides the reference trajectory $\mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\text{last}})$ for the follower predictive controller. The algorithm is initialized with $t_{\text{last}} = -\infty$ and $\mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\text{last}})$ null trajectory.

Algorithm 7 Position reference generator. Output: $\mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\text{last}})$.

1: **loop**
2: Listen to possible transmission of information from the leader;
3: **if** A packet $\left(t_{\kappa^{\mathrm{L}}}, x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}}), \underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})\right)$ is received with $t_{\kappa^{\mathrm{L}}} > t_{\text{last}}$ **then**
4: $t_{\text{last}} \leftarrow t_{\kappa^{\mathrm{L}}}$
5: Compute

$$\mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\text{last}}) = z^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\text{last}})_{t \in [t_{\text{last}}, t_{\text{last}} + H - \bar{\tau})}, \tag{6.45}$$

with $z^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\text{last}})$ computed according to (6.44) with the help of (6.37)-(6.43).

Let us denote with $\mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\text{last}})[t]$ the value of trajectory $\mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\text{last}})[t]$ at time t. The following theorem formalizes the role of reference $\mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\text{last}})$.

Theorem 6.6.2. *Consider the trains system of form* (6.4) *with* $i \in \{\mathrm{L}, \mathrm{F}\}$ *and the safe set* $\mathcal{S}(x) = \{x : b(x) \leq 0\}$, *with* $b(x)$ *defined in* (6.15c). *Suppose the system starts in a safe state, that is* $[x^{\mathrm{L}^T}(t_0), x^{\mathrm{F}^T}(t_0)]^T \in \mathcal{S}$. *Consider the control architecture encompassing the Safety control block (Algorithm 5) and the Position reference generator (Algorithm 7), with* $\bar{\tau}$ *satisfying* (6.36). *Then, any valid predictive controller* $K^{\mathrm{F}}(x^{\mathrm{F}})$ *able to enforce*

$$x^{\mathrm{F}}_1(t) \leq \mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\text{last}})[t], \forall t \in [t_{\text{last}}, t_{\text{last}} + H - \bar{\tau}) \tag{6.46}$$

guarantees that at least a $\bar{\tau}$ is wait before an emergency braking is triggered, that is $t_{\kappa^{\mathrm{E}}+1} - t_{\kappa^{\mathrm{E}}} \geq \bar{\tau}$ for any κ^{E}.

Proof. To prove the result let us assume that the follower's controller plans a path forward such that, at any time t and for the time interval $[t, t+\bar{\tau}]$ it keeps an average velocity as $v^{\mathrm{L}}_{\mathrm{ave}}(t)$ in (6.40). We are therefore supposing that, roughly speaking, the follower controller is able to track the RLP leader average velocity computed over a $\bar{\tau}$ ahead time interval. This hypothesis will not necessarily be met and, in general, it is not satisfied. Nevertheless, let us keep it as working hypothesis for now (we will then address later the case in which such hypothesis is not met). Saying this, even if the follower keeps $v^{\mathrm{L}}_{\mathrm{ave}}(t)$ as average velocity in $[t, t+\bar{\tau}]$ the instant velocity could vary. In particular, we can consider the worst case were the follower assumes, at time $t+\bar{\tau}$, the instant velocity of $v^{\max,\mathrm{F}}_{\bar{\tau}}(t, x^{\mathrm{F}}, \underline{x}^{\mathrm{L}})$ (that is, an upper bound computed at time t on the maximum velocity achievable by the follower). In view of this, $\tilde{z}^{\mathrm{F},1}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t))$ represents the position that the follower should have, at time t, such that when proceeding at the planned RLP leader average velocity and supposing the leader is instead activating an emergency braking, at time $t+\bar{\tau}$ and with the (worst) case of a follower velocity (we omit the dependency for the sake of brevity) $v^{\max,\mathrm{F}}_{\bar{\tau}}$ the barrier $b'(s^{\mathrm{L}}_{\bar{\tau}}, v^{\mathrm{L}}_{\bar{\tau}}, \tilde{z}^{\mathrm{F},1}_{\bar{\tau}} + v^{\mathrm{L}}_{\mathrm{ave}}\bar{\tau}, v^{\max,\mathrm{F}}_{\bar{\tau}}) = 0$. In the latter formula, we slight abuse the notation and explicitly point out (not in a stack vector form) the variables upon which $b'(\cdot)$ in (6.15a) depends. Specifically, $s^{\mathrm{L}}_{\bar{\tau}}\left(t, \underline{x}^{\mathrm{L}}\right)$ in (6.38) (for brevity, we omit the dependencies) is, as said, the position of the leader RLP at time $t+\bar{\tau}$ after a braking with controller $\hat{K}^{\mathrm{L}}$. Such a position is a lower bound for that achieved by the true leader after an emergency braking (notice that $\hat{K}^{\mathrm{L}}$ is more aggressive than K^{L}_{E}). Analogously, $v^{\mathrm{L}}_{\bar{\tau}}\left(t, \underline{x}^{\mathrm{L}}\right)$ in (6.37) is the achieved RLP velocity.

As for $\tilde{z}^{\mathrm{F},1}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t))$, the term $\tilde{z}^{\mathrm{F},2}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t))$ is the follower position such that $b''(s^{\mathrm{L}}_{\bar{\tau}}, v^{\mathrm{L}}_{\bar{\tau}}, \tilde{z}^{\mathrm{F},2}_{\bar{\tau}} + v^{\mathrm{L}}_{\mathrm{ave}}\bar{\tau}, v^{\max,\mathrm{F}}_{\bar{\tau}}) = 0$ at time $t+\bar{\tau}$, with $b''(\cdot)$ defined in (6.15b).

The term $\tilde{z}^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t))$ represents, in view of what stated above, the position the follower should have such that, proceeding at an average velocity $v^{\mathrm{L}}_{\mathrm{ave}}(t)$ for the time interval $[t, t+\bar{\tau}]$ and considering instead a leader emergency braking, the barrier $b(s^{\mathrm{L}}_{\bar{\tau}}, v^{\mathrm{L}}_{\bar{\tau}}, \tilde{z}^{\mathrm{F}}_{\bar{\tau}} + v^{\mathrm{L}}_{\mathrm{ave}}\bar{\tau}, v^{\max,\mathrm{F}}_{\bar{\tau}})$ will assume null value a time $t+\bar{\tau}$, with $b(\cdot)$ given in (6.15c).

Therefore, any controller of the follower able to guarantee

$$x^{\mathrm{F}}_1(t) \leq \tilde{z}^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t)), \tag{6.47}$$

and an average velocity $v^{\mathrm{L}}_{\mathrm{ave}}(t)$ guarantees that the CBF is not intercepted before a time $t + \bar{\tau}$. Since this is valid for any time t, keeping the constraint (6.47) guarantees that, at any t_{last} upon which the Leader RLP predictor of Algorithm 4 is reset, an emergency braking event does not happen before time $t + \bar{\tau}$.

The reasoning conducted so far is based on the hypothesis that the follower controller is able not only to satisfy (6.47), but also to keep an average velocity $v^{\mathrm{L}}_{\mathrm{ave}}(t)$ for the time interval $[t, t + \bar{\tau}]$. This, in general, cannot be assumed. Nevertheless, it is immediate to notice that if the follower has an average velocity $\tilde{v}$ lower than $v^{\mathrm{L}}_{\mathrm{ave}}(t)$ the whole reasoning still holds. Indeed, the follower would reach, at time $t + \bar{\tau}$, a position $\tilde{x}^{\mathrm{F}}(t + \bar{\tau}) = x^{\mathrm{F}}(t) + \tilde{v}\bar{\tau}$ with $\tilde{x}^{\mathrm{F}}(t + \bar{\tau}) \leq x^{\mathrm{F}}(t) + v^{\mathrm{L}}_{\mathrm{ave}}(t)\bar{\tau}$ therefore attaining a more conservative barrier value $b(s^{\mathrm{L}}_{\bar{\tau}}, v^{\mathrm{L}}_{\bar{\tau}}, \tilde{x}^{\mathrm{F}}(t + \bar{\tau}), v^{\max,F}_{\bar{\tau}}) \leq b(s^{\mathrm{L}}_{\bar{\tau}}, v^{\mathrm{L}}_{\bar{\tau}}, \tilde{z}^{\mathrm{F}}_{\bar{\tau}} + v^{\mathrm{L}}_{\mathrm{ave}}\bar{\tau}, v^{\max,F}_{\bar{\tau}})$.

To cope instead with an average velocity greater than $v^{\mathrm{L}}_{\mathrm{ave}}(t)$, we first need to observe that this might happen when the leader-follower inter distance $\tilde{z}^{\mathrm{FL}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t))$ in (6.42) is decreasing. It might therefore happen then, when tracking $x^{\mathrm{F}}_1(t) \leq \tilde{z}^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t))$, the follower average velocity might grow bigger than $v^{\mathrm{L}}_{\mathrm{ave}}(t)$, violating the hypothesis on which the theorem is valid. This aspect can be easily fixed by replacing constraint (6.47) with the more conservative

$$x^{\mathrm{F}}_1(t) \leq z^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\mathrm{last}}), \tag{6.48}$$

with $z^{\mathrm{F}}_{\bar{\tau}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\mathrm{last}})$ defined in (6.44). □

6.6.5 Cruise virtual coupling block: switched controller

Section 6.6.4 does not explicitly state the expression of the predictive controller to be applied for the follower. Rather, it provides, through Algorithm 7 the trajectory constraint $\mathbf{z}^{\mathrm{F}}_{\bar{\tau}}(t_{\mathrm{last}})$ generated at any update event t_{last} which, if enforced by the predictive controller on the follower position, allows no to have any transmission before a $\bar{\tau}$ time interval is passed.

Since $\bar{\tau}$ is a design value, whose choice might follow a criterion as that depicted at the beginning of Section 6.6.4, it is in principle possible to consider several values of $\bar{\tau}$ so as to have several $\bar{\tau}$-distances levels and develop switched controllers.

In this section we propose a three-level predictive switched controller considering three $\tau_i = \omega_i\bar{\tau}$ values, with $i \in \{1, 2, 3\}$, with $1 < \omega_1 < \omega_2 < \omega_3$ scaling parameters. The whole reasoning conducted in Section 6.6.4 can therefore

be repeated substituting in the formulas $\bar{\tau}$ with τ_i.

The switched controller is therefore implemented in Algorithm 8, where synthetically we call $K_i^{\mathrm{F}}(x^{\mathrm{F}})$, with $i \in \{1,2,3\}$ the three controller modes. Notice that the trajectories (6.49)–(6.51) are used to switch among the different control modes and used by the controller not as hard constraint, but in a soft way (included in the optimization function). Indeed, since the minimum time before an emergency brake is possibly triggered is a desired performance index which does not affect safety, we prefer to include it in the optimization function together with other control effort costs. Simulations later provided confirm the validity of our choice. Notice also that other choice, such as considering the position reference trajectories (6.49) — (6.51) as hard constraints is also possible and can be seamlessly included in our control framework.

Algorithm 8 Switched controller. Output: $K_i^{\mathrm{F}}(x^{\mathrm{F}}(t))$.

loop
  Listen to possible transmission of information from the leader;
  if A packet $\left(t_{\kappa^{\mathrm{L}}}, x^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}}), \underline{\mathbf{x}}^{\mathrm{L}}(t_{\kappa^{\mathrm{L}}})\right)$ is received with $t_{\kappa^{\mathrm{L}}} > t_{\text{last}}$ **then**
    $t_{\text{last}} \leftarrow t_{\kappa^{\mathrm{L}}}$
    Compute

$$\mathbf{z}_{\tau_1}^{\mathrm{F}}(t_{\text{last}}) = z_{\tau_1}^{\mathrm{F}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\text{last}})_{t \in [t_{\text{last}},\, t_{\text{last}}+H-\tau_1)}, \tag{6.49}$$

$$\mathbf{z}_{\tau_2}^{\mathrm{F}}(t_{\text{last}}) = z_{\tau_2}^{\mathrm{F}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\text{last}})_{t \in [t_{\text{last}},\, t_{\text{last}}+H-\tau_2)}, \tag{6.50}$$

$$\mathbf{z}_{\tau_3}^{\mathrm{F}}(t_{\text{last}}) = z_{\tau_3}^{\mathrm{F}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\text{last}})_{t \in [t_{\text{last}},\, t_{\text{last}}+H-\tau_3)}. \tag{6.51}$$

    if $x_1^{\mathrm{F}}(t) \leq z_{\tau_3}^{\mathrm{F}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\text{last}})$ **then**
      Select controller $K_3^{\mathrm{F}}(x^{\mathrm{F}}(t))$;
    else if $z_{\tau_3}^{\mathrm{F}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\text{last}}) < x_1^{\mathrm{F}}(t) \leq z_{\tau_1}^{\mathrm{F}}(t, \underline{x}^{\mathrm{L}}(t), x^{\mathrm{F}}(t), t_{\text{last}})$ **then**
      Select controller $K_2^{\mathrm{F}}(x^{\mathrm{F}}(t))$;
    else
      Select controller $K_1^{\mathrm{F}}(x^{\mathrm{F}}(t))$;

The controller $K_1^{\mathrm{F}}(x^{\mathrm{F}}(t))$ is deployed in scenarios where the VC approaches the critical condition when the follower is reaching the safety barrier. The primary objective of this controller is to ensure safety by maximizing the distance from the safety boundary while also minimizing control effort. Its cost function is defined

as follows

$$K_1^{\mathrm{F}}(x^{\mathrm{F}}(t)) = \arg \min_{x^{\mathrm{F}}(t),u^{\mathrm{F}}(t)} \int_t^{t+H} \Theta_1^u \left(u^{\mathrm{F}}(r)\right)^2 + \Theta_1^s \left(x_1^{\mathrm{F}}(r) - \mathbf{z}_{\tau_2}^{\mathrm{F}}(t_{\mathrm{last}})[r]\right)^2 dr, \quad \text{subject to} \quad \text{Eqs. (6.6)}, \quad \text{with } i = \mathrm{F}. \tag{6.52}$$

The controller $K_2^{\mathrm{F}}(x^{\mathrm{F}}(t))$ is tasked with supporting the VC under normal operating conditions. Its primary objective is to maintain the average velocity of the leader, ensuring smooth and efficient operation. This controller works to minimize deviations from the desired velocity and position, while also optimizing control effort. Its functional cost is defined as follows

$$K_2^{\mathrm{F}}(x^{\mathrm{F}}(t)) = \arg \min_{x^{\mathrm{F}}(t),u^{\mathrm{F}}(t)} \int_t^{t+H} \Theta_2^s (x_1^{\mathrm{F}}(r) - \mathbf{z}_{\tau_2}^{\mathrm{F}}(t_{\mathrm{last}})[r])^2 + \Theta_2^v (x_2^{\mathrm{F}}(r) - \underline{\mathbf{x}}_2^{\mathrm{L}}(r))^2 + \Theta_2^u (u^{\mathrm{F}}(r))^2 dr, \quad \text{subject to} \quad \text{Eqs.(6.6)}, \quad \text{with } i = \mathrm{F}. \tag{6.53}$$

The controller $K_3^{\mathrm{F}}(x^{\mathrm{F}}(t))$, designated as the L3-VC transition controller, is specifically designed to minimize the distance between the two trains by employing an aggressive control strategy to catch up to the leader. This controller is activated at the onset of the VC operation, serving as the initial control mechanism to quickly establish the desired train spacing.. The optimization problem it aims to solve is outlined as follows

$$K_3^{\mathrm{F}}(x^{\mathrm{F}}(t)) = \arg \min_{x^{\mathrm{F}}(t),u^{\mathrm{F}}(t)} \int_t^{t+H} \Theta_3^s (x_1^{\mathrm{F}}(r) - \mathbf{z}_{\tau_2}^{\mathrm{F}}(t_{\mathrm{last}})[r])^2, \quad \text{subject to} \quad \text{Eqs.(6.6)}, \quad \text{with } i = \mathrm{F}. \tag{6.54}$$

6.7 Simulations

To verify the efficacy of our control system, we will utilize a specialized simulation tool named *RV4565*, a video of its functionalities is reported in [100]. This tool, developed with MATLAB enables detailed observation of train variables, including those influencing platoon systems, such as inter-train distance and velocity. All simulations were conducted on a laptop equipped with an Intel i7-10710U processor (6 cores at 4.7GHz) and 16GB of RAM, running the

Windows operating system.

The data used during the simulation for the controller and train models parameters are reported in Table 6.1 and Table 6.2, respectively.

Parameter	Value	Parameter	Value
H(s)	12	ΔT^{mpc}(s)	0.06
ω_1	1	ω_2	3
ω_3	6	$T^{\max}$(s)	8
Θ_1^s	2	Θ_1^v	0.225
Θ_1^u	$1 \cdot 10^{-2}$	Θ_2^s	$2 \cdot 10^{-2}$
Θ_2^v	10	Θ_2^u	$5 \cdot 10^{-2}$
Θ_3^s	1	Θ_3^u	$4 \cdot 10^{-3}$
d(m)	1000	p_{loss}	0.01

Table 6.1: Controller parameters.

Parameter	Value	Parameter	Value
M^{L}(kg)	$4.9 \cdot 10^5$	M^{F}(kg)	$5.1 \cdot 10^5$
γ	10^6	L^{L}(m)	202
A^{L}(N)	27.30	A^{F}(N)	26.15
$B^{\text{L}}(\frac{\text{Nm}}{\text{s}})$	9.5	$B^{\text{F}}(\frac{\text{Nm}}{\text{s}})$	8.36
$C^{\text{L}}(\frac{\text{Nm}^2}{\text{s}^2})$	2.05	$C^{\text{F}}(\frac{\text{Nm}^2}{\text{s}^2})$	1.92
$a_{\text{br}}^{\text{L}}(\frac{\text{m}}{\text{s}^2})$	-0.65	$a_{\text{br}}^{\text{F}}(\frac{\text{m}}{\text{s}^2})$	-0.65
$a_{\text{dr}}^{\text{L}}(\frac{\text{m}}{\text{s}^2})$	1	$a_{\text{dr}}^{\text{F}}(\frac{\text{m}}{\text{s}^2})$	1
P_{br}^{L}(W)	$-30.82 \cdot 10^4$	P_{br}^{F}(W)	$-30.82 \cdot 10^4$
P_{dr}^{L}(W)	$30.82 \cdot 10^4$	P_{dr}^{F}(W)	$30.82 \cdot 10^4$
$\underline{M}$(kg)	$4.5 \cdot 10^5$	$\overline{M}$(kg)	$5.5 \cdot 10^5$
$\underline{A}$(N)	25.11	$\overline{A}$(N)	28.24
$\underline{B}(\frac{\text{Nm}}{\text{s}})$	8.02	$\overline{B}(\frac{\text{Nm}}{\text{s}})$	10.12
$\underline{C}(\frac{\text{Nm}^2}{\text{s}^2})$	1.88	$\overline{C}(\frac{\text{Nm}^2}{\text{s}^2})$	2.11
$\underline{a}_{\text{br}}(\frac{\text{m}}{\text{s}^2})$	-0.70	$\overline{a}_{\text{br}}(\frac{\text{m}}{\text{s}^2})$	-0.59
$\underline{a}_{\text{dr}}(\frac{\text{m}}{\text{s}^2})$	0.92	$\overline{a}_{\text{dr}}(\frac{\text{m}}{\text{s}^2})$	1.08
$\underline{P}_{\text{br}}$(W)	$-33.28 \cdot 10^4$	$\overline{P}_{\text{br}}$(W)	$-30 \cdot 10^4$
$\underline{P}_{\text{dr}}$(W)	$30 \cdot 10^4$	$\overline{P}_{\text{dr}}$(W)	$33.28 \cdot 10^4$
$V^{\max,\text{F}}(\frac{\text{m}}{\text{s}})$	83	$T^{\max}$(s)	7

Table 6.2: Trains parameters.

The controllers (6.52)–(6.54) presented in 6.6.5 have been implemented as

NMPCs using the OpEn framework and the PANOC solver [101], the parameters used to configure the controller are listed in Table 6.1.

The dynamic parameters of the trains were modeled after those of two Frecciarossa ETR1000 trains [102]. Furthermore, it is assumed that the leader transmits messages at a synchronous sampling interval of 1s. For the simulations the model parameters are reported in Table 6.2, it was specifically assumed that the follower train is fully loaded, while the leader train is not.

Figure 6.3 and 6.4 reports the data profile of the railway line used to conduct the simulation is the Italian Firenze-Bologna railway line. The maximum velocity of the railway line is assumed to be constant and is set to $V^{\text{line}}(s) = 83\frac{\text{m}}{\text{s}}$ which is the maximum permitted by the infrastructure along the entire route.

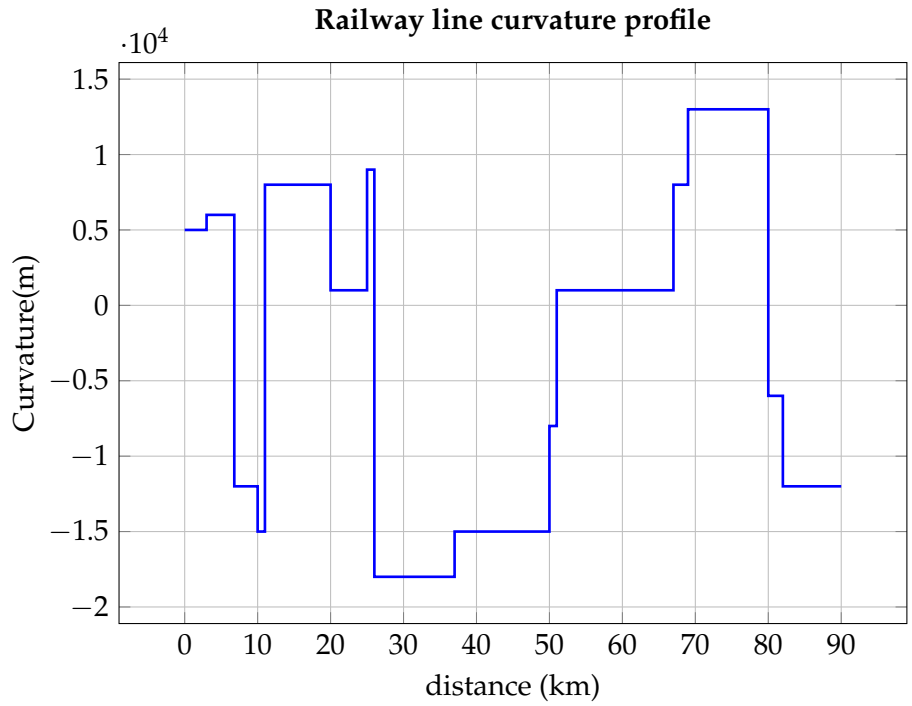


Figure 6.3: Curvature profile of the Firenze-Bologna railway line, showing the variation of curvature over the track distance.

6.7.1 Operational scenarios

Two different operational scenarios Operational Scenario (OS) were considered to evaluate the safety and benefits of the control system proposed in this study.

In the first scenario, referred to as OS1, the leader and follower initiate a virtual coupling (VC) operation, starting from a L3 with a separation distance

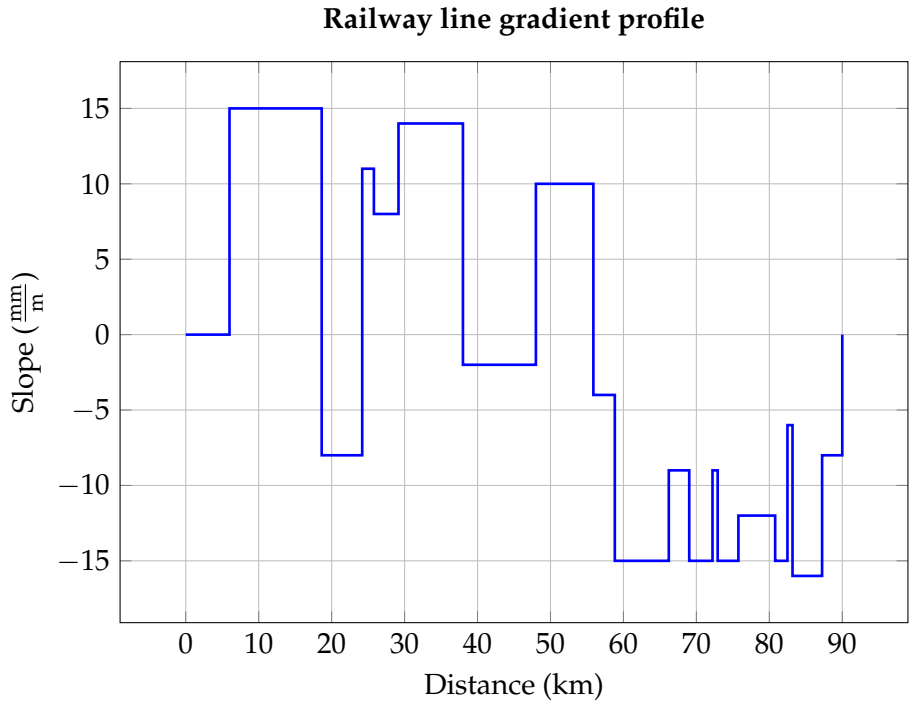


Figure 6.4: Gradient profile of the Firenze-Bologna railway line, illustrating the slope changes along the track distance.

of 6000m. Throughout the simulation, the leader adjusts its velocity dynamically, introducing velocity variations along the route. At one stage, the leader stabilizes its velocity and maintains a constant velocity. Despite these fluctuations, the follower successfully closes the gap and establishes the virtual coupling. As the scenario advances, the leader unexpectedly begins a braking maneuver, gradually decelerating until coming to a complete stop. This scenario is designed to assess the control system's ability to handle dynamic velocity variations while ensuring precise coordination between the leader and follower. It particularly evaluates the follower's capacity to react effectively and safely to sudden changes in the leader's behavior, such as abrupt braking.

The second scenario, designated as OS2, begins with the leader and follower engaged in a VC operation where the leader maintains a steady, constant velocity, and the follower ensures a safe following distance. In this scenario, the focus shifts to a situation where communication between the leader and follower starts to degrade, leading to increased delays in the transmission of control data. As the scenario unfolds, the communication further deteriorates until updates from the leader fail to arrive on time. This delay poses a critical challenge, testing the

robustness of the follower's control system. Without real-time data, the follower must rely on the last received information to ensure safety, maintain proper spacing, and continue operations despite the communication breakdown. This scenario emphasizes the importance of the Delay estimator block, as described in Subsection 6.6.3, and the role of adaptive control strategies in maintaining operational stability under unexpected communication failures. The control system's ability to handle these disruptions without compromising safety is a key focus of this evaluation.

The two operational scenarios, OS1 and OS2, represent the core situations encountered in railway operations, highlighting the control system's critical role in ensuring safety and efficiency. These scenarios underscore the system's ability to manage the most fundamental and challenging aspects of railway operations, ensuring reliability in both routine and disrupted conditions.

6.7.2 Results

The simulation results for scenarios OS1 and OS2 are shown in Figures 6.5a–6.5h and Figures 6.6a–6.6h, respectively (for the sake of readability are reported at the end of this Section). For clarity, time dependencies of the variables in the legends have been removed, and the x-axis unit, expressed in seconds, is defined here.

In the first scenario (Figure 6.5b), the follower initially starts far from the leader. As a result, the control system selects the controller K_3^{F} (Figure 6.5d), which applies a force (shown in Figure 6.5f) to accelerate the follower to approximately $80\frac{\mathrm{m}}{\mathrm{s}}$, as illustrated in Figure 6.5c. By 126s (Figure 6.5h), the follower's position reaches the region beyond $z_{\tau_1}^{\mathrm{F}}$ due to the high velocity, triggering controller K_1^{F} (Figure 6.5d). This controller decelerates the follower to match the leader's velocity. Around time 600s, the control system stabilizes the follower at a distance of approximately 1300m from the leader, as seen in Figure 6.5b, achieving virtual coupling between the two trains. The latter Figure presents two scenarios: one with a communication delay of 3s and the other with 800ms. It demonstrates that as the communication delay decreases, the distance between the leader and follower also reduces. During the simulation, the leader alters its velocity at 800s and 1600s to $60\frac{\mathrm{m}}{\mathrm{s}}$ and $40\frac{\mathrm{m}}{\mathrm{s}}$, respectively. In both cases, the control system maintains the virtual coupling between the two trains. It is important to note that, up to this point, the safety barrier in Figure 6.5a remains negative.

To test the control system under emergency conditions, the leader initiates a

braking maneuver at 2250s (Figure 6.5b), decelerating to a full stop. As shown in Figures 6.5e and 6.5g, the control system activates the emergency controller since the predicted safety barrier value becomes positive. Notably, in Figure 6.5e, the safety barrier is initially negative when the packet arrives from the leader. Around time 2272s, however, it becomes positive due to two main factors. First, before the emergency controller is triggered, controller K_1^{F} is engaged. Since K_1^{F} is not particularly aggressive, the safety barrier gradually increases until the emergency controller activates. Second, the Leader RLP predictor, described in Section 6.6.1, updates the leader's state during the subsequent period without communication, thereby triggering the first condition in Algorithm 5. It is important to note, however, that the true safety barrier to consider is the one depicted by crosses; the barrier that becomes positive is the robust safety barrier (Figure 6.5e). In this emergency situation, the control system ensures safety by stopping both trains at a distance greater than the safety threshold, as confirmed in Figures 6.5a and 6.5b. Moreover, the latter figure presents two cases: one with a delay of 3s and another with 800ms, from which it can be observed that the interdistance between the two trains decreases as the communication delay decreases.

The second operational scenario OS2 starts with the two trains already in a VC operation, traveling at the same velocity of $50\frac{\mathrm{m}}{\mathrm{s}}$ (Figure 6.6c) and maintaining a separation of approximately 1260m (Figure 6.6b). During this phase, the control system uses controller K_2^{F} to maintain the virtual coupling, as shown in Figure 6.6d. At 500s, the communication channel degrades as reported in Figure 6.6h. The system's behavior is evaluated both with and without the Delay estimator block (Figure legends specify cases without the Delay Estimator block), as described in Section 6.6.3. Figure 6.6g shows that when the Delay estimator is active, the separation increases due to the calculation of reference trajectories $z_{\tau_1}^{\mathrm{F}}$, $z_{\tau_2}^{\mathrm{F}}$, and $z_{\tau_3}^{\mathrm{F}}$ based on the estimated delay $\bar{\tau}$ which dynamically changes. Without the Delay estimator, the control system does not detect any communication degradation and behaves non-adaptively. Moreover, it is possible to note that when the Delay estimator block is not used the frequency of emergency braking activations increase (Figure 6.6d).

Finally, at 1200s, communication between the two trains is lost, as seen in Figure 6.6e, where packet receptions occur at extended intervals, causing the estimated value of the safety barrier to become positive. As in the previous scenario, the Leader RLP predictor triggers the emergency controller. This

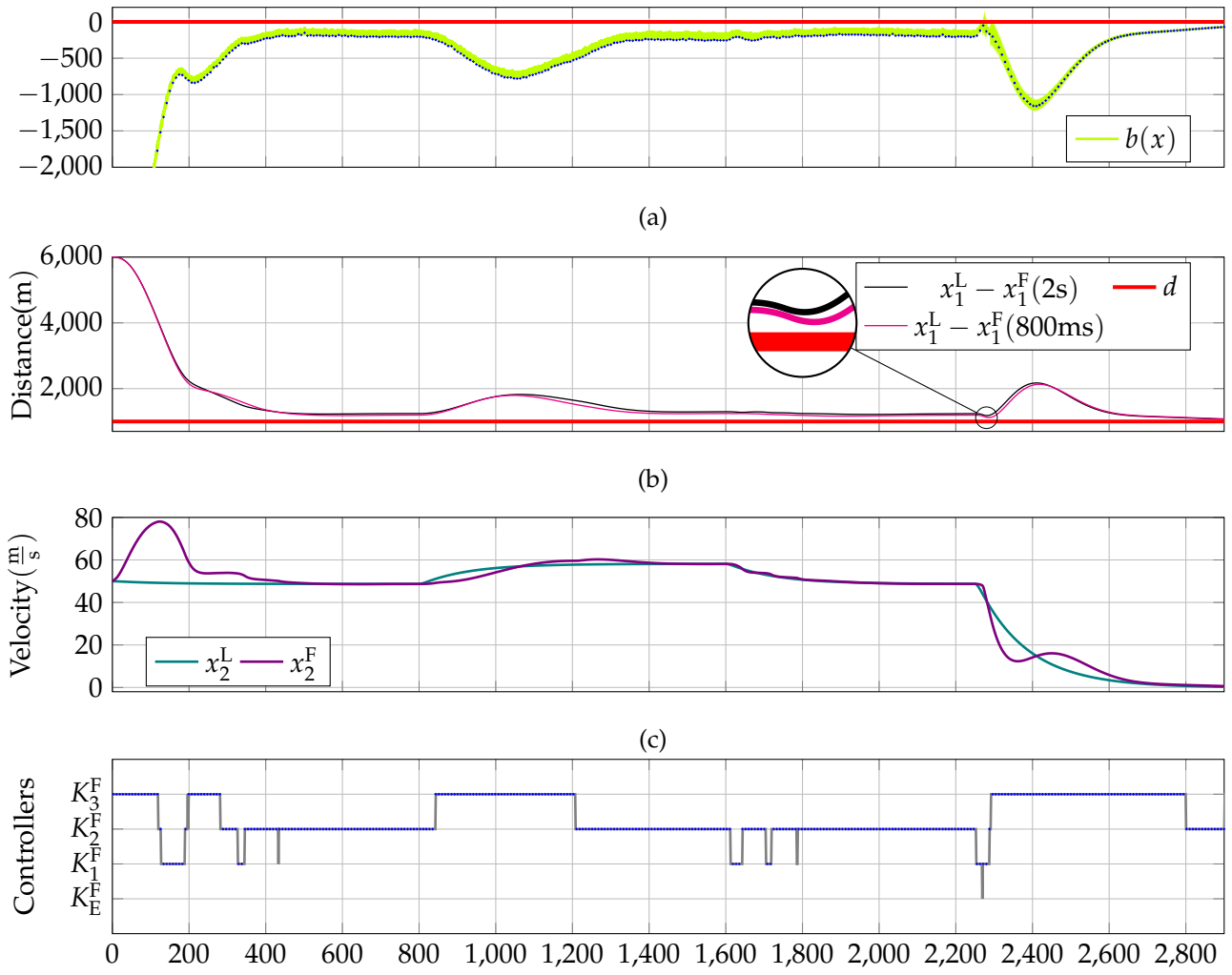
emergency braking is evident in Figure 6.6f, where a large negative force is applied by controller $K_{\mathrm{E}}^{\mathrm{F}}$, which is activated, as shown in Figure 6.6d.

Once communication with the leader is restored, the control system recalculates the safety barrier and verifies that the trains are safe. Controller K_3^{F} is reactivated to gradually recover the lost space during the emergency braking until the VC is engaged again. Furthermore, the absence of the Delay Estimator block (Figure 6.6d) reveals that, even after the re-engagement of the VC, emergency braking activations continue to occur.

To demonstrate the real-time capabilities of the control system architecture described in our work, Table 6.3 presents the statistical computational times of the follower control system architecture proposed in this work. The values obtained indicate that the system can be effectively utilized in railway applications, where the controller's frequency rate is typically around 100ms.

Table 6.3: Computation time for the follower control system.

Op. scenario	**Mean (**ms**)**	**Variance (**ms^2**)**	**Max. time (**ms**)**
OS1	0.16	0.002	5.53
OS2	0.07	0.07	6.88


0
−500
−1,000
−1,500
−2,000
$b(x)$
(a)
Distance(m)
6,000
4,000
2,000
$x_1^{\mathrm{L}} - x_1^{\mathrm{F}}(2\mathrm{s})$
d
$x_1^{\mathrm{L}} - x_1^{\mathrm{F}}(800\mathrm{ms})$
(b)
Velocity($\frac{\mathrm{m}}{\mathrm{s}}$)
80
60
40
20
0
x_2^{L}
x_2^{F}
(c)
Controllers
K_3^{F}
K_2^{F}
K_1^{F}
$K_{\mathrm{E}}^{\mathrm{F}}$
0 200 400 600 800 1,000 1,200 1,400 1,600 1,800 2,000 2,200 2,400 2,600 2,800
(d)

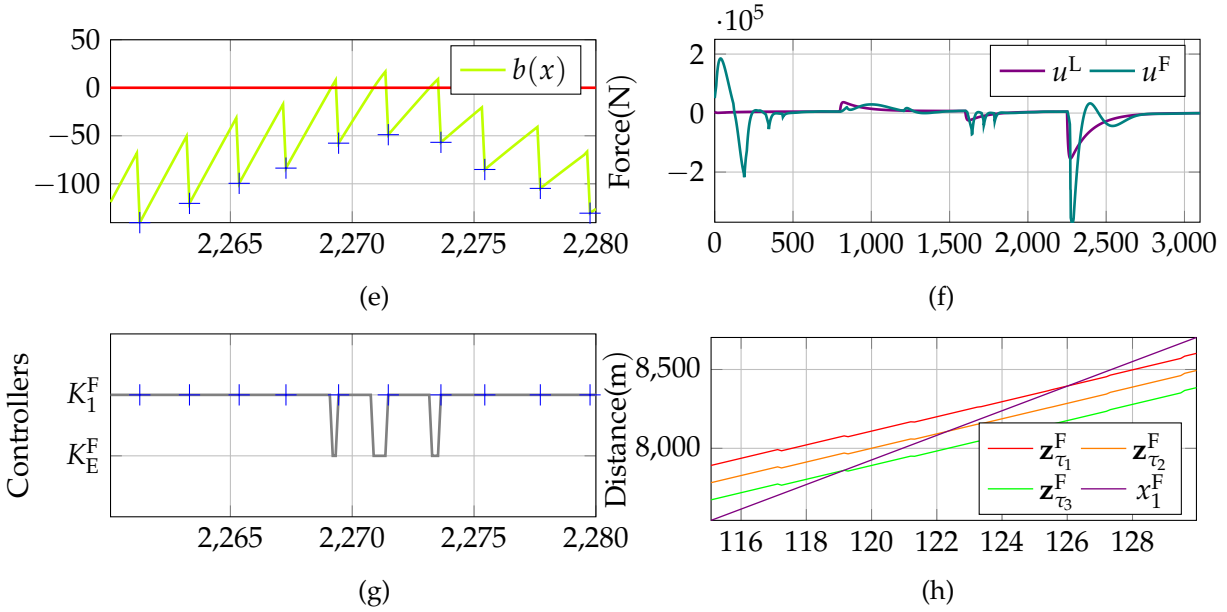
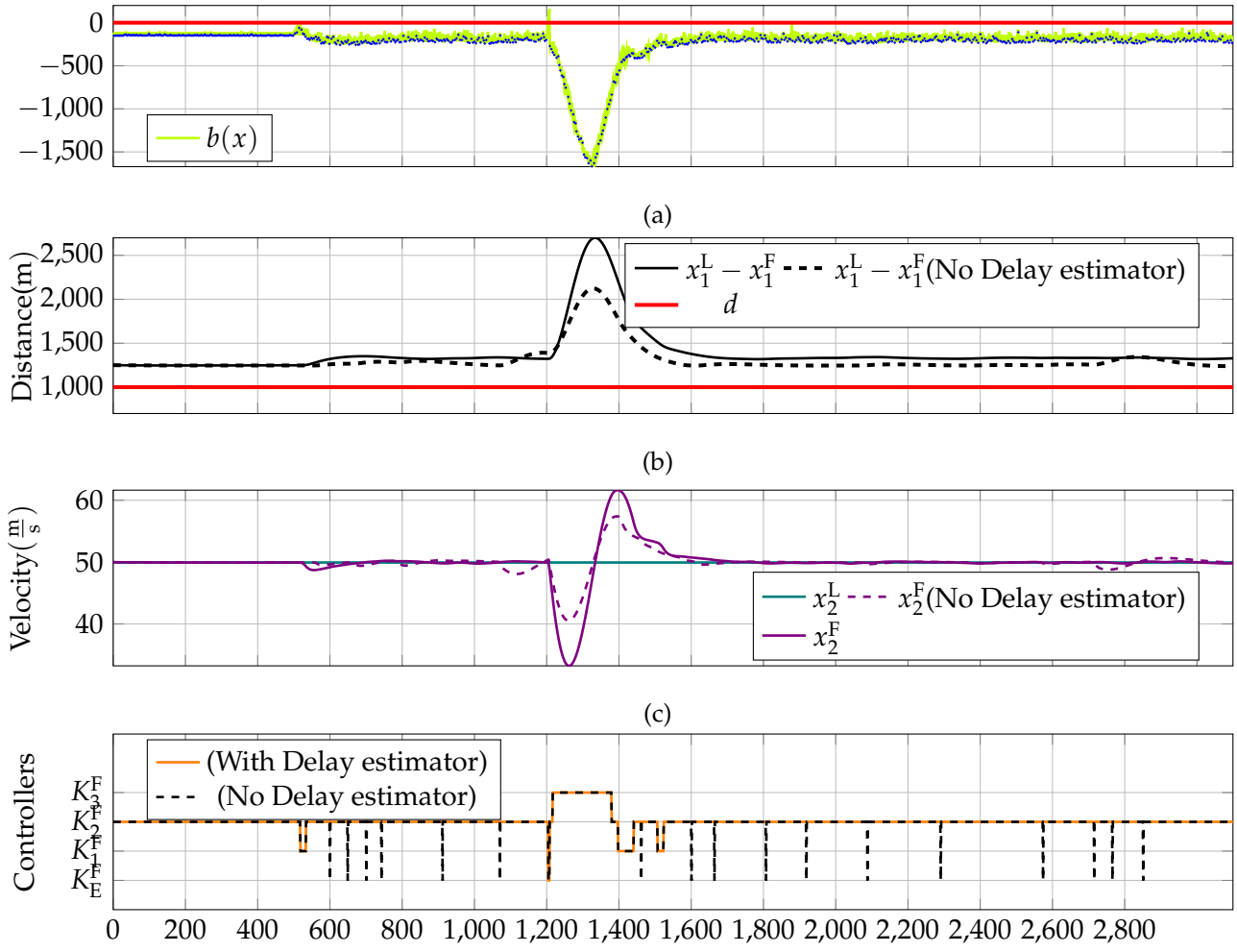


Figure 6.5: Simulation results for operational scenario OS1: (a) shows the safety barrier (lime) and reception events marked by crosses (blue). (b) illustrates the interdistance between the leader and follower with a communication delay of 3s (black) and 800ms (magenta) alongside the safety distance (red). (c) presents the velocities of the leader (teal) and follower (violet). (d) depicts the switching controllers (gray) with reception events again indicated by crosses (blue). (e) and (g) provide zoomed views of subplots (a) and (d), respectively. (f) displays the control inputs for the leader (teal) and follower (violet), while (h) shows the reference trajectories (green, orange, and red) along with the follower's position (violet).


0
−500
−1,000
−1,500
$b(x)$
(a)
Distance(m)
2,500
2,000
1,500
1,000
$x_1^{\mathrm{L}} - x_1^{\mathrm{F}}$
$x_1^{\mathrm{L}} - x_1^{\mathrm{F}}$(No Delay estimator)
d
(b)
Velocity($\frac{\mathrm{m}}{\mathrm{s}}$)
60
50
40
x_2^{L}
x_2^{F}(No Delay estimator)
x_2^{F}
(c)
Controllers
(With Delay estimator)
(No Delay estimator)
K_3^{F}
K_2^{F}
K_1^{F}
$K_{\mathrm{E}}^{\mathrm{F}}$
0 200 400 600 800 1,000 1,200 1,400 1,600 1,800 2,000 2,200 2,400 2,600 2,800
(d)

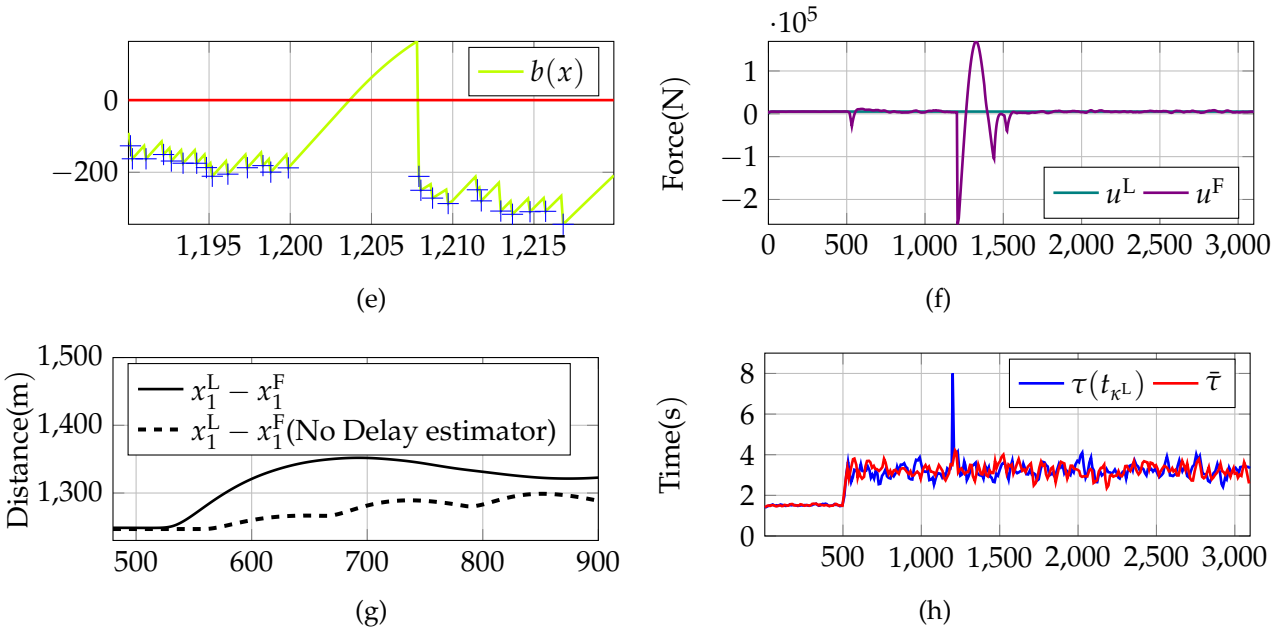


Figure 6.6: Simulation results for operational scenario OS2: (a) shows the safety barrier (lime) and reception events marked by crosses (blue). (b) illustrates the interdistance between the leader and follower, with the dashed black line representing the scenario without the delay estimator block, and the solid black line representing the scenario with the delay estimator, alongside the safety distance (red). (c) presents the velocities of the leader (teal) and follower (violet), the dashed line is the case when Delay estimator block is not used. (d) depicts the switching controllers activations for both cases, with Delay estimator block (orange) and without (black). (e) and (g) provide zoomed views of subplots (a) and (b), respectively. (f) displays the control inputs for the leader (teal) and follower (violet), while (h) shows the elapsed time of follower reception for packets (blue) and estimated communication delay (red).

6.8 Chapter Summary

In this study, we have presented a comprehensive control system architecture for the transition from ERTMS Level 3 to VC operations in railway systems.

The key contributions of our work include the development of a robust control framework that accommodates variabilities in train system parameters and communication throughput. By incorporating a safety control barrier function, we ensure operational safety while minimizing unnecessary emergency braking. Our simulations, conducted using a specialized railway simulation tool, validate the efficacy of our control system on the test case of an Italian railway line and underscore its foundational role in advancing railway safety.

The simulation results confirm the control system's robust capability to maintain virtual coupling between trains under diverse conditions. In both operational scenarios, the system effectively managed acceleration and deceleration, adapting to the leader's velocity changes and ensuring safe distances. It demonstrated resilience by activating appropriate controllers in response to high velocities, communication delays, and emergency braking situations. The system ensures safety even during communication degradations. Overall, the control strategy proved adept at handling dynamic environments, highlighting its potential to enhance safety and efficiency in rail operations through adaptive and responsive control mechanisms.

Since our control architecture is flexible to allow different controllers (still keeping the robustness and the safety features as well as a adjustable time before triggering emergency brake), future work will focus on extending the architecture to handle multiple trains in a platoon and exploring other possible controllers, such as those integrating AI-based solutions to further enhance system performance. Our ongoing research aims to refine the control algorithms and validate the system through field trials, paving the way for the widespread adoption of Virtual Coupling in modern railway networks.

Acknowledgement

We sincerely thank Rete Ferroviaria Italiana for their support through knowledge transfer and background insights, significantly contributing to the advancement of our work.

Conclusions

This thesis presents a comprehensive exploration of critical advancements in perception, learning, and cooperation mechanisms for autonomous vehicles (AVs), addressing pivotal challenges in their development and deployment. Through a multi-faceted approach combining theoretical analysis, algorithm development, and practical validation, the research has made significant contributions to the fields of UAV path following, real-time lane detection, pedestrian collision avoidance, and cybersecurity for AVs.

Advancements in Perception

The research introduced a novel perception algorithm for UAVs that integrates the pure pursuit method with advanced image processing techniques. This hybrid approach leverages the computational simplicity of the pure pursuit method while significantly enhancing the UAV's environmental perception and path-tracking ability. The algorithm's ability to adapt to varied and complex operational environments demonstrates a critical advancement in autonomous navigation, particularly for applications requiring high precision and computational efficiency, such as search and rescue operations, agricultural monitoring, and surveillance.

Additionally, the development of an Iterative Tree Search (ITS) approach for real-time lane detection marks a significant leap forward for autonomous driving systems. The ITS algorithm's pixel-level analysis streamlines the detection process, making it more time-efficient and suited for embedded systems within AVs. This method represents an evolution towards greater computational efficiency, enabling autonomous vehicles to operate within the stringent temporal constraints of urban and highway driving scenarios.

Enhancements in Learning Algorithms

In the domain of learning algorithms, the thesis presents a Deep Deterministic Policy Gradient (DDPG) method tailored for pedestrian collision avoidance in AVs. This reinforcement learning-based approach optimizes the AV's ability to navigate complex environments where pedestrian behavior is unpredictable. The DDPG algorithm's continuous state and action space learning capabilities allow AVs to make nuanced decisions that significantly reduce collision risks. Through a series of simulations, the algorithm demonstrated robust performance in learning speed and driving policy quality, ensuring operational safety in pedestrian-rich environments.

The research further explores the integration of machine learning and control systems to enhance cybersecurity for AVs. The proposed Informative Model Predictive Scheme (I-MPS) architecture provides a proactive approach to detecting and mitigating cyber-attacks, particularly during critical driving maneuvers such as overtaking. The I-MPS utilizes machine learning algorithms to predict and neutralize cyber-attacks in real-time, preserving the integrity of AV control systems. This system is pivotal in maintaining safety and performance standards, even in the face of sophisticated cyber-attacks, thereby addressing a significant concern as AVs become more interconnected.

Advancements in Cooperation

A key highlight of the thesis is the development of a robust control system architecture to facilitate the transition from ETCS Level 3 to Virtual Coupling operations in railway systems. The proposed system effectively manages the dynamic interaction between trains, ensuring safety and enhancing operational efficiency. By incorporating a safety control barrier function, the control architecture ensures operational safety while minimizing unnecessary emergency braking. The simulations validate the efficacy of the control system, underscoring its foundational role in advancing railway safety.

6.9 Broader Implications and Future Directions

The contributions detailed in this thesis are instrumental in advancing the state of AV technology. The enhanced path tracking algorithm for UAVs, the

ITS algorithm for lane detection, the DDPG method for pedestrian collision avoidance, and the I-MPS for cybersecurity collectively address critical aspects of AV deployment in urban environments. Moreover, these advancements contribute to the broader discourse on AV reliability and trustworthiness by advocating for a multifaceted approach to AV safety that accommodates both operational unpredictability and digital security threats.

The research underscores the potential of combining perception, learning, and cooperation mechanisms to drive the future of AV systems. As the autonomous vehicle industry continues to evolve, the emphasis on these areas will undoubtedly remain at the forefront, with the potential to redefine transportation as we know it. The insights from this thesis will be vital in shaping the trajectory of AV development, prioritizing resilience, adaptability, and security in an ever-evolving vehicular landscape.

In conclusion, this thesis presents pivotal contributions to the field of autonomous vehicle perception, learning, and control. These innovations not only advance the technical capabilities of AVs but also offer insights into the trajectory of future research and development. By enhancing the safety, reliability, and efficiency of AV systems, this research contributes to the broader goal of integrating autonomous vehicles into everyday life, paving the way for safer, more efficient, and resilient transportation ecosystems.

This comprehensive analysis and the innovative solutions proposed in this thesis underscore the significance of interdisciplinary research in driving technological advancements. As the field of autonomous systems continues to grow, the foundational work presented here will serve as a cornerstone for future developments, ensuring that autonomous vehicles can meet the complex challenges of the modern world.

Part IV

Appendix

Appendix A

Optimization and Optimal Control

A.1 Model Predictive Control

Introduction

Model Predictive Control (MPC) is a sophisticated control strategy used extensively in industrial processes and advanced engineering systems. Unlike traditional control methods, MPC employs a model of the system to predict future behavior and optimize performance over a specified time horizon. This approach allows for the handling of multivariable control problems and the accommodation of constraints on inputs and outputs, making it particularly powerful for complex applications.

Fundamental Principles

At the core of MPC is the use of a dynamic model to forecast future system behavior. This model, typically derived from physical principles or system identification techniques, allows the controller to predict the future states of the system over a specified horizon. The control action is then determined by solving an optimization problem that minimizes a cost function subject to the system constraints.

The optimization problem in MPC can be formulated as follows:

$$\min_{u_0,u_1,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k \right) + x_N^T P x_N$$

subject to:

$$x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots, N-1$$
$$x_k \in \mathcal{X},\ u_k \in \mathcal{U}, \quad k = 0, 1, \ldots, N-1$$

where:

- x_k is the state vector at time step k,
- u_k is the control input vector at time step k,
- Q, R, and P are positive semi-definite weighting matrices,
- f is the system model,
- $\mathcal{X}$ and $\mathcal{U}$ are the sets of allowable states and inputs, respectively,
- N is the prediction horizon.

State Space Model

The system model f is often represented in a state-space form, particularly for linear systems:

$$x_{k+1} = Ax_k + Bu_k + w_k$$
$$y_k = Cx_k + Du_k + v_k$$

where:

- A, B, C, and D are matrices defining the system dynamics,
- w_k and v_k represent process and measurement noise, respectively,
- y_k is the output vector.

For nonlinear systems, the state-space model takes a more general form:

$$x_{k+1} = f(x_k, u_k) + w_k$$
$$y_k = h(x_k, u_k) + v_k$$

where f and h are nonlinear functions that describe the system dynamics and output, respectively.

Constraints

MPC explicitly handles constraints on states and inputs, which can be critical for ensuring safe and feasible operation. These constraints can be expressed as:

$$x_k \in \mathcal{X} = \{x \in \mathbb{R}^n \mid g_x(x) \leq 0\}$$

$$u_k \in \mathcal{U} = \{u \in \mathbb{R}^m \mid g_u(u) \leq 0\}$$

where g_x and g_u are constraint functions that define the allowable regions for states and inputs.

Cost Function

The cost function in MPC typically includes terms for both state deviation and control effort:

$$J = \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k \right) + x_N^T P x_N$$

- The term $x_k^T Q x_k$ penalizes deviations of the state from the desired trajectory.
- The term $u_k^T R u_k$ penalizes the use of control inputs.
- The terminal cost $x_N^T P x_N$ ensures stability and performance over the prediction horizon.

Optimization Problem

The optimization problem solved at each time step in MPC can be summarized as:

$$\begin{aligned} \min_{u_0, u_1, \ldots, u_{N-1}} \quad & \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k \right) + x_N^T P x_N \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots, N-1 \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U}, \quad k = 0, 1, \ldots, N-1 \\ & x_0 = x(t) \end{aligned}$$

where $x(t)$ is the current state of the system.

The solution to this optimization problem yields a sequence of control actions $u_0, u_1, \ldots, u_{N-1}$. However, only the first control action u_0 is implemented. The optimization is then repeated at the next time step with updated state measurements, following the receding horizon strategy.

Nonlinear Model Predictive Control (NMPC)

For systems with nonlinear dynamics, the optimization problem becomes more complex due to the nonlinear functions f and h. The cost function and constraints must be adapted to handle these nonlinearities.

The NMPC problem can be formulated as:

$$\begin{aligned}
\min_{u_0,u_1,\ldots,u_{N-1}} \quad & \sum_{k=0}^{N-1} (l(x_k, u_k)) + l_f(x_N) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots, N-1 \\
& y_k = h(x_k, u_k), \quad k = 0, 1, \ldots, N-1 \\
& x_k \in \mathcal{X},\ u_k \in \mathcal{U}, \quad k = 0, 1, \ldots, N-1 \\
& x_0 = x(t)
\end{aligned}$$

where l and l_f are the running cost and terminal cost functions, respectively.

Solving NMPC Problems

The nonlinear nature of NMPC problems requires specialized numerical techniques for their solution. Common methods include:

1. **Sequential Quadratic Programming (SQP)**: This method approximates the nonlinear problem by a sequence of quadratic programming subproblems, which are easier to solve.

2. **Interior Point Methods**: These methods handle constraints by incorporating them into the objective function through barrier functions, which are then optimized iteratively.

3. **Direct Methods**: These methods discretize the control problem directly, transforming it into a large-scale nonlinear programming problem. Examples include direct collocation and multiple shooting.

NMPC builds on the strengths of linear MPC by effectively managing multivariable systems and constraints. Moreover, NMPC excels in handling systems with strong nonlinearities, offering precise and effective control where linear approximations fall short.

NMPC is extensively utilized in industries characterized by nonlinear behaviors, such as chemical process control, aerospace, automotive systems,

and robotics. For instance, NMPC optimizes control in chemical reactors with highly nonlinear reaction kinetics and in autonomous vehicles with complex, varying dynamics. In aerospace, NMPC enhances flight control systems, while in robotics, it facilitates precise maneuvering in dynamic environments.

However, NMPC faces significant challenges, particularly in computational complexity and the need for precise models. Real-time nonlinear optimization demands substantial computational resources, and model inaccuracies can degrade performance. Addressing these issues requires robust algorithms capable of real-time execution. Future research is directed towards enhancing computational efficiency through advanced algorithms and leveraging machine learning for more accurate models. Integrating NMPC with artificial intelligence and real-time optimization techniques shows promise for expanding its applicability in increasingly complex environments. Innovations in hardware, such as parallel processing and dedicated optimization chips, offer potential solutions to current computational limitations.

Model Predictive Control, including its nonlinear variant, represents a significant advancement in control theory, providing powerful tools for managing complex, multivariable systems with constraints. The ability to predict future behavior and optimize performance in real-time makes MPC and NMPC invaluable across various industries. As computational capabilities advance and new techniques emerge, the effectiveness and scope of MPC are poised to grow, cementing its role as a cornerstone of modern control engineering. NMPC's proficiency in handling nonlinearities and real-time optimization underscores its continued relevance in advancing industrial automation and control systems.

Appendix B

Learning Algorithms

B.1 Support Vector Machines

Support Vector Machines (SVMs) are a set of supervised learning methods used for classification, regression, and outliers detection. The basic SVM takes input data and predicts, for each given input, which of two possible classes the input is a part of, making it a non-probabilistic binary linear classifier.

The core idea behind SVM is to find the hyperplane that best divides a dataset into two classes. Features of the input data are used to plot each data item as a point in n-dimensional space (where n is the number of features), with the value of each feature being the value of a particular coordinate. Then, SVM performs classification by finding the hyperplane that best separates the two classes.

Mathematical Formulation

The decision function of an SVM is defined as:

$$f(x) = \mathbf{w}^T \mathbf{x} + b$$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the feature vector, and b is the bias.

The goal of the SVM algorithm is to find the values of $\mathbf{w}$ and b that maximize the margin between the two classes. This is typically done by solving a convex optimization problem.

The optimization problem for finding $\mathbf{w}$ and b is formulated as:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$

Subject to:

$$y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, ..., n$$

where $y_i \in \{-1, 1\}$ are class labels, C is the regularization parameter, and ξ_i are slack variables.

The Lagrangian dual problem is given by:

$$\max_{\alpha} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x_i}, \mathbf{x_j} \rangle \right]$$

Subject to:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, ..., n$$

After solving the dual problem, $\mathbf{w}$ and b are obtained as follows:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}$$

$$b = y_s - \mathbf{w}^T\mathbf{x_s}$$

for any support vector x_s with $0 < \alpha_s < C$.

The decision function can be expressed as:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}, \mathbf{x_i} \rangle + b$$

B.2 Learning Alghoritms

Deep Deterministic Policy Gradient (DDPG)

When considering learning, one key concept is interaction with the environment, which provides feedback on actions taken to achieve goals. Reinforcement Learning (RL) is a machine learning technique that utilizes this interaction to identify optimal behavior within an environment. The objective in RL is to determine a policy—a mapping from states to actions—that maximizes the total expected reward, which can be accrued in both terminal and intermediate states. RL involves an agent gaining experience with states, actions, state transitions, and rewards to iteratively refine its behavior towards optimality. Unlike other machine learning paradigms, RL systems are evaluated concurrently with the learning process through trial-and-error, balancing exploration and exploitation.

RL employs a sequential decision process formalized as a Markov Decision Process (MDP), defined by a tuple $M = \langle S, A, T, R \rangle$:

- A finite set of states S.
- A finite set of actions A.
- A state transition function $T(s'|s, a)$ giving the probability of transitioning to state s' when action a is performed in state s.
- A reward function R indicating the desirability of state-action transitions, which can be simplified to $R(s, a)$ representing the expected reward for taking action a in state s.

The cumulative reward G_t obtained from time t onwards is defined as:

$$G_t = \sum_{k=t}^{\infty} \gamma^{k-t} R(s_k, a_k)$$

where γ is the discount factor.
Key elements in RL include:

- The policy π, a mapping from states to actions that determines the system's behavior.
- The value function $V^{\pi}(s)$, representing the expected cumulative reward from state s under policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|s_t = s]$$

- The action-value function $Q^{\pi}(s,a)$, representing the expected cumulative reward from state s by taking action a and then following policy π:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t|s_t = s, a_t = a]$$

The Bellman equation for the state-value function V^{π} is:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R(s,a) + \gamma V^{\pi}(s')|s_t = s]$$

Dynamic Programming (DP), Monte Carlo (MC), and Temporal Differences (TD) are three main methods to solve MDPs. DP assumes knowledge of transition and reward functions, MC learns from complete episodes, and TD combines elements of both by updating based on the current estimate and sampling.

In scenarios with large state-action spaces, traditional tabular methods become impractical. This is where deep RL, which combines RL with deep neural networks, becomes valuable. The Deep-Q Network (DQN) algorithm was a groundbreaking advancement, achieving superhuman performance in Atari games by approximating Q-values with neural networks. DQN introduced key innovations like experience replay and fixed Q-targets to stabilize learning.

However, DQN is limited to discrete action spaces. To address this, the Deep Deterministic Policy Gradient (DDPG), reported in Figure B.1, algorithm was developed for continuous action spaces [51]. DDPG employs an actor-critic architecture with separate networks for the actor and critic:

- **Critic Network:** Approximates the action-value function $Q_{\phi}(s,a)$. It is trained to minimize the Bellman error using a mean squared error loss:

$$L(\phi) = \mathbb{E}\left[\left(Q_{\phi}(s_t,a_t) - \left(r_{t+1} + \gamma \max_{a_{t+1}} Q_{\phi'}(s_{t+1},a_{t+1})\right)\right)^2\right]$$

  Here, ϕ represents the parameters of the critic network, and ϕ' are the parameters of a target critic network, which are periodically updated to provide stable targets for learning.

- **Actor Network:** Represents the policy $\pi_{\theta}(s)$, mapping states to actions. It

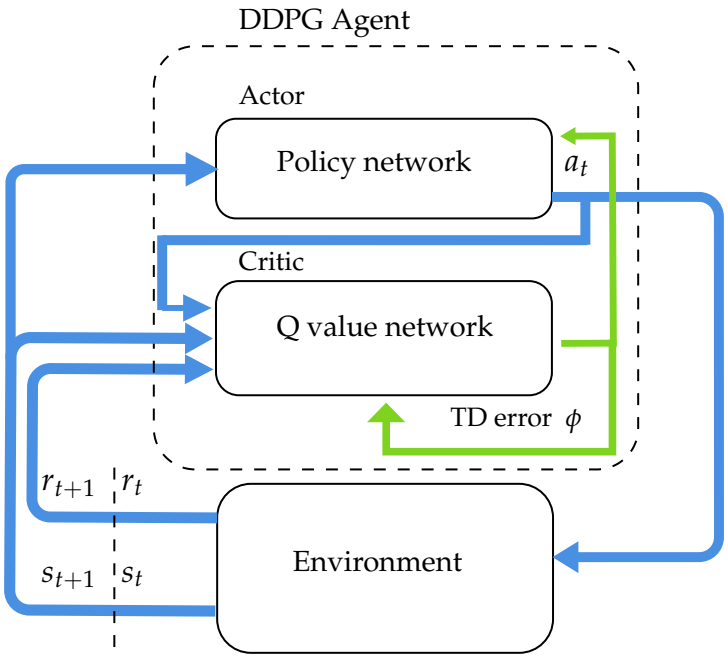


Figure B.1: Actor-Critic architecture: the actor produces an action given the current state of the environment, and the critic produces a temporal difference (TD) error signal given the state, the action, and the reward.

is trained to maximize the expected Q-value, using the critic's estimates to guide the policy improvement:

$$\max_{\theta} \mathbb{E}_{s\sim\mathcal{D}} \left[Q_\phi(s, \pi_\theta(s)) \right]$$

Here, θ represents the parameters of the actor network.

To improve stability and efficiency, DDPG employs techniques like experience replay and target networks:

- **Experience Replay:** The agent's experiences are stored in a replay buffer $\mathcal{D}$, allowing the networks to be trained on random mini-batches of experiences. This breaks the temporal correlation between consecutive experiences and leads to more stable learning.
- **Target Networks:** Separate, slowly-updated target networks for both the actor and critic provide stable targets for the Q-value updates, reducing oscillations and divergence during training.

In summary, RL is a powerful framework for learning optimal policies through environmental interaction. The combination of RL with deep learning techniques like DDPG enables effective handling of complex, continuous action spaces, making it a versatile and robust approach for a wide range of applications. When thinking about learning, one concept that immediately springs to mind is the notion of learning through interaction with the environment. This interaction provides us with insights into the effects of our actions, which can then be used to achieve goals. Reinforcement Learning (RL) is a machine learning technique that uses interactions to identify optimal behavior within an environment. In RL, the goal is to identify a policy—mapping states to actions—that maximizes the total expected reward. Rewards can accrue in both terminal and intermediate states. The agent seeks to gain valuable experience with states, actions, state transitions, and rewards to iteratively refine its behavior towards optimality. Unlike other machine learning paradigms, the evaluation of RL systems unfolds concurrently with the learning process. RL operates through a trial-and-error mechanism, navigating through delayed rewards while striking a balance between exploration and exploitation. More details on the DDPG algorithm can be found in [68].

Bibliography

[1] M. Terlizzi, G. Silano, L. Russo, M. Aatif, A. Basiri, V. Mariani, L. Iannelli, and L. Glielmo, "A vision-based algorithm for a path following problem," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2021, pp. 1630–1635.

[2] M. Terlizzi, L. Russo, E. Picariello, and L. Glielmo, "A novel algorithm for lane detection based on iterative tree search," in *2021 IEEE International Workshop on Metrology for Automotive (MetroAutomotive)*. IEEE, 2021, pp. 205–209.

[3] L. Russo, M. Terlizzi, M. Tipaldi, and L. Glielmo, "A reinforcement learning approach for pedestrian collision avoidance and trajectory tracking in autonomous driving systems," in *2021 5th International Conference on Control and Fault-Tolerant Systems (SysTol)*. IEEE, 2021, pp. 44–49.

[4] M. Terlizzi, V. Mariani, and L. Glielmo, "A model predictive scheme for autonomous vehicles cybersecurity," in *2021 5th International Conference on Control and Fault-Tolerant Systems (SysTol)*. IEEE, 2021, pp. 66–71.

[5] S. Hayes, "Industrial automation and stress, c.1945–79," in *Stress in Post-War Britain, 1945–85*, M. Jackson, Ed. New York, NY: Routledge, 2015, ch. 5. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK436950/

[6] K. Bimbraw, "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology," in *2015 12th international conference on informatics in control, automation and robotics (ICINCO)*, vol. 1. IEEE, 2015, pp. 191–198.

[7] C. G. Atkeson, P. B. Benzun, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, *et al.*, "What happened at the darpa robotics challenge finals," *The DARPA robotics challenge finals: Humanoid robots to the rescue*, pp. 667–684, 2018.

[8] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *IEEE/RSJ international conference on intelligent robots and systems*, vol. 3. IEEE, 2002, pp. 2478–2483.

[9] H. Chen, Y. Wen, M. Zhu, Y. Huang, C. Xiao, T. Wei, and A. Hahn, "From automation system to autonomous system: An architecture perspective," *Journal of Marine Science and Engineering*, vol. 9, no. 6, p. 645, 2021.

[10] S. Choi, F. Thalmayr, D. Wee, and F. Weig, "Advanced driver-assistance systems: Challenges and opportunities ahead," *McKinsey & Company*, pp. 1–11, 2016.

[11] U. Suddamalla, S. Kundu, S. Farkade, and A. Das, "A novel algorithm of lane detection addressing varied scenarios of curved and dashed lanemarks," in *2015 International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE, 2015, pp. 87–92.

[12] S. Sehestedt, S. Kodagoda, A. Alempijevic, and G. Dissanayake, "Robust lane detection in urban environments," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 123–128.

[13] G. Liu, S. Li, and W. Liu, "Lane detection algorithm based on local feature extraction," in *2013 Chinese Automation Congress*. IEEE, 2013, pp. 59–64.

[14] J. Wang, Y. Chen, J. Xie, and H. Lin, "Model-based lane detection and lane following for intelligent vehicles," in *2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2. IEEE, 2010, pp. 170–175.

[15] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision computing*, vol. 22, no. 4, pp. 269–280, 2004.

[16] J. Kim and M. Lee, "Robust lane detection based on convolutional neural network and random sample consensus," in *International conference on neural information processing*. Springer, 2014, pp. 454–461.

[17] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust lane detection from continuous driving scenes using deep neural networks," *IEEE transactions on vehicular technology*, vol. 69, no. 1, pp. 41–54, 2019.

[18] M. G. Albanesi, "Time complexity evaluation of algorithms for the hough transform on mesh connected computers," in *Proceedings, Advanced Computer Technology, Reliable Systems and Applications*. IEEE Computer Society, 1991, pp. 253–254.

[19] P. Mongkonyong, C. Nuthong, S. Siddhichai, and M. Yamakita, "Lane detection using randomized hough transform," in *IOP Conference Series: Materials Science and Engineering*, vol. 297, no. 1. IOP Publishing, 2018, p. 012050.

[20] A. Borkar, M. Hayes, and M. T. Smith, "A novel lane detection system with efficient ground truth generation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 365–374, 2012.

[21] H. Yoo, U. Yang, and K. Sohn, "Gradient-enhancing conversion for illumination-robust lane detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1083–1094, 2013.

[22] B. Maik and d. E. Pignaton, "A UAV guidance system using crop row detection and line follower algorithms," *Journal of Intelligent & Robotic Systems*, pp. 1–17, 2019.

[23] G. Silano, T. Baca, R. Penicka, D. Liuzza, and M. Saska, "Power Line Inspection Tasks with Multi-Aerial Robot Systems via Signal Temporal Logic Specifications," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4169–4176, 2021.

[24] B. Dahroug, J.-A. Séon, A. Oulmas, T. Xu, B. Tamadazte, N. Andreff, and S. Régnier, "Some Examples of Path Following in Microrobotics," in *International Conference on Manipulation, Automation and Robotics at Small Scales*, 2018, pp. 1–6.

[25] M. A. Rafique and A. F. Lynch, "Output-Feedback Image-Based Visual Servoing for Multirotor Unmanned Aerial Vehicle Line Following," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 4, pp. 3182–3196, 2020.

[26] D. Dagao, X. Meng, M. Qian, H. Zhongming, and W. Yueliang, "An improved Hough transform for line detection," in *2010 International Conference on Computer Application and System Modeling*, vol. 2, 2010, pp. V2–354.

[27] S. Du, B. J. van Wyk, C. Tu, and X. Zhang, "An Improved Hough Transform Neighborhood Map for Straight Line Segments," *IEEE Transactions on Image Processing*, vol. 19, no. 3, pp. 573–585, 2010.

[28] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[29] V. Nhan, J. Robert, and R. Davide, "LS-Net: Fast Single-Shot Line-Segment Detector," *Machine Vision and Applications*, vol. 12, no. 32, 2020.

[30] J. Tang, S. Li, and P. Liu, "A review of lane detection methods based on deep learning," *Patter Recognition*, vol. 111, pp. 1–15, 2021.

[31] P. Sujit, S. Saripalli, and J. B. Sousa, "An evaluation of UAV path following algorithms," in *2013 European Control Conference*, 2013, pp. 3332–3337.

[32] G. V. Pelizer, N. B. Da Silva, and K. R. Branco, "Comparison of 3d path-following algorithms for unmanned aerial vehicles," in *2017 International Conference on Unmanned Aircraft Systems*, 2017, pp. 498–505.

[33] S. Keshmiri, A. R. Kim, D. Shukla, A. Blevins, and M. Ewing, "Flight Test Validation of Collision and Obstacle Avoidance in Fixed-Wing UASs with High Speeds Using Morphing Potential Field," in *2018 International Conference on Unmanned Aircraft Systems*, 2018, pp. 589–598.

[34] T. Tuttle, T. T. Moleski, and J. Wilhelm, "Multi-Rotor Path-following Performance using Vector Field Guidance and Velocity Control," in *AIAA Scitech 2021 Forum*, 2021.

[35] A. S. Baqir and A. A. Ammar, "Navigation of Mini Unmanned Aerial Vehicle in Unknown Environment," *IOP Conference Series: Materials Science and Engineering*, vol. 745, 2020.

[36] A. Gautam, P. B. Sujit, and S. Saripalli, "Application of guidance laws to quadrotor landing," in *2015 International Conference on Unmanned Aircraft Systems*, 2015, pp. 372–379.

[37] D. M. Xavier, B. F. S. Natassya, and R. Branco Kalinka, "Path-following algorithms comparison using Software-in-the-Loop simulations for UAVs," in *2019 IEEE Symposium on Computers and Communications*, 2019, pp. 1216–1221.

[38] G. Silano, P. Oppido, and L. Iannelli, "Software-in-the-loop simulation for improving flight control system design: a quadrotor case study," in *IEEE International Conference on Systems, Man and Cybernetics*, 2019, pp. 466–471.

[39] Mathworks, *Mathworks Minidrone Competition IFAC20*. [Online]. Available: https://it.mathworks.com/academia/student-competitions/minidrones/ifac-2020.html

[40] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[41] G. Silano and L. Iannelli, "MAT-Fly: An Educational Platform for Simulating Unmanned Aerial Vehicles Aimed to Detect and Track Moving Objects," *IEEE Access*, vol. 9, pp. 39 333–39 343, 2021.

[42] T. N. Dief and S. Yoshida, "Review: Modeling and Classical Controller Of Quad-rotor," *International Journal of Computer Science and Information Technology & Security*, vol. 5, no. 4, pp. 314–319, 2015.

[43] MathWorks, "Simulink 3D Animation toolbox." [Online]. Available: https://www.mathworks.com/products/3d-animation.html

[44] ——, "Simulink Support Package for Parrot Minidrones." [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/63318-simulink-support-pack

[45] M. Terlizzi, "MATLAB Minidrone Competition IFAC20," YouTube. [Online]. Available: https://youtu.be/9VySp0j-1hc

[46] T. Kumar and K. Verma, "A theory based on conversion of rgb image to gray image," *International Journal of Computer Applications*, vol. 7, no. 2, pp. 7–10, 2010.

[47] R. C. Gonzalez, R. E. Woods, *et al.*, "Digital image processing," 2002.

[48] A. Kumar and P. Simon, "Review of lane detection and tracking algorithms in advanced driver assistance system," *International Journal of Computer Science and Information Technology*, vol. 7, pp. 65–78, 08 2015.

[49] M. Aly, "Real time detection of lane markers in urban streets," in *2008 IEEE Intelligent Vehicles Symposium*. IEEE, 2008, pp. 7–12.

[50] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[51] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[52] N. Deshpande and A. Spalanzani, "Deep reinforcement learning based vehicle navigation amongst pedestrians using a grid-based state representation," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2081–2086.

[53] M. Kim, S. Lee, J. Lim, J. Choi, and S.-G. Kang, "Unexpected collision avoidance driving strategy using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 17 243–17 252, 2020.

[54] G. D'Angelo, M. Tipaldi, F. Palmieri, and L. Gliemo, "A data-driven approximate dynamic programming approach based on association rule learning: Spacecraft autonomy as a case study," *Information Sciences*, vol. 504, pp. 501–519, 2019.

[55] M. Tipaldi, L. Feruglio, P. Denis, and G. D'Angelo, "On applying ai-driven flight data analysis for operational spacecraft model-based diagnostics," *Annual Reviews in Control*, vol. 49, pp. 197–211, 2020.

[56] N. C. for Statistics and Analysis, "Pedestrians: 2017 data," 2017.

[57] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[58] D. Bertsekas, "Multiagent reinforcement learning: rollout and policy iteration," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 249–272, 2021.

[59] M. Everett, Y.-F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, no. 2, pp. 10 357–10 377, 2021.

[60] C. S. Arvind and J. Senthilnath, "Autonomous rl: Autonomous vehicle obstacle avoidance in a dynamic environment using mlpsarsa reinforcement learning," in *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, 2019, pp. 120–124.

[61] J. Deichmann. The race for cybersecurity: Protecting the connected car in the era of new regulation. [Online]. Available: https://mck.co/3B5ga1S

[62] T. Zhang, H. Antunes, and S. Aggarwal, "Defending connected vehicles against malware: Challenges and a solution framework," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 10–21, 2014.

[63] Q. Liu, Y. Mo, X. Mo, C. Lv, E. Mihankhah, and D. Wang, "Secure pose estimation for autonomous vehicles under cyber attacks," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1583–1588.

[64] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "{SAVIOR}: Securing autonomous vehicles with robust physical invariants," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 895–912.

[65] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Transactions on Automatic control*, vol. 59, no. 6, pp. 1454–1467, 2014.

[66] Q. Sun, K. Zhang, and Y. Shi, "Resilient model predictive control of cyber–physical systems under dos attacks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4920–4927, 2019.

[67] Y. Liu, Y. Chen, M. Li, and Z. Wan, "Mpc for the cyber-physical system with deception attacks," in *2020 Chinese Control And Decision Conference (CCDC)*. IEEE, 2020, pp. 3847–3852.

[68] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[69] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[70] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 1094–1099.

[71] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of can bus security challenges," *Sensors*, vol. 20, no. 8, p. 2364, 2020.

[72] P. Polack, F. Altché, B. d'Andréa 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)*. IEEE, 2017, pp. 812–818.

[73] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015.

[74] L. Schenato, "To zero or to hold control inputs with lossy links?" *IEEE Transactions on Automatic Control*, vol. 54, no. 5, pp. 1093–1099, 2009.

[75] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, 2004.

[76] R. Quirynen, M. Vukov, and M. Diehl, "Multiple shooting in a microsecond," in *Multiple shooting and time domain decomposition methods*. Springer, 2015, pp. 183–201.

[77] Z. Khan, M. Chowdhury, M. Islam, C.-y. Huang, and M. Rahman, "Long short-term memory neural network-based attack detection model for in-vehicle network security," *IEEE Sensors Letters*, 2020.

[78] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[79] Q. Wu, X. Ge, Q.-L. Han, and Y. Liu, "Railway virtual coupling: A survey of emerging control techniques," *IEEE Transactions on Intelligent Vehicles*, 2023.

[80] Q. Wu, X. Ge, Q.-L. Han, B. Wang, H. Wu, C. Cole, and M. Spiryagin, "Dynamics and control simulation of railway virtual coupling," *Vehicle System Dynamics*, vol. 61, no. 9, pp. 2292–2316, 2023.

[81] X. Zhang, D. Yang, W. Zhang, and J. Huang, "Optimal robust constraints-following control for rail vehicle virtual coupling," *Journal of Vibration and Control*, vol. 29, no. 5-6, pp. 1352–1365, 2023.

[82] B. Wang, D. Yang, X. Zhang, and X. Jia, "Constraint-force driven control design for rail vehicle virtual coupling," *Journal of Vibration and Control*, vol. 28, no. 5-6, pp. 551–563, 2022.

[83] J. Park, B.-H. Lee, and Y. Eun, "Virtual coupling of railway vehicles: Gap reference for merge and separation, robust control, and position measurement," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1085–1096, 2022.

[84] Y. Liu, D. Ou, Y. Yang, and D. Dong, "A method for maintaining virtually coupled states of train convoys," *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 237, no. 2, pp. 243–252, 2023.

[85] G. Basile, E. Napoletano, A. Petrillo, and S. Santini, "Roadmap and challenges for reinforcement learning control in railway virtual coupling," *Discover Artificial Intelligence*, vol. 2, no. 1, p. 27, 2022.

[86] H. Wang, Q. Zhao, S. Lin, D. Cui, C. Luo, L. Zhu, X. Wang, and T. Tang, "A Reinforcement Learning Empowered Cooperative Control Approach for IIoT-Based Virtually Coupled Train Sets," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4935–4945, 2021.

[87] G. Basile, D. G. Lui, A. Petrillo, and S. Santini, "Deep deterministic policy gradient virtual coupling control for the coordination and manoeuvring of heterogeneous uncertain nonlinear high-speed trains," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108120, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197624002781

[88] J. Felez, Y. Kim, and F. Borrelli, "A model predictive control approach for virtual coupling in railways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2728–2739, 2019.

[89] Z. Wu, C. Gao, and T. Tang, "A virtually coupled metro train platoon control approach based on model predictive control," *IEEE Access*, vol. 9, pp. 56 354–56 363, 2021.

[90] W. Y. O. Peri Smith, Arnab Majumdar, "An overview of lessons learnt from ertms implementation in european railways," *Journal of Rail Transport Planning & Management*, vol. 2, no. 4, pp. 79–87, 2012.

[91] E. T. S. Institute, "Future rail mobile communication system (frmcs); study on system architecture," ETSI, Tech. Rep., 2020.

[92] . Shift2Rail Joint Undertaking, "Multi-annual action plan," Shift2Rail, Tech. Rep., 2020.

[93] N. Furness, H. Van Houten, L. Arenas, and M. Bartholomeus, "ERTMS level 3: the game-changer," *IRSE news*, vol. 232, pp. 2–9, 2017.

[94] F. Flammini, S. Marrone, R. Nardone, A. Petrillo, S. Santini, and V. Vittorini, "Towards railway virtual coupling," in *2018 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC)*, 2018, pp. 1–6.

[95] C. Di Meo, M. Di Vaio, F. Flammini, R. Nardone, S. Santini, and V. Vittorini, "ERTMS/ETCS Virtual Coupling: Proof of Concept and Numerical Analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2545–2556, 2020.

[96] I. Mitchell, "ERTMS Level 4, Train Convoys or Virtual Coupling," *IRSE news*, 2016.

[97] W. Xiao, C. G. Cassandras, and C. Belta, *Safe Autonomy with Control Barrier Functions: Theory and Applications*. Springer, 2023.

[98] H. K. Khalil, *Control of nonlinear systems*. Prentice Hall, New York, NY, 2002.

[99] Q. Wu, M. Spiryagin, and C. Cole, "Longitudinal train dynamics: an overview," *Vehicle System Dynamics*, vol. 54, no. 12, pp. 1688–1714, 2016.

[100] M. Terlizzi. (2024, aug) Railway tool transition level virtual coupling, https://www.youtube.com/watch?v=6Jr4a$_{w}$j9z8t = 1s.*YouTubevideo*.[*Online*].*Available* : *https* : *//www.youtube.com/watch*?$v = 6Jr4a_{w}j9z8t = 1s$

[101] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," in *IFAC World Congress*, Berlin, Germany, 2020.

[102] A. Frilli, E. Meli, D. Nocciolini, L. Pugi, A. Rindi, B. Romani, M. Ceraolo, and G. Lutzemberger, "The tesys rail project: Innovative models to enhance the energy sustainability of railway systems," *International Journal of Railway Technology*, vol. 6, pp. 1–28, 12 2017.