UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

University of Modena and Reggio Emilia

Department of Sciences and Methods for Engineering

Doctoral school of Industrial Innovation Engineering

XXXVIII cycle

Safe and Feasible Control of Autonomous Vehicles: A Control Barrier Function and Viability Approach

---

Author:
Filippo Bernabei

Supervisor:
Cristian Secchi

PhD Coordinator:
Franco Zambonelli

Reggio Emilia, Italy, February 2025

To my donor

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Cristian Secchi, for his guidance, support, and encouragement throughout my doctoral journey. He has played a key role in shaping both my research work and my professional development, helping me face challenges with greater clarity and confidence. I am sincerely grateful to my colleagues, whom I am proud to call friends, for the stimulating discussions, fruitful collaborations, and the positive environment we have built together. Beyond research, this journey has been enriched by shared experiences, conferences, travels, and the opportunity to meet extraordinary people from diverse backgrounds and cultures, which broadened my perspective both professionally and personally. Finally, I would like to thank my family for their unwavering support and belief in me. Their constant encouragement has been a source of motivation and strength throughout this journey.

Abstract

In recent years, autonomous driving has emerged as one of the most transformative and multidisciplinary challenges in robotics and control engineering. Enabling a vehicle to navigate complex, dynamic, and uncertain environments without human intervention demands seamless integration of perception, planning, decision making, and control. Among these components, the design of a robust and safe control layer is fundamental: it must ensure collision avoidance, adherence to vehicle constraints, and responsiveness to unpredictable agents such as pedestrians or other vehicles. In urban settings especially, the density of obstacles, frequent intersections, occlusions, and highly variable traffic dynamics raise significant difficulties for classical control architectures.

While many approaches focus on perception, mapping, or high-level planning, this work concentrates on the control problem. To achieve this goal, different types of Control Barrier Functions (CBFs) have been adopted to encode safety constraints and represent the interaction between the vehicle and surrounding obstacles. The use of pairwise CBFs allows one to handle multiple dynamic agents and obstacles, effectively mapping their relative behavior and enforcing collision avoidance in simulation. However, when transitioning from a kinematic to a dynamic vehicle model, necessary to represent higher speed and more realistic scenarios, the standard CBF formulation becomes insufficient. In such cases, Higher Order Control Barrier Functions (HOCBFs) are employed to incorporate higher order dynamics. Nonetheless, HOCBF based formulations often face limitations in feasibility when control inputs are bounded like in the real world.

To overcome these issues, this research explores the use of viability theory, which inherently guarantees the existence of safe control actions that keep the system trajectories within a viable set. Although conceptually powerful, direct implementation of viability based approaches is computationally demanding and unsuitable for real time control.

Therefore two works have been developed to approximate the viability kernel and exploit its safety guarantees in a more efficient manner. The first decoupled the dynamic bicycle model to make it affine and the second leverages the system's differential flatness to significantly reduce the complexity of the control problem. Since many robotic systems are

differentially flat, this solution can be a new approach to transform the original nonlinear system into a lower-dimensional representation, where safety and viability constraints can be enforced more efficiently.

The proposed framework demonstrates that it is possible to combine the formal guarantees of control barrier functions with the generality of viability theory while maintaining computational tractability. This hybrid approach ensures both safety and feasibility in autonomous driving tasks, even under dynamic and uncertain urban conditions. The developed control algorithms have been extensively validated both in a high-fidelity simulation environment and through real-world experimental tests, confirming the effectiveness and robustness of the proposed methods. The resulting control architecture provides a viable foundation for future extensions toward cooperative multi vehicle scenarios and real world experimental validation.

Sommario

Negli ultimi anni, la guida autonoma è emersa come una delle sfide più complesse e multidisciplinari nell'ambito della robotica e dell'ingegneria del controllo. Consentire a un veicolo di muoversi in ambienti complessi, dinamici e incerti senza intervento umano richiede un'integrazione sinergica tra percezione, pianificazione, decisione e controllo. Tra questi elementi, la progettazione di uno strato di controllo robusto e sicuro è fondamentale: deve garantire l'evitamento delle collisioni, il rispetto dei vincoli del veicolo e la capacità di reagire a comportamenti imprevedibili di altri agenti, come pedoni o veicoli. In particolare, negli ambienti urbani la densità degli ostacoli, la presenza di incroci, le occlusioni e la variabilità del traffico rappresentano una sfida notevole per le architetture di controllo classiche.

Sebbene molte ricerche si concentrino su percezione, mappatura o pianificazione ad alto livello, questo lavoro si focalizza sul problema del controllo. A tal fine, sono state adottate diverse tipologie di Control Barrier Function (CBF) per codificare i vincoli di sicurezza e rappresentare l'interazione tra il veicolo e gli ostacoli circostanti. L'utilizzo delle pair-wise CBF consente di gestire più agenti dinamici e ostacoli, mappando i comportamenti relativi e imponendo l'evitamento delle collisioni in simulazione. Tuttavia, passando da un modello cinematico a uno dinamico, necessario per descrivere scenari più realistici e ad alta velocità, la formulazione standard delle CBF non risulta più sufficiente. In tali casi vengono impiegate le Higher Order Control Barrier Function (HOCBF) per includere dinamiche di ordine superiore, ma queste possono presentare problemi di fattibilità quando gli input di controllo sono limitati, come nei sistemi reali.

Per superare tali problematiche, questo lavoro esplora l'utilizzo della teoria della viabilità, che garantisce intrinsecamente l'esistenza di azioni di controllo sicure in grado di mantenere le traiettorie del sistema all'interno di un insieme viabile. Sebbene potente dal punto di vista teorico, l'applicazione diretta della viabilità risulta computazionalmente onerosa e difficilmente applicabile in tempo reale.

Per questo motivo sono state sviluppate due metodologie atte ad approssimare il kernel di viabilità e a sfruttarne le proprietà di sicurezza in modo più efficiente. La prima decoppia

il modello dinamico del veicolo a bicicletta per renderlo affine, mentre la seconda sfrutta la proprietà di differential flatness per ridurre significativamente la complessità del problema di controllo. Poiché molti sistemi robotici, inclusi i modelli dinamici dei veicoli, sono flat, tale approccio consente di trasformare un sistema non lineare in una rappresentazione a dimensionalità ridotta, in cui i vincoli di sicurezza e viabilità possono essere imposti in maniera più efficiente.

Il framework proposto dimostra la possibilità di combinare le garanzie formali delle funzioni di barriera con la generalità della teoria della viabilità, mantenendo al contempo una trattabilità computazionale adeguata. Questo approccio ibrido assicura sicurezza e fattibilità nelle manovre di guida autonoma anche in condizioni urbane dinamiche e incerte. Gli algoritmi di controllo sviluppati sono stati ampiamente validati sia in un ambiente di simulazione ad alta fedeltà sia mediante test sperimentali su veicoli reali, confermando l'efficacia e la robustezza delle metodologie proposte. L'architettura di controllo risultante rappresenta una base solida per future estensioni verso scenari cooperativi multi-veicolo e validazioni su larga scala.

Contents

List of Figures

Notation and Acronyms

Acronyms

Acronym	*Meaning*
CBF	*Control Barrier Function*
ZCBF	*Zeroing Control Barrier Function*
HOCBF	*High Order Control Barrier Function*
QP	*Quadratic Program*
QCQP	*Quadratically Constrained Quadratic Program*
NLP	*Nonlinear Program*
MPC	*Model Predictive Control*
V2I	*Vehicle-to-Infrastructure communication*
I2V	*Infrastructure-to-Vehicle communication*
V2V	*Vehicle-to-Vehicle communication*
DOF	*Degree of Freedom*
ROS	*Robot Operating System*
LIMO	*AgileX LIMO scaled mobile robotic platform*

Notation

Symbol	*Description*	*Context*
x	*Generic system state vector*	*General*
u	*Generic control input vector*	*General*
$f(x), g(x)$	*Drift and input vector fields of a control-affine system*	*General*
$h(x)$	*Barrier function defining a safety constraint*	*CBF/HOCBF*
$\mathcal{C}$	*Safe set induced by* $h(x) \geq 0$	*CBF/HOCBF*
$\alpha(\cdot)$	*Extended class-*$\mathcal{K}$ *function used in barrier conditions*	*CBF/HOCBF*
λ, λ_i	*Barrier gains regulating the response near constraint boundaries*	*CBF/HOCBF*
ψ_i	*Auxiliary functions used in the recursive HOCBF construction*	*HOCBF*

Symbol	*Description*	*Context*
u_{des}	*Nominal input before safety filtering*	*Control*
$u^\star$	*Optimized safe control input*	*Control*
x_1, x_2	*Planar position coordinates of the kinematic vehicle model*	*Kinematic model*
x_3	*Vehicle yaw angle in the kinematic vehicle model*	*Kinematic model*
x_4	*Front steering angle in the kinematic vehicle model*	*Kinematic model*
J	*Vehicle wheelbase in the kinematic bicycle model*	*Kinematic model*
P	*Virtual control point located ahead of the vehicle*	*Kinematic model*
q	*Distance of the virtual point from the front axle along the steering direction*	*Kinematic model*
x_P	*Planar position of the virtual point P*	*Kinematic model*
u_P	*Virtual planar velocity input associated with x_P*	*Kinematic model*
$T(x_3, x_4)$	*Transformation matrix mapping physical kinematic inputs to virtual point velocity*	*Kinematic model*
η	*Slack variable used for controlled relaxation of selected constraints*	*CBF-QP*
ζ	*Binary activation variable enabling dashed-lane relaxation during overtaking*	*CBF-QP*
c	*Confidence factor regulating conservativeness at intersections*	*Intersection handling*
dist_{sf}	*Speed-dependent safety distance for stop and right-of-way constraints*	*Intersection handling*
v_x, v_y	*Longitudinal and lateral velocities in the body-fixed frame*	*Dynamic model*
X, Y	*Vehicle position in the inertial frame*	*Dynamic model*
θ	*Vehicle heading angle in the inertial frame*	*Dynamic model*
F_{xf}	*Longitudinal traction or braking force at the front axle*	*Dynamic model*
δ	*Steering angle in the dynamic bicycle model*	*Dynamic model*
m	*Vehicle mass*	*Dynamic model*
I_{zz}	*Yaw moment of inertia*	*Dynamic model*
l_f, l_r	*Distances from the center of mass to the front and rear axles*	*Dynamic model*
R_w	*Wheel radius*	*Dynamic model*
K_f, K_r	*Front and rear cornering stiffness coefficients*	*Dynamic model*
α_f, α_r	*Front and rear tire slip angles*	*Dynamic model*
e, ϕ	*Cross-track and heading errors used by the Stanley controller*	*Tracking control*
k, k_s	*Stanley-controller gains*	*Tracking control*
$a^V_{x,m}, a^V_{x,M}$	*Lower and upper viable longitudinal acceleration bounds*	*Viability*
$X_{\min}, X_{\max}$	*Longitudinal admissible position bounds used in the viability algorithm*	*Viability*

Symbol	*Description*	*Context*
Δt	*Discrete control or prediction time step*	*Viability*
y	*Flat output of a differentially flat system*	*Flatness*
x_e	*Extended flat-output state collecting y and its derivatives*	*Flatness*
u_v	*Virtual input of the flat-output integrator chain*	*Flatness*
A_e, B_e	*Companion matrices of the flat-output integrator-chain representation*	*Flatness*
φ_x, φ_u	*Algebraic reconstruction maps for state and input from flat outputs*	*Flatness*
$s^\star$	*Viability-consistent snap or projected recovery bound*	*Flatness/viability*

List of Publications

Journal papers

- F. Bernabei and C. Secchi, "Smart infrastructure and autonomous vehicles: Ensuring safety and efficiency in urban traffic with control barrier functions," *Mechatronics*, 2024
- F. Bernabei, C. Secchi, and G. Notomista, "Viable and computationally efficient safety for differentially flat dynamical systems with application to autonomous vehicles," *IEEE Robotics and Automation Letters*, 2026, submitted

Conference papers

- F. Bernabei and C. Secchi, "A constraint-based control architecture for urban autonomous vehicles," in *International Conference on Methods and Models in Automation and Robotics*, 2024, young Author Best Paper Award
- ——, "A constraint-driven control system for autonomous vehicles navigating in urban environments," in *European Control Conference (ECC)*, 2026, submitted

Chapter 1

Introduction

Urban mobility is undergoing a profound transformation driven by increasing urbanization, rising traffic density, and growing societal demands for safety, efficiency, and sustainability. In this context, autonomous driving has emerged as a key technological enabler with the potential to significantly reduce traffic accidents, improve traffic flow, and enhance accessibility to transportation services. Urban environments, however, represent one of the most challenging operating conditions for autonomous vehicles. Dense traffic, frequent interactions with vulnerable road users such as pedestrians and cyclists, complex road infrastructures, and strict traffic regulations create highly dynamic and uncertain scenarios that demand advanced and reliable control strategies [5, 6]. Unlike structured environments such as highways, urban driving requires autonomous vehicles to continuously negotiate right-of-way at intersections, perform lane keeping and lane changes, execute overtaking maneuvers, and react to both static and dynamic obstacles, often under limited visibility conditions. These tasks must be accomplished while respecting traffic rules, physical limitations of the vehicle, and comfort constraints, all within a real-time operational framework. As a consequence, safety in urban autonomous driving is not merely a matter of collision avoidance at the planning level, but a systemic requirement that must be enforced throughout the control architecture. Autonomous driving systems are typically organized into a hierarchical architecture comprising perception, decision-making, motion planning, and control layers [7, 8]. While perception and planning are responsible for understanding the environment and generating desired trajectories, the control layer plays a central role in ensuring that such trajectories are executed safely and feasibly on the physical vehicle. In particular, the control component is directly responsible for enforcing safety constraints arising from vehicle dynamics, actuator limitations, environmental boundaries, and traffic regulations, while compensating for modeling errors, disturbances, and rapidly changing conditions [9, 10]. Therefore, the ability to guarantee safety at the control level is a fundamental requirement for the deployment of autonomous vehicles in real-world urban scenarios. From a control perspective, these requirements translate into constraints on both the vehicle state and control inputs. Some constraints are persistent, such as lane boundaries or actuator limits, while others are transient and situation-dependent, such as stopping at intersections, yielding to other road users, or executing overtaking maneuvers. Moreover, these constraints may

be mutually conflicting: ensuring collision avoidance, for instance, may require aggressive braking that approaches actuator saturation, while maintaining comfort or tracking performance. As a result, the set of admissible control actions is fundamentally restricted by the physical capabilities of the vehicle and by the need to satisfy multiple safety constraints simultaneously. These considerations highlight a key aspect of urban autonomous driving: safety cannot be treated as a purely geometric or planning-level problem. Even when a collision-free trajectory is generated at the planning stage, its execution may be infeasible due to dynamic limitations, actuator saturations, or unexpected disturbances. This requirement places autonomous driving within the broader class of safety-critical control problems, where guaranteeing constraint satisfaction is as important as achieving performance objectives. In urban autonomous driving, control architectures must enforce safety constraints online, adapt to rapidly changing environments, and remain computationally efficient for real-time implementation. These requirements have led to a wide range of approaches, each addressing specific aspects of the problem while often relying on simplifying assumptions. Planning-based methods rely on spatial discretizations such as occupancy grids or lattice representations to generate collision-free trajectories [11]. Although structured and interpretable, these approaches may suffer from scalability issues and typically enforce safety through geometric margins without explicitly accounting for dynamic and actuation limits. Trajectory generation and sampling-based strategies search directly in the space of feasible motions [12], enabling the inclusion of dynamic constraints. However, safety is usually guaranteed through conservative buffers or cost penalties, which complicates the provision of formal guarantees under uncertainty. Learning-based approaches have demonstrated strong empirical performance in complex scenarios [13], yet their limited interpretability and lack of formal safety assurances restrict their standalone use in safety-critical systems. Model Predictive Control (MPC) explicitly incorporates system dynamics and constraints within a finite-horizon optimization framework [14]. While flexible, MPC formulations for urban driving may lead to high-dimensional nonlinear programs that must be solved repeatedly in real time. Control Barrier Functions (CBFs) provide a formal mechanism to enforce forward invariance of safe sets through online optimization [15]. Their computational efficiency makes them attractive for real-time applications, but classical formulations are limited to relative-degree-one systems. High Order Control Barrier Functions (HOCBFs) extend this framework to dynamic models [16], yet feasibility may be lost when hard input constraints and actuator saturation are considered. This limitation reveals a fundamental gap between local safety enforcement and long-term feasibility. In urban environments, multiple interacting constraints and strict actuation limits may cause the feasible set of the online optimization problem to shrink or become empty, even if safety is locally enforced. Viability theory offers a complementary perspective by characterizing the set of states from which constraint satisfaction can be maintained indefinitely [17]. However, exact computation of viability kernels for nonlinear systems is generally intractable in real time. This thesis bridges these perspectives by integrating barrier-based safety enforcement with viability-inspired reasoning. Control Barrier Functions provide efficient local constraint enforcement, while viability considerations ensure that the system remains within regions from which safe evolution is still possible. Differential flatness plays a central enabling role in this integration. Many vehicle models, including dynamic

bicycle models, are differentially flat [18], allowing the nonlinear dynamics to be transformed into an equivalent integrator-chain representation in the flat output space. This transformation simplifies constraint handling and enables the formulation of HOCBF and viability-inspired bounds within computationally tractable online optimization problems. Building upon these principles, this thesis develops a unified safety-critical control framework tailored to urban autonomous driving, reconciling formal guarantees with real-time feasibility under realistic dynamic and actuation constraints.

Thesis Outline and Contributions Autonomous driving in urban environments requires control strategies that go beyond trajectory generation and motion planning, explicitly addressing safety, feasibility, and real-time implementability under realistic operating conditions. In dense and dynamic urban scenarios, autonomous vehicles must simultaneously satisfy heterogeneous safety constraints, operate under strict actuator limitations, and react to rapidly changing environments, all while ensuring that safety guarantees remain enforceable over time. Building upon the limitations of purely local safety enforcement discussed in the previous section, this thesis adopts the perspective that safety-critical control for autonomous driving must account not only for instantaneous constraint satisfaction, but also for the long-term feasibility of admissible control actions under physical and operational constraints.

This thesis develops a unified constraint-based control architecture for urban autonomous driving through a progressive and structured integration of Control Barrier Functions, High Order Control Barrier Functions, viability-inspired reasoning, and differential flatness. Rather than proposing a monolithic solution, the thesis constructs the architecture incrementally: starting from CBF-based safety enforcement for kinematic models, extending to dynamic models via HOCBFs, addressing feasibility through viability-driven constraints, and finally achieving computational scalability through differential flatness and cooperative safety mechanisms. This step-by-step development results in a complete, feasible, and real-time-capable control framework for urban autonomous vehicles.

The research developed in this thesis is guided by the following questions:

1. How can heterogeneous urban driving requirements, such as lane keeping, obstacle avoidance, overtaking, and right-of-way management, be encoded within a unified safety-critical control layer?
2. How can barrier-based safety guarantees be preserved when moving from kinematic vehicle models to dynamic models with higher relative degree constraints and bounded actuation?
3. How can recursive feasibility be maintained when hard input constraints make classical CBF or HOCBF conditions insufficient?
4. How can the resulting safety filter remain computationally suitable for real-time implementation and experimental deployment?

In response to these questions, the thesis provides the following main contributions:

1. A modular CBF-based control architecture for urban autonomous driving, in which traffic rules and safety requirements are represented as constraints in a single online optimization problem.

2. A pairwise CBF formulation for infrastructure-assisted driving, enabling smart infrastructure information to reshape intersection-related safe sets while preserving convexity and forward-invariance guarantees.

3. A dynamic HOCBF-based control strategy combined with viability-inspired acceleration bounds, addressing the loss of feasibility caused by actuator saturation and higher relative degree safety constraints.

4. A flatness-based viability-aware safety filter for dynamic vehicle models, reducing the online optimization complexity while maintaining safety, feasibility, and real-time computational performance.

5. A validation campaign combining MATLAB simulations, high-fidelity robotic simulation, comparison with alternative approaches, and preliminary experimental deployment on a scaled autonomous vehicle platform.

The dissertation is organized as follows.

Before presenting the progressive development of the proposed architecture, **Chapter 2** provides the theoretical foundations required throughout the thesis.

Chapter 3 introduces the foundational layer of the proposed architecture by presenting a constraint-based control framework for autonomous vehicles operating in urban environments. Urban traffic rules and road constraints—including lane keeping, lane changes, overtaking maneuvers, obstacle avoidance, and right-of-way management—are systematically encoded using Control Barrier Functions. The architecture relies exclusively on onboard and infrastructure-based sensing, without requiring vehicle-to-vehicle communication, and allows the shaping of driving behavior through appropriate tuning of barrier function parameters.

The work presented in this chapter has resulted in the following publication:

- **A Constraint-Based Control Architecture for Urban Autonomous Vehicles**
  Filippo Bernabei, Cristian Secchi
  IEEE International Conference on Mechatronics (ICM/MMAR), 2024.
  Young Author Best Paper Award.

Chapter 4 extends the baseline CBF architecture by incorporating smart infrastructure through vehicle-to-infrastructure interaction. Pairwise Control Barrier Functions are employed to integrate information from infrastructure-mounted sensors, enabling infrastructure-assisted safety enforcement in challenging urban scenarios such as low-visibility and partially occupied intersections. This chapter demonstrates how cooperative perception can be integrated into the control layer to enhance safety and traffic efficiency while preserving the decentralized structure of the architecture.

This work has led to the following publication:

- **Smart Infrastructure and Autonomous Vehicles: Ensuring Safety and Efficiency in Urban Traffic with Control Barrier Functions**
  Filippo Bernabei, Cristian Secchi
  Mechatronics, 2024.

Chapter 5 addresses the limitations of CBF-based approaches when applied to dynamic vehicle models subject to actuator saturations. By integrating High Order Control Barrier Functions with viability-inspired reasoning, this chapter introduces a dynamic formulation that explicitly ensures the existence of admissible control inputs over time. Viability-driven bounds are derived to prevent infeasibility in the online optimization problem, enabling the enforcement of safety constraints at the acceleration level under realistic actuation limits in urban driving scenarios.

The results presented in this chapter are based on the following work:

- **A Constraint-Driven Control System for Autonomous Vehicles Navigating in Urban Environments**
  Filippo Bernabei, Cristian Secchi
  European Control Conference (ECC), 2026
  submitted.

Chapter 6 completes the proposed architecture by exploiting differential flatness as a unifying tool for computational efficiency and scalability. By transforming dynamic vehicle models into equivalent chain-of-integrators representations in the flat output space, this chapter enables the tractable formulation of both High Order Control Barrier Functions and viability-inspired constraints. The resulting framework significantly reduces computational complexity and facilitates real-time implementation, while preserving formal safety guarantees. The approach is validated on dynamic vehicle models for autonomous driving and compared with state-of-the-art methods, highlighting improvements in feasibility, efficiency, and ease of implementation.

The results presented in this chapter are based on the following work:

- **Viable and Computationally Efficient Safety for Differentially Flat Dynamical Systems with Application to Autonomous Vehicles**
  Filippo Bernabei, Cristian Secchi, Gennaro Notomista
  under review.

Finally, **Chapter 7** summarizes the main contributions of the thesis, discusses current limitations, and outlines future research directions in safety-critical control for autonomous vehicles.

Chapter 2

Theoretical Background

2.1 Control Barrier Functions

The enforcement of safety constraints is a fundamental requirement in the control of dynamical systems operating in uncertain and dynamic environments. In many applications, particularly in robotics and autonomous systems, safety specifications are naturally expressed as state constraints that must be satisfied at all times, such as collision avoidance, actuator saturation limits, or workspace boundaries. Unlike performance objectives, which can often be relaxed or traded off, safety constraints must be enforced rigorously and continuously throughout the system evolution.

The notion of barrier functions has its roots in optimization theory and interior point methods, where barrier terms are introduced to prevent optimization trajectories from approaching the boundary of feasible sets [19, 20]. This idea has subsequently been extended to dynamical systems, where barrier like conditions are used to characterize the forward invariance of sets under the system flow. Early formulations of barrier certificates and invariance conditions can be traced back to Lyapunov based arguments and viability theory [21, 22].

Within the control community, these concepts have been formalized and unified under the framework of Control Barrier Functions , which provide sufficient conditions for enforcing state constraints through feedback control laws [15, 23]. In particular, the work of Ames et al. has played a central role in establishing CBFs as a practical tool for safety critical control, especially in conjunction with real time optimization and quadratic programming [24].

Several classes of barrier functions have been proposed in the literature. Reciprocal barrier functions enforce safety by penalizing the inverse of the constraint function and are closely related to interior point formulations [23]. Logarithmic and exponential barrier functions have also been considered, particularly in optimal control and constrained stabilization problems [21]. While effective in certain contexts, these approaches often lead to nonlinear or nonconvex constraints on the control input, limiting their applicability in real time control architectures.

A particularly influential formulation is that of Zeroing Control Barrier Functions (ZCBFs), which encode safety requirements through differential inequalities that directly constrain the time derivative of the barrier function [25]. ZCBFs offer a mathematically transparent condition for forward invariance and, for control affine systems with relative degree one, lead to affine constraints in the control input. This property makes them especially well suited for integration with quadratic programs and other convex optimization based control schemes [15, 24].

For these reasons, and in line with the objectives of this thesis, we adopt the ZCBF framework as the primary tool for safety enforcement. In the following, we introduce the basic notions of safe sets and forward invariance, and then formalize the definition of Control Barrier Functions within the zeroing framework. From this point onward, and unless explicitly stated otherwise, all references to Control Barrier Functions in this thesis are to be understood as referring to Zeroing Control Barrier Functions.

Consider a nonlinear control affine system of the form

$$\dot{x} = f(x) + g(x)u, \tag{2.1}$$

where $x \in \mathcal{D} \subset \mathbb{R}^n$ denotes the system state and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input. The vector fields $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are assumed to be locally Lipschitz continuous, ensuring existence and uniqueness of solutions.

Safety constraints are encoded through a continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$, which defines the safe set

$$\mathcal{C} := \{x \in \mathcal{D} \mid h(x) \geq 0\}. \tag{2.2}$$

The boundary of the set is given by $\partial\mathcal{C} := \{x \in \mathcal{D} \mid h(x) = 0\}$, while its interior corresponds to $\mathrm{Int}(\mathcal{C}) := \{x \in \mathcal{D} \mid h(x) > 0\}$.

Definition 1 (Forward Invariance)**.** *The set $\mathcal{C}$ is said to be forward invariant with respect to system* (2.1) *if, for any initial condition $x(0) \in \mathcal{C}$, the corresponding trajectory satisfies $x(t) \in \mathcal{C}$ for all $t \geq 0$.*

Forward invariance formalizes the notion of safety in dynamical systems: if $\mathcal{C}$ is forward invariant, then trajectories starting from a safe configuration can never violate the constraint $h(x) \geq 0$.

A sufficient condition for forward invariance can be obtained by analyzing the evolution of the function $h(x(t))$ along the system trajectories. Differentiating h along (2.1) yields

$$\dot{h}(x, u) = \frac{\partial h}{\partial x}(x)\big(f(x) + g(x)u\big). \tag{2.3}$$

Intuitively, safety can be enforced by ensuring that the value of $h(x)$ does not decrease too rapidly as the state approaches the boundary $\partial\mathcal{C}$. In particular, when $h(x)$ is close to zero, the control input should be chosen so as to prevent $\dot{h}$ from becoming negative in a way that would drive the system outside the safe set.

This idea can be formalized using extended class $\mathcal{K}_\infty$ functions.

Definition 2 (Extended Class $\mathcal{K}_\infty$ Function). *A function $\alpha : \mathbb{R} \to \mathbb{R}$ is an extended class $\mathcal{K}_\infty$ function if it is continuous, strictly increasing, satisfies $\alpha(0) = 0$, and is defined for all real arguments.*

The barrier condition is then expressed as

$$\sup_{u \in \mathcal{U}} \left[\frac{\partial h}{\partial x}(x) f(x) + \frac{\partial h}{\partial x}(x) g(x) u \right] \geq \alpha(h(x)). \tag{2.4}$$

This inequality ensures that the rate of change of $h(x)$ is constrained in a manner that prevents trajectories from crossing the boundary of the safe set in finite time.

Definition 3 (Control Barrier Function). *A continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$ is called a Zeroing Control Barrier Function (ZCBF) for system (2.1) if there exists an extended class $\mathcal{K}_\infty$ function α such that condition (2.4) holds for all $x \in \mathcal{D}$.*

If h is a ZCBF, then for any initial condition $x(0) \in \mathcal{C}$ there exists a control input that renders the safe set $\mathcal{C}$ forward invariant. A commonly adopted choice is

$$\alpha(h(x)) = \lambda h(x), \qquad \lambda > 0, \tag{2.5}$$

which enforces an exponential repulsion from the boundary of the safe set.

In practical control architectures, safety constraints are typically enforced by minimally modifying a nominal control input u_{des}, designed to achieve performance objectives such as tracking or stabilization. This leads naturally to a Quadratic Program of the form

$$\begin{aligned} u^* = \arg\min_{u \in \mathcal{U}} \quad & \|u - u_{\text{des}}\|^2 \\ \text{subject to} \quad & \frac{\partial h}{\partial x}(x) f(x) + \frac{\partial h}{\partial x}(x) g(x) u \geq \alpha(h(x)). \end{aligned} \tag{2.6}$$

The limitation of standard ZCBFs becomes evident when considering systems in which the control input does not appear in the first time derivative of the safety function.

Consider the double integrator

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = u, \tag{2.7}$$

which can be written in control affine form as in (2.1) with

$$f(x) = \begin{bmatrix} x_2 \\ 0 \end{bmatrix}, \qquad g(x) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Assume that safety is defined by a position constraint

$$h(x) = x_1, \tag{2.8}$$

so that the safe set is

$$\mathcal{C} = \{x \in \mathbb{R}^2 \mid x_1 \geq 0\}.$$

Differentiating h along the system trajectories yields

$$\dot{h}(x,u) = \frac{\partial h}{\partial x}(f(x) + g(x)u) = [1 \quad 0] \begin{bmatrix} x_2 \\ 0 \end{bmatrix} = x_2. \tag{2.9}$$

Crucially, the control input u does not appear in $\dot{h}$. Therefore, condition (2.4) cannot be enforced through the control input. In particular, if the system reaches the boundary $x_1 = 0$ with negative velocity $x_2 < 0$, then

$$\dot{h} = x_2 < 0,$$

and no admissible control action can prevent the state from leaving the safe set.

This shows that $h(x) = x_1$ has relative degree two with respect to system (2.7). As a consequence, standard Zeroing Control Barrier Functions are not sufficient to guarantee forward invariance.

The previous example highlights a structural limitation of standard ZCBFs: they require the safety function to have relative degree one with respect to the system dynamics. When the relative degree is greater than one, the control input does not explicitly appear in the first derivative of h, and the barrier constraint cannot directly influence the evolution of the safety function.

This limitation motivates the introduction of High Order Control Barrier Functions (HOCBFs), which systematically extend the barrier framework to constraints of higher relative degree by recursively differentiating the safety function until the control input explicitly appears.

2.2 High Order Control Barrier Functions

The Control Barrier Functions introduced in the previous section provide a constructive mechanism to enforce the forward invariance of safe sets for nonlinear control affine systems, under the assumption that the safety function has relative degree one with respect to the control input. While this assumption is satisfied in a number of simplified models, it is often violated in realistic dynamical systems, where control inputs act on higher order dynamics such as forces or accelerations rather than directly on configuration variables [15, 24].

In many safety critical applications, including autonomous driving, aerial vehicles, and robotic manipulation, safety constraints are naturally defined on quantities such as position, inter agent distance, or distance to obstacles. When dynamic models are employed, these quantities typically exhibit relative degree greater than one, making standard (first order) Control Barrier Function formulations inapplicable. This limitation has been widely recognized in the literature and has motivated the development of generalized barrier formulations capable of handling higher relative degrees [26].

To address this issue, the concept of High Order Control Barrier Functions has been intro-

duced as a systematic extension of Zeroing Control Barrier Functions to safety functions with arbitrary relative degree [25, 26]. HOCBFs preserve the invariance based interpretation of CBFs while enabling the enforcement of safety constraints for a much broader class of dynamical systems.

Consider again the nonlinear control affine system

$$\dot{x} = f(x) + g(x)u, \tag{2.10}$$

and let $h : \mathcal{D} \to \mathbb{R}$ be a sufficiently smooth function defining the safety constraint. The function $h(x)$ is said to have relative degree m with respect to the input u if

$$L_g L_f^k h(x) = 0 \quad \forall k < m-1, \qquad L_g L_f^{m-1} h(x) \neq 0, \tag{2.11}$$

where $L_f h$ and $L_g h$ denote the Lie derivatives of h along the vector fields f and g, respectively.

The notion of relative degree captures the number of system differentiations required before the control input explicitly influences the evolution of the safety function. In the context of autonomous driving, for instance, safety constraints defined on vehicle position or inter vehicle distance typically have relative degree two when acceleration based vehicle models are considered, reflecting the physical structure of the dynamics [24].

To extend barrier based safety guarantees to functions with relative degree $m > 1$, HOCBFs rely on the recursive construction of a sequence of auxiliary functions that progressively shape the evolution of the safety constraint [26]. Specifically, a sequence of functions $\psi_0, \psi_1, \ldots, \psi_m$ is defined as

$$\psi_0(x,t) := h(x,t), \tag{2.12}$$
$$\psi_1(x,t) := \dot{\psi}_0(x,t) + \alpha_1(\psi_0(x,t)), \tag{2.13}$$
$$\psi_2(x,t) := \dot{\psi}_1(x,t) + \alpha_2(\psi_1(x,t)), \tag{2.14}$$
$$\vdots$$
$$\psi_m(x,t) := \dot{\psi}_{m1}(x,t) + \alpha_m(\psi_{m1}(x,t)), \tag{2.15}$$

where $\alpha_i(\cdot)$, $i = 1, \ldots, m$, are class $\mathcal{K}$ functions.

Each auxiliary function ψ_i induces a time varying set

$$\mathcal{C}_i(t) := \{x \in \mathbb{R}^n \mid \psi_{i-1}(x,t) \geq 0\}, \tag{2.16}$$

and the intersection of these sets characterizes the region of the state space where the original safety constraint is satisfied up to order $m1$. Enforcing the forward invariance of this intersection guarantees that the original constraint $h(x) \geq 0$ is preserved over time.

Definition 4 (High Order Control Barrier Function)**.** *A function $h : \mathbb{R}^n \times [t_0, \infty) \to \mathbb{R}$ is called a High Order Control Barrier Function (HOCBF) of relative degree m for the*

considered system if there exist class $\mathcal{K}$ functions $\alpha_1, \ldots, \alpha_m$ such that

$$L_f^m h(x,t) + L_g L_f^{m-1} h(x,t) u + \frac{\partial h(x,t)}{\partial t} + \mathcal{O}(h(x,t)) + \alpha_m(\psi_{m1}(x,t)) \geq 0 \tag{2.17}$$

for all $(x,t) \in \mathcal{C}_1(t) \cap \cdots \cap \mathcal{C}_m(t)$.

The term $\mathcal{O}(h(x,t))$ collects all lower order Lie derivatives and time derivative terms arising from the recursive differentiation process. As in the standard CBF case, this condition defines a set of admissible control inputs that render the safe set forward invariant [26].

Similarly to Zeroing Control Barrier Functions, a commonly adopted choice is

$$\alpha_m(\psi_{m1}) = \lambda_m \psi_{m1}, \qquad \lambda_m > 0, \tag{2.18}$$

which introduces an exponential type constraint and allows tuning the responsiveness of the system near the boundary of the safe set [24, 26].

In practical implementations, the HOCBF constraint induces a state-dependent admissible control set

$$\begin{aligned} \mathcal{K}_{\text{hocbf}}(x,t) = \Big\{ u \in \mathcal{U} \;\Big|\; & L_f^m h(x,t) + L_g L_f^{m-1} h(x,t)\, u \\ & + \frac{\partial^m h(x,t)}{\partial t^m} + \mathcal{O}(h(x,t)) + \alpha_m\big(\psi_{m-1}(x,t)\big) \geq 0 \Big\}. \end{aligned} \tag{2.19}$$

which is affine in the control input and therefore preserves convexity of the resulting optimization problem.

At each time instant, the control input is obtained by solving an online optimization problem that balances performance and safety, leading to the Quadratic Program

$$\begin{aligned} u^* = \arg\min_{u \in \mathcal{U}} \quad & \|u - u_{\text{des}}\|^2 \\ \text{subject to} \quad & u \in \mathcal{K}_{\text{hocbf}}(x), \end{aligned} \tag{2.20}$$

where $\mathcal{K}_{\text{hocbf}}(x)$ denotes the state-dependent set of admissible control inputs induced by the HOCBF constraint, as defined in (2.19).

However, when input constraints or actuator saturations are present, the admissible control set is effectively given by the intersection $\mathcal{K}_{\text{hocbf}}(x) \cap \mathcal{U}$. In this case, it is possible for the HOCBF constraint to demand control actions that exceed the physical capabilities of the system, causing the intersection to become empty. As a consequence, the quadratic program becomes infeasible and the forward invariance of the safe set can no longer be guaranteed.

2.3 Viability Theory

Viability theory provides a rigorous mathematical framework for the analysis of dynamical systems subject to state and input constraints [17, 27]. Originally introduced by Aubin in the context of differential inclusions and set-valued analysis, viability theory shifts the focus of control from stabilization or trajectory tracking to the characterization of admissible system evolutions under constraints.

At the core of the theory lies the concept of the viability kernel, defined as the set of initial states from which there exists at least one admissible control strategy that keeps the system within the prescribed constraint set for all future times [17]. From this perspective, safety is not a property of the current state alone, but a global property of the system's future evolution. In particular, constraint violations may arise not because of an unsafe present configuration, but due to the absence of any feasible continuation that respects physical and environmental limits.

This anticipative view of safety allows viability theory to formalize notions such as points of no return and capture basins, which characterize the loss or recovery of admissible behaviors under constraints. These concepts are especially relevant for safety-critical systems, where delayed or overly aggressive actions may irreversibly compromise future feasibility.

While initially developed in a purely mathematical setting, viability theory has found applications across economics, biology, and engineering systems [17]. More recently, it has emerged as a fundamental tool for reasoning about admissible behaviors in autonomous and robotic systems, particularly in the presence of bounded control authority, dynamic constraints, and uncertainty [28, 29].

Consider a generic dynamical system described by the differential inclusion

$$\dot{x}(t) \in F(x(t)), \tag{2.21}$$

where $x(t) \in \mathbb{R}^n$ denotes the system state and $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is a set valued map representing the set of admissible velocities. Such inclusions naturally arise when control inputs are constrained or when uncertainty and disturbances are modeled explicitly [20].

Let $\mathcal{F} \subset \mathbb{R}^n$ be a closed set representing the admissible state space, defined by constraints on the state and its derivatives, for instance

$$x_{\min} \leq x \leq x_{\max}, \qquad \dot{x}_{\min} \leq \dot{x} \leq \dot{x}_{\max}. \tag{2.22}$$

In applications such as autonomous driving or robotic manipulation, these constraints may represent bounds on position, velocity, acceleration, or higher order derivatives induced by physical limitations or safety requirements.

In addition to guaranteeing indefinite constraint satisfaction, many control problems require ensuring that the system can reach a desired target set in finite time while remaining within the admissible set. This notion is captured by the concept of the capture basin,

introduced in the viability literature as a generalization of reachability under constraints [20, 28].

Let:

- $\mathcal{F} \subset \mathbb{R}^n$ be the admissible set,
- $\mathcal{Q} \subset \mathcal{F}$ be a target set,
- $S(x_0)$ denote the set of admissible evolutions starting from x_0, i.e., all trajectories $x(\cdot)$ such that $x(0) = x_0$ and $\dot{x}(t) \in F(x(t))$.

Definition 5 (Capture Basin). *The capture basin of $\mathcal{Q}$ with respect to $\mathcal{F}$, denoted by* $\mathrm{Capt}_S(\mathcal{F}, \mathcal{Q})$*, is defined as*

$$\begin{aligned}\mathrm{Capt}_S(\mathcal{F}, \mathcal{Q}) = \{x_0 \in \mathcal{F} \mid \exists x(\cdot) \in S(x_0),\ \exists T < +\infty \ s.t. \\ x(t) \in \mathcal{F}\ \forall t \in [0, T],\ \textit{and}\ x(T) \in \mathcal{Q}\ \}\,.\end{aligned} \tag{2.23}$$

The capture basin therefore contains all initial states from which the system can be steered toward the target set without violating the admissible constraints. This notion is particularly relevant for safety critical maneuvers such as emergency stopping, intersection crossing, or collision avoidance.

The viability kernel can be interpreted as a limiting case of the capture basin, where the objective is not to reach a target set in finite time, but to remain indefinitely within the admissible set [20].

Definition 6 (Viability Kernel). *Given an admissible set $\mathcal{F}$, the viability kernel $\mathcal{V}(\mathcal{F})$ is defined as*

$$\mathcal{V}(\mathcal{F}) := \{x_0 \in \mathcal{F} \mid \exists x(\cdot) \in S(x_0) \textit{ such that } x(t) \in \mathcal{F}\ \forall t \geq 0\}\,. \tag{2.24}$$

Any initial condition outside the viability kernel, while possibly satisfying the constraints instantaneously, is guaranteed to violate them in finite time, regardless of the applied control strategy. As such, the viability kernel provides a precise characterization of the set of safe initial conditions.

From a geometric perspective, the viability kernel corresponds to the region of the state space where the admissible vector field points inward or remains tangent to the boundary of the constraint set, in accordance with Nagumo type conditions [19]. From a dynamical standpoint, viability theory introduces the fundamental concept of anticipation: the safety of a state depends not only on its instantaneous admissibility, but on the existence of at least one future evolution compatible with the constraints.

For nonlinear systems with moderate to high state dimension, the exact computation of the viability kernel is generally intractable. Classical numerical methods based on state space discretization or level set techniques suffer from the curse of dimensionality, with computational complexity growing exponentially with the dimension of the system [28].

2.4 Differential Flatness

Differential flatness is defined for nonlinear control systems of the form

$$\dot{x} = f(x, u), \tag{2.25}$$

where $x \in \mathbb{R}^n$ denotes the system state and $u \in \mathbb{R}^m$ the control input, with f assumed sufficiently smooth.

Definition 7 (Differential Flatness)**.** *In general terms, such a system is said to be (locally) differentially flat if there exists a set of so-called flat outputs*

$$y = \varphi_v(x, u, \dot{u}, \ldots, u^{(p)}), \tag{2.26}$$

with dim(y)=m, that are differentially independent, and such that all components of the state and the input can be expressed as algebraic functions of the flat outputs and a finite number of their time derivatives:

$$x = \varphi_x(y, \dot{y}, \ldots, y^{(q_x)}), \qquad u = \varphi_u(y, \dot{y}, \ldots, y^{(q_u)}). \tag{2.27}$$

The key feature of this definition is that no integration is required: the system dynamics are completely parameterized by $y(t)$ through explicit differential relations.

This property leads to several practical advantages. As emphasized by Murray, Rathinam, and Sluis [30], flat systems naturally fit into a two degree of freedom design paradigm. In this approach, the control problem is split into two distinct phases:

1. generation of a feasible nominal trajectory for the nonlinear system model;
2. robust regulation around that trajectory using appropriate stabilizing controllers.

For differentially flat systems, the first phase becomes particularly simple: motion planning can be performed directly in the flat output space, reducing the trajectory generation problem to algebraic computations and to constraints on geometric quantities such as curvature or higher order derivatives.

Flat outputs are not arbitrary combinations of state variables, but usually correspond to geometric points, angles, or elementary kinematic quantities.

From an application viewpoint, differential flatness has proven to be effective in many robotics contexts:

- motion planning for mobile robots, with or without trailers;
- lifting and transportation systems such as cranes;
- chains of coupled rigid bodies.

For all these systems, flatness allows the transformation of input constraints into equivalent constraints in the output space. For example, limitations on force magnitude and

direction typically become bounds on the curvature or on higher order derivatives of the flat output trajectory.

A fundamental theoretical result states that every differentially flat system is also dynamically feedback linearizable on an open dense set of the state space. In the case of single input systems, such linearization can be achieved without explicit time dependence and without the need for dynamic extensions, whereas multi input systems often require specific relationships among mechanical parameters.

2.5 Pairwise Control Barrier Functions for Collaborative Safety

In multi-agent systems, safety is typically enforced by assigning to each agent an individual Control Barrier Function (CBF) that guarantees forward invariance of its corresponding safe set. Under this paradigm, each agent independently enforces its own safety set, ensuring that trajectories remain within predefined admissible regions.

However, when multiple agents operate in a shared environment, safety constraints often become inherently relational. In many practical scenarios, safety does not depend solely on the absolute state of a single agent, but rather on the relative configuration among agents. Examples include collision avoidance, cooperative manipulation, shared resource allocation, and infrastructure-assisted navigation.

To address this limitation, the concept of pairwise Control Barrier Functions has been introduced in the context of heterogeneous and collaborative multi-agent systems [31, 32]. In particular, Nguyen, Jabbari, and Egerstedt proposed a compositional framework in which safety sets can be dynamically reshaped through mutualistic or collaborative interactions among agents. Their work formalizes the idea that the safe set of an agent can be expanded, restricted, or otherwise modified through structured pairwise interactions, while still preserving forward invariance guarantees under suitable control laws.

A pairwise Control Barrier Function introduces an additional term that explicitly encodes how the presence or behavior of agent j influences the admissible configurations of agent i. The resulting formulation provides a systematic and modular approach to embedding relational safety constraints into optimization-based controllers, while retaining the convexity and real-time feasibility properties characteristic of standard CBF-based quadratic programs.

Consider a collection of N agents evolving in a common workspace. Each agent i is characterized by a state vector $x_i \in \mathbb{R}^{n_i}$ and is associated with an individual safe set defined as

$$C_i = \{x_i \in D_i \subset \mathbb{R}^{n_i} \mid h_i(x_i) \geq 0\}, \tag{2.28}$$

where $h_i : D_i \to \mathbb{R}$ is a Control Barrier Function ensuring forward invariance of C_i under suitable admissible control inputs.

This formulation assumes that safety can be fully characterized as a function of the

individual state x_i. While sufficient in decoupled systems, this assumption becomes restrictive when safety depends on the presence or behavior of other agents.

To capture relational safety properties, we introduce a pairwise barrier function

$$h_{ij}(x_i, x_j), \tag{2.29}$$

which models how the state of agent j influences the safety of agent i. Unlike the individual barrier $h_i(x_i)$, the function h_{ij} depends explicitly on both x_i and x_j, thereby encoding interaction constraints such as collision avoidance, cooperative support, or mutualistic behaviors.

The safety of agent i relative to agent j is then characterized through the composite barrier function

$$H_i(x_i, x_j) = h_i(x_i) + h_{ij}(x_i, x_j). \tag{2.30}$$

The associated extended safe set is defined as

$$C_{ij} = \{x_i \in D_i \mid \exists x_j \in D_j \text{ such that } H_i(x_i, x_j) \geq 0\}. \tag{2.31}$$

This construction modifies the admissible state space of agent i by incorporating information about agent j. The individual safe set C_i is therefore not replaced, but augmented through composition.

The composite barrier H_i can be interpreted as a mechanism for dynamically reshaping the safe set of agent i. Depending on the structure of h_{ij}, the safe region may:

- be restricted (e.g., in collision avoidance scenarios),
- be enlarged (e.g., when collaboration enables new feasible configurations),
- be shifted in the state space,
- or experience a change in its geometric properties.

In particular, when collaboration enables agent i to safely access regions that would otherwise be forbidden, the pairwise term h_{ij} acts as a compensatory contribution that relaxes the constraint imposed by h_i. The safety condition

$$H_i(x_i, x_j) \geq 0 \tag{2.32}$$

thus defines a relational safe region parametrized by the state of agent j.

To ensure safety under the composite barrier, the CBF condition is imposed on H_i. Assuming control-affine dynamics, forward invariance of the extended safe set can be guaranteed by enforcing

$$\frac{\partial H_i}{\partial x_i}\dot{x}_i + \frac{\partial H_i}{\partial x_j}\dot{x}_j \geq -\alpha\big(H_i(x_i, x_j)\big), \tag{2.33}$$

for some extended class $\mathcal{K}$ function α.

If the evolution of x_j is known, measurable, or bounded, this condition can be incorporated into the control design of agent i. In decentralized implementations, x_j may be

obtained through sensing or communication, allowing the composite constraint to remain compatible with standard optimization-based safety filters.

The pairwise construction naturally extends to the multi-agent case. For agent i, one may define

$$H_i(x_1, \ldots, x_N) = \sum_{j=1}^{N} h_{ij}(x_i, x_j), \tag{2.34}$$

which aggregates the pairwise safety contributions of all other agents.

This additive structure preserves modularity and scalability. When each h_{ij} induces affine constraints in the control input of agent i, the overall safety enforcement problem can be formulated as a single convex Quadratic Program, maintaining computational tractability even in large-scale systems.

Pairwise Control Barrier Functions provide a compositional and modular framework for modeling collaborative safety in multi-agent systems. By combining individual and relational barrier functions, the framework enables the systematic encoding of interaction-dependent constraints, the dynamic reshaping of safe sets based on shared information, and the scalable integration of such constraints into optimization-based control architectures.

This formulation is particularly well suited to scenarios in which safety depends on relative configurations and inter-agent information exchange, including cooperative robotics, connected autonomous vehicles, and infrastructure-assisted control systems.

Chapter 3

Kinematic Control with CBFs

This chapter introduces the first core component of the control architecture developed in this thesis, focusing on a constraint-based control framework for autonomous vehicles operating in structured urban environments, where safety and compliance with traffic regulations must be enforced online.

Urban driving cannot be reduced to collision avoidance alone: an autonomous vehicle must also respect lane boundaries, manage overtaking, handle intersections, and enforce right-of-way in the presence of heterogeneous agents and time-varying road contexts. Although these requirements are diverse in nature, they can be interpreted within a common control perspective, namely as constraints that restrict the admissible evolution of the vehicle state. The key idea of this chapter is to treat urban driving as a constrained dynamical system and to encode both safety and traffic rules directly at the control level through Control Barrier Functions. CBFs are by now a well-established tool in the control and robotics literature for ensuring forward invariance of safe sets via real-time optimization [33, 34]. In autonomous driving, they have been adopted extensively as safety filters, with a predominant focus on collision avoidance and related safety tasks (e.g., adaptive cruise control) [15]. Here, their role is deliberately extended: rather than acting only as virtual barriers around obstacles, barrier functions are used as a unified mathematical language to encode urban traffic regulations, including lane keeping, overtaking permissions, stopping behaviors near intersections, and priority management.

By adopting this perspective, the proposed architecture does not rely on discrete switching among maneuver-specific controllers; instead, the vehicle continuously solves a constrained optimization problem that minimally modifies a nominal command while guaranteeing that all currently relevant constraints remain satisfied. Each traffic rule corresponds to a specific constraint in the optimization, and the resulting behavior emerges from their interaction in a coherent and interpretable way. At this stage of the thesis, we employ a kinematic bicycle model to emphasize clarity and tractability in the formulation of urban constraints. The kinematic choice enables a direct geometric interpretation of road rules and allows most constraints to be enforced through relative-degree-one barrier conditions.

A distinctive aspect of the framework is that the driving behavior can be shaped through

the parameters of the barrier functions. By tuning these parameters, it is possible to modulate how strongly the controller reacts as the system approaches the boundary of the safe set, thus obtaining more conservative or more assertive driving styles without altering the overall structure of the control architecture. This offers a principled mechanism for behavioral personalization while preserving formal constraint satisfaction.

3.1 Kinematic Bicycle Model and Virtual Control Point

We consider a four-wheeled ground vehicle operating in the automotive context. To capture the essential planar motion of the vehicle while maintaining a tractable model for control design, we adopt the well-established kinematic bicycle model with front-da steering. Although the physical system consists of four wheels, the bicycle model provides an equivalent representation obtained by lumping the two front wheels into a single virtual steerable wheel and the two rear wheels into a single fixed wheel. This modeling abstraction is justified under the assumption of pure rolling and no lateral slip, conditions that are typically satisfied at low to moderate velocities and in nominal urban driving scenarios [35]. The kinematic bicycle model therefore captures the fundamental nonholonomic constraint imposed by wheel rolling without slipping, namely that the velocity of the vehicle at the contact point with the ground is aligned with the wheel orientation. As a result, the model accurately describes the evolution of the vehicle position and heading angle in the plane while remaining sufficiently simple to enable analysis, trajectory planning, and control synthesis.

We define the state of the kinematic model as

$$x := (x_1, x_2, x_3, x_4), \tag{3.1}$$

where x_1 and x_2 represent the vehicle planar position (X, Y), x_3 denotes the vehicle orientation with respect to the global reference frame, i.e., the yaw angle θ, and x_4 is the front steering angle γ.

The kinematic control inputs are

$$u := (v, \omega), \tag{3.2}$$

where v is the longitudinal velocity along the vehicle body-fixed axis X_v, and ω is the steering rate, i.e., the time derivative of the steering angle.

The kinematic model is given by

$$\begin{aligned} \dot{x}_1 &= v\cos(x_3), \\ \dot{x}_2 &= v\sin(x_3), \\ \dot{x}_3 &= \frac{v}{J}\tan(x_4), \\ \dot{x}_4 &= \omega, \end{aligned} \tag{3.3}$$

where $J > 0$ denotes the wheelbase, i.e., the distance between the rear and front axles see Fig.3.1.

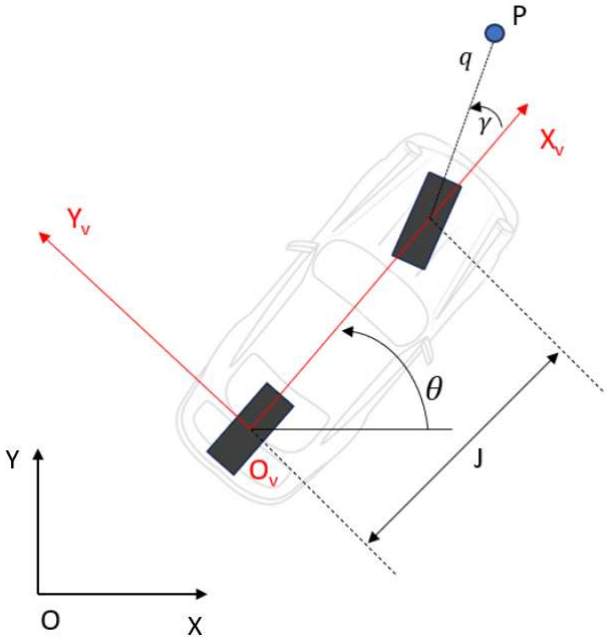


Figure 3.1: Kinematic Model

The nonholonomic constraint implicitly encoded in (3.3) makes the direct enforcement of geometric constraints such as lane boundaries, obstacles, or right-of-way regions non-trivial. To overcome this limitation, a virtual control point P is introduced.

3.1.1 Definition of the Virtual Point P

In order to enable input output state feedback linearization, we introduce a suitably defined output whose planar dynamics can be directly influenced by the control inputs. Rather than considering the vehicle center of mass or the rear axle position, we define a virtual point located ahead of the vehicle. This construction allows the system to exhibit a well-defined vector relative degree and facilitates the derivation of a linear input output relationship.

We define a virtual point P in front of the vehicle and is selected so that its planar dynamics are directly controllable. Specifically, its coordinates are defined as

$$\begin{aligned} x_{P,1} &= x_1 + J\cos(x_3) + q\cos(x_3 + x_4), \\ x_{P,2} &= x_2 + J\sin(x_3) + q\sin(x_3 + x_4), \end{aligned} \tag{3.4}$$

where $q > 0$ is a design parameter determining the distance of point P from the front axle along the steering direction.

The parameter q determines the look-ahead distance and affects the conditioning of the mapping matrix $T(x_3, x_4)$. A strictly positive value is required, since $q = 0$ removes the dependence of $\dot{x}_P$ on ω and makes the inversion in (3.11) ill-posed. Larger values of q increase anticipation and may lead to smoother corrections, whereas very small values reduce preview capability and can compromise numerical robustness. In this work, q is treated as a tuning parameter chosen to balance responsiveness, smoothness, and robust

invertibility of $T(x_3, x_4)$ over the admissible steering range.

Defining

$$x_P := \begin{bmatrix} x_{P,1} \\ x_{P,2} \end{bmatrix} \in \mathbb{R}^2, \tag{3.5}$$

the position of the virtual point P is uniquely determined by the vehicle kinematic state.

3.1.2 Relation Between Vehicle Inputs and Virtual Point Velocity

By differentiating (3.4) with respect to time and substituting the kinematic model (3.3), a linear relationship between the kinematic inputs (v, ω) and the velocity of the virtual point $(\dot{x}_{P,1}, \dot{x}_{P,2})$ is obtained:

$$\begin{bmatrix} \dot{x}_{P,1} \\ \dot{x}_{P,2} \end{bmatrix} = T(x_3, x_4) \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{3.6}$$

The transformation matrix $T(x_3, x_4) \in \mathbb{R}^{2\times 2}$ is explicitly given by

$$T(x_3, x_4) = \begin{bmatrix} \cos(x_3) - \tan(x_4)\Big(\sin(x_3) + \dfrac{q}{J}\sin(x_3 + x_4)\Big) & -q\sin(x_3 + x_4) \\ \sin(x_3) + \tan(x_4)\Big(\cos(x_3) + \dfrac{q}{J}\cos(x_3 + x_4)\Big) & q\cos(x_3 + x_4) \end{bmatrix}. \tag{3.7}$$

At this stage, we introduce the *virtual planar input*

$$u_P := \begin{bmatrix} v_{P,1} \\ v_{P,2} \end{bmatrix} := \begin{bmatrix} \dot{x}_{P,1} \\ \dot{x}_{P,2} \end{bmatrix}. \tag{3.8}$$

Accordingly, (3.6) can be compactly written as

$$u_P = T(x_3, x_4)\, u. \tag{3.9}$$

To employ u_P as a control variable on which Control Barrier Functions are imposed it must be possible to recover the physical input $u = (v, \omega)$ from u_P. This requires the matrix $T(x_3, x_4)$ to be invertible.

In practical vehicle operation, the steering angle x_4 is physically bounded and typically satisfies

$$x_4 \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \tag{3.10}$$

which avoids singularities of $\tan(x_4)$ and guarantees local invertibility of the transformation along the trajectories of interest.

Assuming $T(x_3, x_4)$ to be invertible, the physical input can be reconstructed as

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = T(x_3, x_4)^{-1} \begin{bmatrix} v_{P,1} \\ v_{P,2} \end{bmatrix}. \tag{3.11}$$

This step is crucial: the entire safety architecture is designed by enforcing constraints on the motion of the virtual point P (i.e., on u_P), while the actual vehicle inputs (v, ω) are obtained through the inverse transformation (3.11). The same construction can be straightforwardly adapted to a front-wheel-drive kinematic model by appropriately modifying the transformation matrix T.

3.1.3 Equivalent Single-Integrator Dynamics

The introduction of the virtual input u_P allows the dynamics of the virtual point to be expressed as a single integrator in the plane:

$$\dot{x}_P = u_P, \tag{3.12}$$

where $x_P \in \mathbb{R}^2$ is the state and $u_P \in \mathbb{R}^2$ is the planar velocity input.

This representation effectively removes the nonholonomic constraint from the safety formulation, enabling the direct and intuitive enforcement of geometric constraints such as lanes, obstacles, forbidden regions, and stop zones via Control Barrier Functions.

In the absence of constraints, a simple nominal control law to steer the virtual point toward a (possibly time-varying) target position $x_{tg}(t) \in \mathbb{R}^2$ can be defined through a proportional feedback action augmented with a feedforward velocity term:

$$u_{P,\text{des}} = \dot{x}_{tg}(t) + \xi\,(x_{tg}(t) - x_P), \tag{3.13}$$

with $\xi > 0$.

In the remainder of this work, the nominal input (3.13) is minimally adjusted by solving a Control Barrier Function constrained quadratic program, resulting in the safe control input $u_P^\star$.

The corresponding vehicle kinematic inputs (v, ω) are subsequently recovered via (3.11). For clarity, a block-diagram representation of the overall scheme is provided (Fig.).

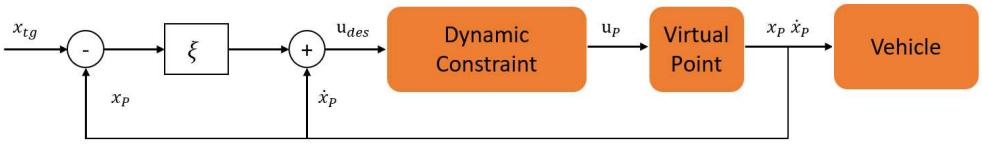


Figure 3.2: Constraint Framework

3.2 CBFs for Lane Keeping, Overtaking, Obstacle Avoidance, and Right-of-Way

This section presents in a comprehensive manner the Control Barrier Functions employed to model safety constraints and urban traffic rules. All constraints are defined on the

dynamics of the virtual point P, which is described by the single-integrator system

$$\dot{x}_P = u_P, \tag{3.14}$$

obtained through the kinematic transformation introduced before.

Thanks to this formulation, all barrier functions exhibit relative degree one with respect to the input u_P, allowing the direct application of the standard Control Barrier Function formalism.

3.2.1 Lane Keeping with Solid Lane Boundaries

Consider a lane delimited by two solid boundaries, one on the right and one on the left of the vehicle. Let $x_{L_i} \in \mathbb{R}^2$, with $i \in \{r, l\}$, denote the point on the i-th lane boundary closest to the virtual point x_P. The corresponding barrier function is defined as

$$h_{L_i}(x_P) := \|x_P - x_{L_i}\|^2 - d_{L_i}^2, \tag{3.15}$$

where $d_{L_i} > 0$ represents the minimum safety distance from the lane boundary, selected according to the lateral dimensions of the vehicle.

Differentiating (3.15) along the virtual point dynamics yields

$$\dot{h}_{L_i}(x_P, u_P) = \frac{\partial h_{L_i}}{\partial x_P}\dot{x}_P = 2(x_P - x_{L_i})^\top u_P. \tag{3.16}$$

The corresponding barrier condition is therefore given by

$$2(x_P - x_{L_i})^\top u_P \geq \lambda_{L_i} h_{L_i}(x_P), \qquad i \in \{r, l\}, \tag{3.17}$$

where $\lambda_{L_i} > 0$ is a design parameter. This constraint guarantees that, if the virtual point starts within the lane boundaries, it will never cross them.

3.2.2 Dashed Lane Boundary and Overtaking

In two-lane roads, the central dashed line can be temporarily crossed to allow overtaking maneuvers. The CBF associated with the dashed lane boundary is defined as

$$h_{DL}(x_P) := \|x_P - x_{DL}\|^2 - d_{DL}^2, \tag{3.18}$$

where x_{DL} denotes the closest point on the dashed line and d_{DL} is the nominal separation distance.

To allow a controlled and temporary violation of this constraint, a slack variable $\eta \geq 0$ is introduced, leading to the relaxed barrier condition

$$2(x_P - x_{DL})^\top u_P \geq \lambda_{DL} h_{DL}(x_P) - \zeta\eta, \tag{3.19}$$

where $\zeta \in \{0, 1\}$ enables the slack variable only when the presence of an obstacle or a slower vehicle to be overtaken is detected.

The penalization of η in the cost function of the optimization problem ensures that the dashed lane boundary is crossed only when strictly necessary.

3.2.3 Obstacle Avoidance

Both static and dynamic obstacles are modeled as circular regions in the plane. Consider an obstacle with center $x_{ca} \in \mathbb{R}^2$ and nominal radius R_{ca}. The associated barrier function is defined as

$$h_{ca}(x_P) := \|x_P - x_{ca}\|^2 - R_{ca}^2. \tag{3.20}$$

The derivative along the virtual point dynamics is given by

$$\dot{h}_{ca}(x_P, u_P) = 2(x_P - x_{ca})^\top u_P, \tag{3.21}$$

leading to the barrier condition

$$2(x_P - x_{ca})^\top u_P \geq \lambda_{ca} h_{ca}(x_P). \tag{3.22}$$

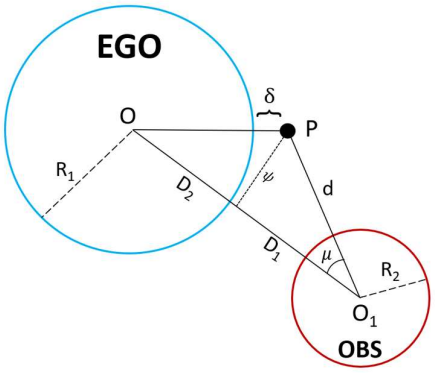


Figure 3.3: Geometric inflation of the vehicle model

Geometric inflation of the vehicle model Since neither the kinematic model nor the virtual point explicitly accounts for the physical dimensions of the vehicle, the obstacle radius R_{ca} is suitably inflated to incorporate the actual vehicle geometry. In particular, the minimum admissible distance between the vehicle and the obstacle is required to satisfy

$$D > D_1 + D_2, \tag{3.23}$$

where D_1 and D_2 depend on the vehicle geometry and on the relative position of the virtual point.

Let d denote the distance between the virtual point and the obstacle center, and let μ be the relative angle(see Fig. 3.3). Defining

$$\psi = d \sin(\mu), \tag{3.24}$$

the non-contact condition can be expressed as

$$\sqrt{d^2 - \psi^2} + \sqrt{(R_1 + \delta)^2 - \psi^2} > D, \tag{3.25}$$

where R_1 denotes the equivalent radius of the vehicle and δ is the distance between the virtual point and the frontmost point of the vehicle. This formulation guarantees collision avoidance while retaining a point-mass representation of the vehicle.

3.2.4 Stop and Right-of-Way at Intersections

Stop and right-of-way situations are modeled using an elliptical-shaped CBF positioned near the intersection area. The barrier function is defined as

$$\begin{cases} h_S(x_P) = \frac{(x_{P,1} - x_{obs,1})^2}{a^2} + \frac{(x_{P,2} - x_{obs,2})^2}{b^2} - 1, & \text{if } \dfrac{dist}{c} < dist_{sf}(v), \\ h_S(x_P) = 0, & \text{if } \dfrac{dist}{c} > dist_{sf}(v), \end{cases} \tag{3.26}$$

where $a, b \in \mathbb{R}^+$ define the ellipse in canonical form $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$.

The derivative of h_S along the virtual point dynamics is given by

$$\dot{h}_S(x_P, u_P) = \frac{2(x_{P,1} - x_{obs,1})}{a^2} u_{P,1} + \frac{2(x_{P,2} - x_{obs,2})}{b^2} u_{P,2}. \tag{3.27}$$

When the constraint is active, the corresponding barrier condition reads

$$\frac{2(x_{P,1} - x_{obs,1})}{a^2} u_{P,1} + \frac{2(x_{P,2} - x_{obs,2})}{b^2} u_{P,2} \geq \lambda_S h_S(x_P). \tag{3.28}$$

The decision to activate the CBF depends on several factors:

- The safety distance ($dist_{sf}$) which is set as a safety buffer between the approaching vehicle and the intersection; this will depend on the vehicle's speed (v).
- The actual distance between the approaching vehicle and the intersection $dist$
- A confidence factor denoted as c.

In this case, it is not possible to use λ_S to adjust the driving style, as it solely impacts the stopping distance from the edge of the road intersection. Higher λ_S values result in a shorter stopping distance, while lower values lead to a longer stopping distance. Instead the confidence factor, indicated by $c > 1$, can be used to define the driving style. Indeed, as c decreases, it corresponds to a more aggressive driving behavior from the autonomous vehicle, which will proceed through the intersection even if there is an approaching vehicle. The minimum possible value is 1 because this way the vehicle will always adhere to the minimum safety distance, ensuring the avoidance of collisions between the two vehicles.

3.2.5 Unified Optimization Problem

All the constraints introduced in the previous sections are combined into a single quadratic optimization problem, which is solved online at each control step:

$$\begin{aligned} u_P^\star = \arg\min_{u_P,\eta} \quad & \|u_P - u_{P,\text{des}}\|^2 + \zeta\eta^2 \\ \text{subject to} \quad & 2(x_P - x_{L_i})^\top u_P \geq \lambda_{L_i} h_{L_i}(x_P), \quad i \in \{r, l\}, \\ & 2(x_P - x_{DL})^\top u_P \geq \lambda_{DL} h_{DL}(x_P) - \zeta\eta, \\ & 2(x_P - x_{ca})^\top u_P \geq \lambda_{ca} h_{ca}(x_P), \\ & \nabla h_S(x_P)^\top u_P \geq \lambda_S h_S(x_P), \\ & u_{P,\min} \leq u_P \leq u_{P,\max}. \end{aligned} \tag{3.29}$$

The objective function enforces minimal deviation from the nominal planar control input $u_{P,\text{des}}$, while the slack variable η is penalized to allow controlled and temporary relaxation of the dashed-lane constraint during overtaking maneuvers.

The bounds on u_P originate from physical limits on the kinematic inputs (v, ω) and are obtained through the inverse transformation of the matrix $T(x_3, x_4)$ introduced in Section 4.1. This guarantees that the optimized virtual input is always realizable by the underlying vehicle kinematics.

3.3 Simulation Results

This section presents the simulation results obtained to validate the kinematic control architecture based on Control Barrier Functions (CBFs) introduced in the previous chapters. The primary objective of these simulations is to demonstrate that the proposed framework is capable of managing complex urban driving scenarios while guaranteeing safety, stability, and compliance with traffic regulations in real time.

The validation process was structured in two progressive stages. In the first stage, the controller was implemented and tested in the MATLAB environment to ensure fast, reproducible, and modular verification of each component of the architecture. This phase enabled systematic tuning of the control parameters and facilitated an in-depth analysis of constraint activation mechanisms and numerical behavior of the quadratic programming formulation. In the second stage, the same control architecture was deployed in three-dimensional robotic simulation environments, namely Webots and CoppeliaSim, in order to assess its performance under more realistic dynamic and geometric conditions. This dual-level validation strategy ensures both theoretical consistency and practical applicability of the proposed method. The computer employed for these purpose is equipped with an Intel i7-10510U processor and 8 GB of RAM in its hardware configuration.

The simulated environment reproduces a structured urban context designed to capture the most representative challenges of autonomous driving. In particular, the scenario includes:

- a single-lane road delimited by solid lane boundaries,
- a two-lane road segment with a central dashed line,
- static circular obstacles and moving vehicles,
- an intersection regulated by right-of-way rules and a stop sign.

A representative overview of the scenario is reported in Fig. 3.4, where the vehicle trajectory, the virtual control point P, lane boundaries, obstacles, and intersection areas are highlighted. The structure of the environment is intentionally heterogeneous in order to simultaneously evaluate lane-keeping performance, overtaking maneuvers, collision avoidance, and intersection management within a unified control framework.

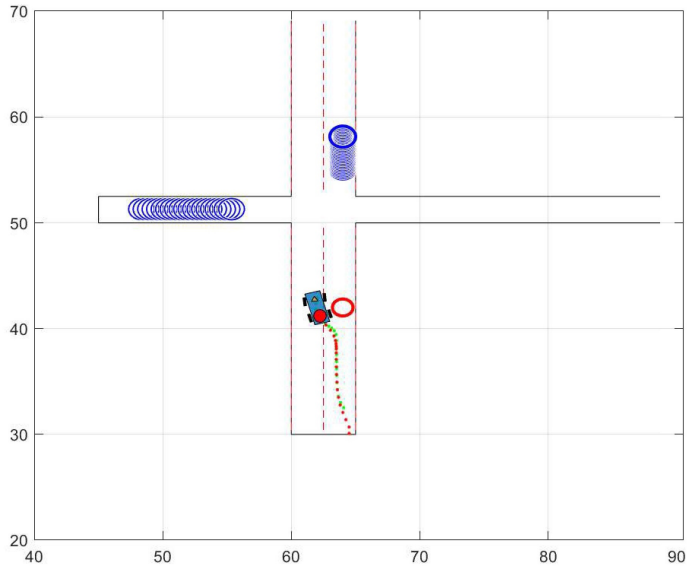


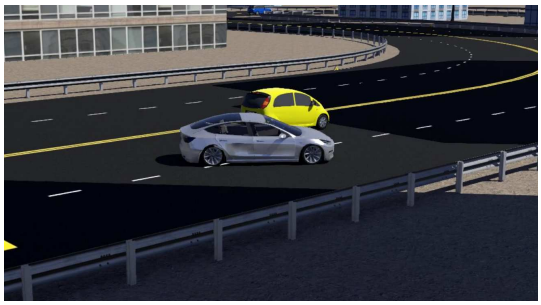
Figure 3.4: Representation of the simulation scenario in MATLAB„ where the red circle is the fixed obstacle while the blue circles represent the moving one.

In the lane-keeping scenario (Fig. 3.5), the vehicle maintains a safe and adaptive distance from the road boundaries. The safety margin is regulated by the parameter λ_L, which defines the rate at which the barrier function enforces constraint satisfaction. The results show that the vehicle trajectory remains smooth and free from oscillatory or chattering behavior, confirming the stability properties of the proposed formulation.

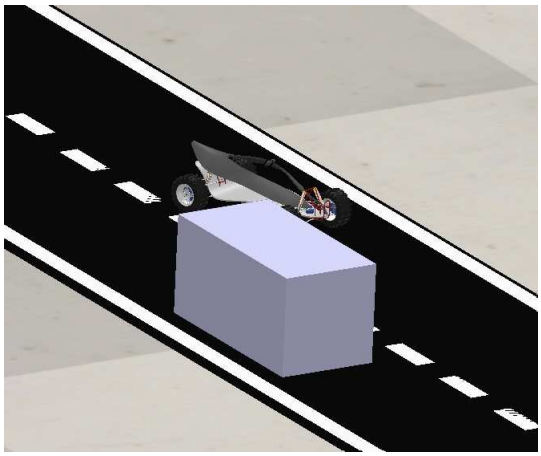
Figure 3.5: Lane-keeping scenario. The vehicle maintains a safe and adaptive distance from the road boundaries through the CBF-based lateral constraint. The safety margin is regulated by the parameter λ_L, ensuring smooth trajectory evolution without oscillatory behavior.

The framework also supports advanced maneuvers such as overtaking across dashed lane

markings (Fig. 3.6). In this case, the decision variable ζ enables selective activation of lateral constraints, while slack variables allow for temporary and bounded relaxation of barrier conditions. This mechanism permits controlled crossing of the dashed centerline to bypass slower obstacles, followed by a safe and stable return to the original lane once the maneuver is completed. Importantly, constraint relaxation remains limited and formally regulated, preserving overall safety guarantees.



(a) Overtaking maneuver in Webots.



(b) Overtaking maneuver in Coppelia.

Figure 3.6: Overtaking maneuver across dashed lane markings. The activation of the decision variable ζ and the use of slack variables allows temporary and bounded relaxation of lateral constraints, enabling safe obstacle bypassing and controlled reintegration into the original lane.

Intersection management and right-of-way compliance are handled using elliptical CBFs that model stop regions and priority constraints (Fig. 3.7). As the vehicle approaches a stop line, the barrier function progressively reduces the admissible forward velocity, ensuring smooth deceleration and a complete stop before entering the protected area. This behavior is achieved without requiring discrete switching logic, demonstrating the ability of the CBF framework to encode traffic rules as continuous constraints within the same optimization-based controller.

For a comprehensive visualization of the vehicle behavior, the reader is referred to the demonstration video https://doi.org/10.5281/zenodo.18755405 .

Figure 3.7: Intersection management with stop sign and right-of-way regulation. Elliptical Control Barrier Functions enforce progressive deceleration of the vehicle, guaranteeing a smooth and complete stop before the stop line without discrete switching logic.

3.4 Conclusions and future work

The simulations confirm that the proposed kinematic CBF based architecture provides an effective and computationally efficient framework for constraint-driven autonomous urban driving. By encoding traffic regulations, obstacle avoidance, lane-keeping, and right-of-way management directly within Control Barrier Functions, the vehicle behavior emerges from the solution of a single convex quadratic program, ensuring real-time feasibility.

The framework exhibits several desirable properties. First, it guarantees forward invariance of the defined safe sets, thereby ensuring formal compliance with safety constraints. Second, it is inherently modular: additional traffic rules or environmental constraints can be incorporated by introducing new barrier functions without altering the overall control structure. Third, the parametrization of the barrier functions enables a principled shaping of the driving behavior, allowing the same architecture to generate more conservative or more assertive driving styles through appropriate tuning of the design parameters.

Despite these advantages, the simulations also highlight intrinsic limitations associated with the adoption of a purely kinematic vehicle model. In particular, the proposed architecture assumes that the virtual-point velocity generated by the quadratic program can be instantaneously realized by the physical vehicle. While this assumption is reasonable at low speeds and under moderate maneuvers, it becomes increasingly unrealistic in the presence of actuator saturation, limited acceleration capabilities, steering rate constraints, or higher longitudinal velocities.

Under such conditions, the mismatch between the single-integrator virtual model and the true vehicle dynamics may lead to degraded tracking performance or, in critical scenarios, to infeasibility when safety constraints are mapped onto the physical system. Furthermore, although input bounds can be imposed at the optimization level, the kinematic formulation does not explicitly account for second-order dynamics. As a consequence, it cannot rigorously guarantee compatibility between safety constraints and physical acceleration limits, particularly in emergency braking, aggressive overtaking, or partially obstructed intersection scenarios.

These considerations motivate the extension of the proposed framework toward dynamic vehicle models, where acceleration, rather than velocity, is treated as the primary control input. In this setting, High-Order Control Barrier Functions are required to address the higher relative degree of the safety constraints, and viability-based bounds can be incorporated to ensure feasibility under actuator limitations.

The transition to a dynamic formulation, developed in Part 5 of this thesis, addresses the aforementioned limitations by integrating HOCBFs and viability constraints within a cascaded optimization architecture. This extension preserves the modular and safety-critical nature of the original CBF framework while ensuring physical consistency with realistic vehicle dynamics.

Chapter 4

Smart Infrastructure and Pairwise CBFs

The CBF-based architecture developed in the previous chapters provides a unified optimization framework for enforcing traffic rules and safety constraints in urban autonomous driving. By encoding lane boundaries, obstacle avoidance, and intersection management as barrier functions within a single convex Quadratic Program, the framework guarantees forward invariance of the defined safe sets while preserving real-time feasibility.

However, the original formulation relies exclusively on onboard perception systems. Although modern sensing technologies such as cameras, LiDAR, and radar provide increasingly accurate environmental reconstruction, they remain fundamentally constrained by physical and geometric limitations. In dense urban environments, occlusions caused by buildings, parked vehicles, vegetation, or complex road geometries restrict the vehicle's field of view. This limitation becomes particularly critical at unregulated intersections, where safe crossing depends on knowledge of cross-traffic conditions beyond the sensing horizon.

Within the CBF framework, incomplete perception translates into uncertainty in the evaluation of safety constraints. Since forward invariance must be guaranteed under all admissible conditions, the controller is forced to assume worst-case scenarios whenever environmental information is unavailable. As a result, intersection-related barrier functions remain active even in the absence of actual conflicting traffic, leading to conservative behaviors such as mandatory stopping before entering blind intersections. While this strategy preserves safety, it reduces operational efficiency and may contribute to unnecessary traffic congestion.

These limitations do not arise from the barrier-function methodology itself, but from restricted environmental awareness. The controller guarantees safety with respect to the information available to it; therefore, improving the quality and range of that information becomes essential for enhancing performance without compromising formal guarantees.

Vehicle-to-Infrastructure (V2I) communication provides a principled solution to this problem. By equipping critical locations with sensing units capable of monitoring ar-

eas beyond the vehicle's line of sight, additional state information can be communicated to approaching autonomous vehicles. Infrastructure sensors may detect the presence, velocity, and trajectory of cross-traffic participants, effectively extending the vehicle's perception horizon.

From a control theoretic standpoint, infrastructure can be interpreted as an auxiliary agent that supplies exogenous state information. Importantly, this additional information does not replace the vehicle's safety mechanism. The autonomous vehicle remains fully responsible for enforcing forward invariance of its safe sets. Instead, infrastructure data are incorporated as structured contributions that dynamically reshape the admissible region of operation.

To ensure consistency with the original framework, the integration of infrastructure information must preserve two fundamental properties: formal safety guarantees and convexity of the underlying optimization problem. These requirements naturally motivate the use of pairwise Control Barrier Functions, through which infrastructure sensing units are modeled as interacting agents whose states influence the vehicle's safety constraints. In this way, improved environmental awareness reduces conservativeness while maintaining a unified, optimization-based control architecture suitable for real-time implementation.

4.1 Pairwise Control Barrier Functions for V2I Collaboration

The integration of infrastructure sensing into our Control Barrier Function framework can be rigorously formalized through the notion of pairwise Control Barrier Functions introduced in the background chapter. In this context, the autonomous vehicle is modeled as agent i with state x_i, while the infrastructure sensing unit is modeled as agent j with state x_j. The state x_j does not necessarily correspond to a physical actuator-driven system, but rather to the measurable traffic state provided by the infrastructure (e.g., occupancy of an intersection, detected vehicle positions, or traffic flow conditions).

Let the autonomous vehicle be associated with an individual safe set

$$C_i = \{x_i \in D_i \subset \mathbb{R}^{n_i} \mid h_i(x_i) \geq 0\}, \tag{4.1}$$

where $h_i(x_i)$ encodes the nominal safety constraint related to intersection handling. In the absence of additional information, this barrier function enforces a conservative behavior, typically requiring the vehicle to stop before entering a blind intersection.

To incorporate infrastructure information, we introduce a pairwise barrier function

$$h_{ij}(x_i, x_j), \tag{4.2}$$

which captures how the infrastructure state influences the safety condition of the vehicle. The composite barrier function is then defined as

$$H_i(x_i, x_j) = h_i(x_i) + h_{ij}(x_i, x_j). \tag{4.3}$$

The extended safe set becomes

$$C_{ij} = \{x_i \in D_i \mid H_i(x_i, x_j) \geq 0\}, \tag{4.4}$$

where x_j is provided through V2I communication.

The function $h_{ij}(x_i, x_j)$ is designed such that it dynamically reshapes the admissible region of operation depending on the traffic conditions detected by the infrastructure. When the intersection is free, the pairwise term can relax the original constraint, effectively enlarging the safe set. Conversely, when conflicting traffic is detected, the pairwise term maintains or reinforces the conservative constraint already encoded in $h_i(x_i)$.

To guarantee safety, the composite barrier H_i must satisfy the standard CBF condition. Assuming a control-affine dynamics for the vehicle giien by the architeture in Section 3,

$$\dot{x}_i = f(x_i) + g(x_i)u_i, \tag{4.5}$$

forward invariance of C_{ij} is ensured by enforcing

$$\frac{\partial H_i}{\partial x_i}\big(f(x_i) + g(x_i)u_i\big) + \frac{\partial H_i}{\partial x_j}\dot{x}_j \geq -\alpha\big(H_i(x_i, x_j)\big), \tag{4.6}$$

for some extended class $\mathcal{K}$ function α.

In the present framework, the infrastructure state x_j is treated as exogenous information. Since it is measured and communicated to the vehicle, $\dot{x}_j$ can be either directly estimated or conservatively bounded. Importantly, the resulting constraint remains affine in the control input u_i, thereby preserving the convexity of the underlying Quadratic Program.

This formulation enables a structured integration of infrastructure data into the safety filter. The original optimization problem remains unified, and infrastructure collaboration manifests as a modification of the barrier constraint rather than as an external supervisory layer. Consequently, the vehicle retains full responsibility for safety enforcement, while benefiting from an expanded perception horizon that reduces unnecessary conservativeness.

4.1.1 Application to Blind and Partially Occupied Intersections

The pairwise formulation introduced in the previous section is now specialized to the intersection handling problem. In the baseline architecture, the vehicle is associated with an elliptical Control Barrier Function that enforces safe stopping before entering the intersection. This function is defined as

$$h_{in}(x_P) = \frac{(x_{P,1} - x_{c,1})^2}{c^2} + \frac{(x_{P,2} - x_{c,2})^2}{d^2} - 1, \tag{4.7}$$

where $x_c = (x_{c,1}, x_{c,2})$ is the center of the ellipse, (c, d) are its semi-axes and $x_P \in \mathbb{R}^2$ is the control point position. In the absence of external information, this constraint remains active, resulting in conservative behavior at blind intersections.

Within the V2I-assisted formulation, the infrastructure is modeled as agent j and provides information about the occupancy of the intersection. This information is incorporated through a pairwise Control Barrier Function $h_{in,c}(x_P, x_c)$, leading to the composite barrier

$$H_{in}(x_P, x_c) = h_{in}(x_P) + h_{in,c}(x_P, x_c). \tag{4.8}$$

Two relevant cases are considered.

Free Intersection. If the infrastructure certifies that the intersection is completely free, the pairwise contribution can be defined as a constant positive term

$$h_{in,c}(x_P, x_c) = 1, \tag{4.9}$$

$$H_{in}(x_P, x_c) = h_{in}(x_P) + 1. \tag{4.10}$$

Since the superlevel set of H_{in} is enlarged with respect to that of h_{in}, the admissible region expands to include the intersection area. The vehicle is therefore allowed to cross without stopping, while safety remains formally enforced through the composite barrier condition. For a visual representation of this concept, refer to the Fig. 4.1, which illustrates the case of a one-dimensional CBF $h_u = \frac{1}{2}x^2 - 2$ and the pairwise $h_{u,j} = \sigma$ with $\sigma > 0$ to create H_u in which safe set and workspace overlaps.

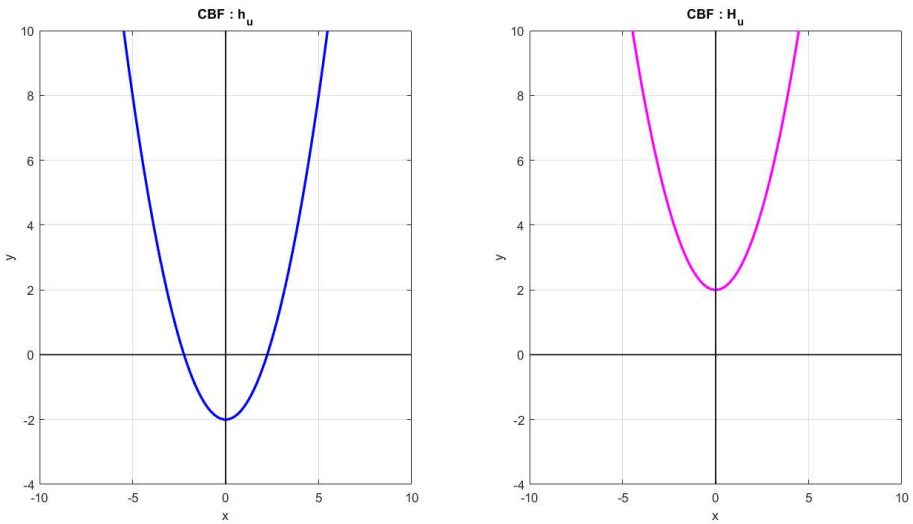


Figure 4.1: A parabolic CBF is represented here. In this case, the unsafe set along the x-axis is defined as the values between -2 and +2. To align the safe set with the workspace, it is sufficient for $\sigma > 2$. In this case, $h_{u,j} = 4$ was chosen.

Partially Occupied Intersection. When the infrastructure detects that only a portion of the intersection is occupied, a complete relaxation of the constraint would be unsafe. Instead, the pairwise barrier is defined as an additional ellipse,

$$h_{in,c}(x_P, x_c) = \frac{(x_{P,1} - x_{g,1})^2}{c_1^2} + \frac{(x_{P,2} - x_{g,2})^2}{d_1^2} - 1, \tag{4.11}$$

where $x_g = (x_{g,1}, x_{g,2})$ denotes the center of the occupied region and (c_1, d_1) are the corresponding semi-axes determined by the geometry of the detected traffic.

The composite barrier

$$H_{in}(x_P, x_c) = h_{in}(x_P) + h_{in,c}(x_P, x_c) \tag{4.12}$$

remains quadratic in x_P and can be expressed as an equivalent ellipse

$$H_{in}(x_P, x_c) = \frac{(x_{P,1} - x_{G,1})^2}{e^2} + \frac{(x_{P,2} - x_{G,2})^2}{f^2} - 1, \tag{4.13}$$

where the new center $x_G = (x_{G,1}, x_{G,2})$ and semi-axes (e, f) are obtained analytically as

$$\frac{1}{e^2} = \frac{1}{c^2} + \frac{1}{c_1^2}, \qquad \frac{1}{f^2} = \frac{1}{d^2} + \frac{1}{d_1^2}, \tag{4.14}$$

$$x_{G,1} = \frac{\frac{x_{c,1}}{c^2} + \frac{x_{g,1}}{c_1^2}}{\frac{1}{c^2} + \frac{1}{c_1^2}}, \qquad x_{G,2} = \frac{\frac{x_{c,2}}{d^2} + \frac{x_{g,2}}{d_1^2}}{\frac{1}{d^2} + \frac{1}{d_1^2}}. \tag{4.15}$$

These expressions correspond to the weighted combination of the two ellipses, where the weights are determined by the inverse squared semi-axes. The resulting barrier effectively shifts and reshapes the original stopping region. The admissible set excludes only the actually occupied portion of the intersection, while allowing traversal of the free segment as ca be seen in Fig.4.2.

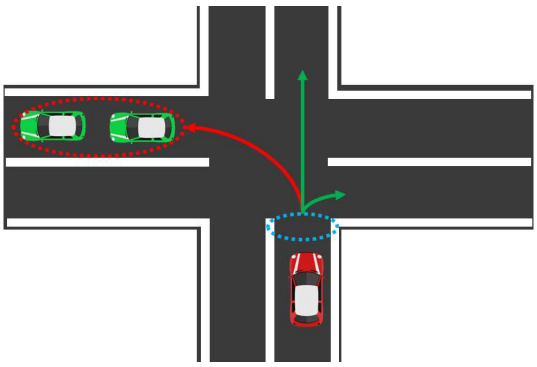


Figure 4.2: In this situation the autonomous vehicle can proceed in green path without any problem, in case it need to pursuit the red one it need the pairwise CBF

This construction preserves the convex quadratic structure of the constraint and ensures that the composite barrier condition can be seamlessly integrated into the same optimization-based safety filter. The vehicle therefore adapts its admissible region according to verified traffic occupancy, maintaining formal forward invariance guarantees

while reducing unnecessary conservativeness.

4.2 Optimization Problem with Infrastructure-Dependent Constraints

The integration of infrastructure information through pairwise Control Barrier Functions modifies the constraint structure of the optimization-based safety filter, while preserving its overall formulation. The autonomous vehicle control input is obtained by solving a Quadratic Program that minimizes the deviation from a desired input subject to a set of barrier-induced constraints.

Let u_i denote the control input of the vehicle and u_{des} the nominal control action generated by the reference tracking controller. In the baseline architecture, the safety filter solves

$$u_i^* = \arg\min_{u_i} \|u_i - u_{des}\|^2 \tag{4.16}$$

subject to a collection of affine constraints derived from the individual barrier functions $h_i(x_i)$ associated with lane keeping, obstacle avoidance, and intersection management.

With the introduction of infrastructure-assisted pairwise barriers, the intersection-related constraint is replaced by the composite barrier $H_i(x_i, x_j)$. The CBF condition becomes

$$\frac{\partial H_i}{\partial x_i}\big(f(x_i) + g(x_i)u_i\big) + \frac{\partial H_i}{\partial x_j}\dot{x}_j \geq -\alpha\big(H_i(x_i, x_j)\big). \tag{4.17}$$

Since x_j is treated as measured or communicated data, $\dot{x}_j$ can be either directly estimated or conservatively bounded. Crucially, the constraint remains affine in u_i because the control input appears only through the term $g(x_i)u_i$. Therefore, the modified safety filter can still be expressed as a convex Quadratic Program of the form

$$\begin{aligned} u_i^* = \arg\min_{u_i} \quad & \|u_i - u_{des}\|^2 \\ \text{subject to} \quad & A_i(x_i)u_i + b_i(x_i) \geq 0 \\ & A(x_i, x_j)u_i \geq b(x_i, x_j) \end{aligned} \tag{4.18}$$

where the matrices $A(\cdot)$ and vectors $b(\cdot)$ now depend on both the vehicle state and the infrastructure state and $A_i(x_i)u_i + b_i(x_i) \geq 0$ represent the other CBFs constraints in the affine form.

The presence of x_j introduces time-varying parameters into the constraints but does not alter the convexity of the optimization problem. As a result, the real-time solvability of the controller is preserved. Infrastructure collaboration therefore manifests as a parametric modification of the admissible input set rather than as an increase in optimization complexity.

An important implication of this formulation is that the architecture remains the same:

lane constraints, obstacle avoidance constraints, and infrastructure-assisted intersection constraints coexist within the same optimization problem. The safety guarantees continue to rely on forward invariance of the composite barrier functions, while efficiency improvements arise from the dynamic reshaping of the feasible set enabled by V2I communication.

This structural preservation of convexity and modularity is a key feature of the proposed approach, ensuring that the integration of infrastructure sensing enhances performance without compromising theoretical guarantees or computational tractability.

4.3 Simulation Setup and Validation

The proposed infrastructure-assisted CBF framework was validated through a two-stage simulation campaign. First, numerical simulations were conducted in MATLAB to evaluate the theoretical behavior of the controller and to compare it with the baseline architecture. Subsequently, the framework was implemented in the Webots robotic simulator to assess its performance in a more realistic and dynamic urban environment.

The urban scenario from Section 3 was reconstructed in Matlab including lane boundaries, static and dynamic obstacles, and an unregulated intersection. In order to better approximate the physical geometry of real vehicles, obstacles were modeled as ellipses rather than circles. This choice enables a more accurate representation of vehicle shape allowing the CBF constraints to reflect realistic spatial occupancy conditions. (Fig.4.3).

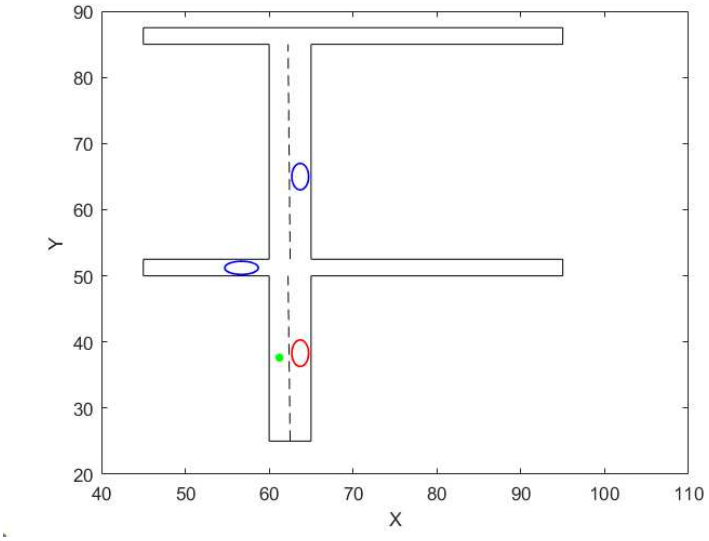
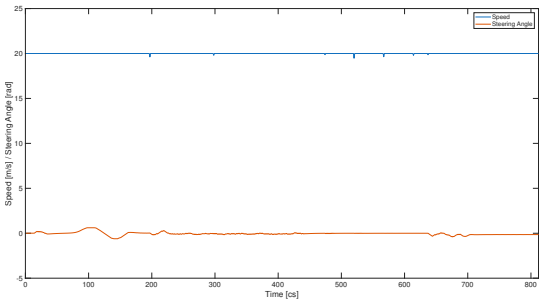


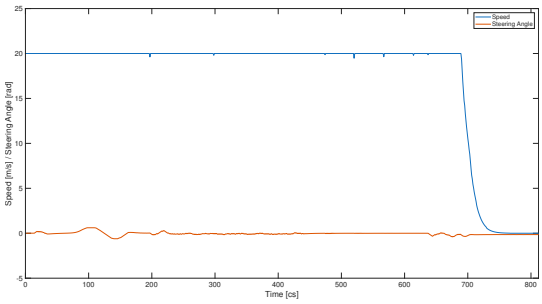
Figure 4.3: Representation of the simulation scenario in MATLAB.

In the absence of infrastructure information, the vehicle consistently adopted a conservative strategy when approaching blind intersections, enforcing a full stop before entering the conflict area. With the introduction of pairwise CBFs, the safe set was dynamically reshaped according to the communicated occupancy state of the intersection. As a result, when the intersection was free, the vehicle was able to proceed without unnecessary stopping, as illustrated in Fig. 4.4. In partially occupied scenarios, the vehicle instead

traversed only the free portion of the intersection while maintaining the safety constraints with respect to the occupied region. In both cases, small spikes can be observed in the velocity profiles. These are mainly due to the solver settings, and in particular to the convergence tolerance, which may cause the constraints to be rapidly activated and deactivated. Such effects do not compromise the overall stability of the system, but they could be further mitigated through a more accurate tuning of the solver parameters or by introducing suitable command-filtering mechanisms.



Intersection completely free. The infrastructure allows continuous motion without unnecessary stopping.



Occupied intersection, the vehicle safely stops before entering the conflict area, preserving safety guarantees.

Figure 4.4: Comparison between the behavior of the vehicle in the presence of a free intersection and an occupied one.

The solution time of the Quadratic Program remained on the order of milliseconds and was comparable to the baseline implementation, confirming that the addition of pairwise constraints did not compromise computational efficiency.

To further validate the practical feasibility of the approach, the controller was integrated within the Webots simulator. A small urban environment was reconstructed, including buildings causing occlusions, multiple intersections, and additional vehicles acting as dynamic traffic participants.

The V2I module provided intersection occupancy information to the autonomous vehicle through simulated communication channels. The vehicle successfully navigated the environment in real time, adapting its behavior according to the infrastructure-provided data. Blind intersections were crossed without unnecessary stops when safe, while appropriate stopping behavior was preserved when conflicting traffic was detected as see in Fig.4.5.

The experiments confirmed that the optimization problem remained solvable at each control cycle and that the unified architecture maintained stable and safe behavior throughout the simulation horizon. These results demonstrate that the proposed framework is not only theoretically consistent but also practically implementable in realistic urban scenarios.

Overall, the simulation campaign validates the core hypothesis of this chapter: infrastructure-assisted pairwise Control Barrier Functions can enhance operational efficiency while preserving formal safety guarantees and real-time feasibility, a video of the experiment is available at: https://doi.org/10.5281/zenodo.18763098 .

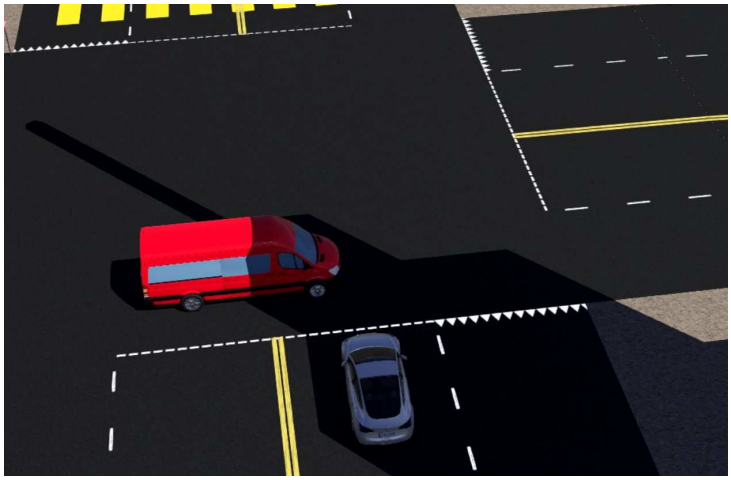


Figure 4.5: Webots simulation: the autonomous vehicle stops before entering an occupied intersection.

4.4 Conclusions and Future Perspectives

This chapter presented an extension of the unified Control Barrier Function framework to infrastructure-assisted autonomous driving scenarios. By leveraging pairwise Control Barrier Functions, Vehicle-to-Infrastructure communication was formally integrated into the optimization-based safety architecture previously developed in Section 3. The proposed formulation allows infrastructure sensing units to influence the admissible operating region of the autonomous vehicle through composite barrier functions, while preserving forward invariance guarantees and convexity of the underlying Quadratic Program.

The central contribution of this work lies in demonstrating that collaborative information can be incorporated without altering the structural properties of the controller. The optimization problem remains unified and computationally tractable, and safety enforcement continues to rely on formally defined barrier conditions. At the same time, the availability of externally communicated environmental data enables a dynamic reshaping of intersection-related constraints, reducing conservativeness in blind or partially occluded scenarios and improving operational efficiency.

Although the proposed framework establishes a consistent foundation for V2I collaboration, several research directions remain open. A natural extension concerns the integration of Vehicle-to-Vehicle communication within the same pairwise CBF formalism, enabling fully distributed multi-agent safety enforcement. In such scenarios, each vehicle could simultaneously act as both an autonomous agent and a collaborative information source, leading to a scalable network of interacting barrier functions.

Another promising direction involves incorporating dynamic vehicle models directly into the safety filter. While the present formulation operates at the kinematic level, extending the framework to acceleration-controlled or fully dynamic models would further improve physical realism and enable tighter safety guarantees under high-speed maneuvers.

Additionally, advanced perception and learning-based components may be integrated to adapt barrier parameters online. For instance, adaptive tuning of class-$\mathcal{K}$ functions or

learning-based estimation of traffic patterns could allow the system to balance safety margins and efficiency in a context-aware manner.

Finally, experimental validation on real autonomous vehicles operating in conjunction with physical smart infrastructure represents a crucial step toward deployment. Real-world testing would provide insights into communication delays, sensing uncertainty, and robustness considerations that cannot be fully captured in simulation.

Chapter 5

Dynamic Control with HOCBFs and Viability

Section 3 demonstrated that a constraint-based architecture built upon Control Barrier Functions effectively handles urban driving scenarios such as lane keeping, overtaking, and intersection management when applied to a kinematic vehicle model. The modularity of the CBF framework enables the systematic encoding of traffic regulations and road constraints into a unified optimization problem, providing a structured and interpretable safety layer.

However, the kinematic formulation relies on the assumption that the desired velocity vector can be instantaneously tracked. This assumption is only valid at low speeds and under sufficiently high tire-road friction conditions. In realistic driving scenarios, vehicle inertia, actuator dynamics, and physical limitations such as torque saturation and steering rate constraints introduce a non-negligible mismatch between the planned kinematic motion and its physical execution.

From a control-theoretic perspective, safety constraints defined in the position space exhibit a relative degree greater than one with respect to the actual control inputs (longitudinal force and steering angle). Consequently, standard CBF formulations, which assume relative degree one, become insufficient when directly applied to the dynamic vehicle model.

This limitation is not exclusive to CBF-based methods. Planning-based approaches typically rely on spatial discretizations of the environment such as occupancy grids, cell decompositions, or state-lattice representations to generate collision-free trajectories [11, 36]. While these methods provide structured environment representations, they often suffer from scalability issues and high computational burden in complex urban scenarios characterized by dynamic obstacles and frequent replanning requirements. Moreover, safety is generally enforced through geometric constraints and heuristic safety margins, without explicitly incorporating vehicle dynamics and actuator limitations [37].

Sampling-based and trajectory optimization methods improve flexibility by directly exploring the space of feasible trajectories and allowing the inclusion of kinematic and

dynamic constraints [12]. Nevertheless, safety guarantees are typically achieved through conservative buffer zones or offline verification procedures. Providing formal, real-time safety guarantees remains challenging, especially in the presence of uncertainty and rapidly evolving traffic conditions.

Furthermore, even when higher-order formulations are considered, hard actuator constraints may render the safety problem infeasible: a control input that satisfies the barrier condition may simply not exist within the physical limits of the vehicle. This issue highlights the necessity of jointly accounting for system dynamics, relative degree, and actuator saturation within the safety framework.

To address these challenges, this chapter extends the previous architecture to a dynamic bicycle model, enabling the representation of coupled longitudinal-lateral dynamics and actuator limitations. High-Order Control Barrier Functions are introduced to handle the increased relative degree of position-based safety constraints. In addition, a viability-based algorithm is integrated to explicitly and recursively compute state-dependent admissible input sets, ensuring long-term recursive feasibility of the safety constraints even under actuator saturation.

This unified approach bridges the gap between formal safety guarantees and physically realizable control actions, enabling the extension of the framework beyond low-speed urban scenarios to more general driving environments.

5.1 Vehicle Dynamic Models

In the initial development phase, the autonomous vehicle was modeled using a kinematic bicycle model, which is widely adopted for low-speed urban driving scenarios due to its simplicity and computational efficiency. In structured urban environments, where vehicle velocities are relatively low and tire slip effects remain limited, kinematic formulations provide sufficiently accurate motion predictions while enabling efficient trajectory planning and real-time control. However, in order to extend the proposed framework beyond purely urban contexts and generalize it to more diverse driving environments, a more comprehensive vehicle representation becomes necessary. While the kinematic model is adequate for low-speed operation, it neglects important dynamic effects that become significant at moderate and high velocities. For this reason, the autonomous road vehicle is modeled using a dynamic bicycle model, which represents a widely adopted compromise between physical accuracy and computational tractability [38, 39]. Compared to purely kinematic formulations, the dynamic bicycle model captures the coupling between longitudinal and lateral dynamics, explicitly accounting for tire slip angles, lateral tire forces, and yaw rate dynamics. in particular, the model describes the interaction between steering inputs, vehicle inertia, and tire-road friction forces, allowing accurate representation of acceleration and braking transients, lateral load transfer effects (in simplified form), and stability-related phenomena such as understeer and oversteer behavior [40]. These effects become increasingly relevant when the vehicle operates outside low-speed regimes, where the assumption of negligible tire slip no longer holds.

We consider a four-wheeled vehicle whose motion is described by a planar rigid-body model (see Fig.5.1). The front and rear axles are merged into two equivalent wheels, resulting in the classical bicycle abstraction. The vehicle state is defined as

$$x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T \in \mathbb{R}^6, \tag{5.1}$$

where:

- $x_1 = v_x$: longitudinal velocity expressed in the vehicle body-fixed frame;
- $x_2 = v_y$: lateral velocity expressed in the vehicle body-fixed frame;
- $x_3 = \omega$: yaw rate;
- $x_4 = X$: vehicle position along the global inertial X-axis;
- $x_5 = Y$: vehicle position along the global inertial Y-axis;
- $x_6 = \theta$: yaw angle (vehicle heading) with respect to the inertial frame.

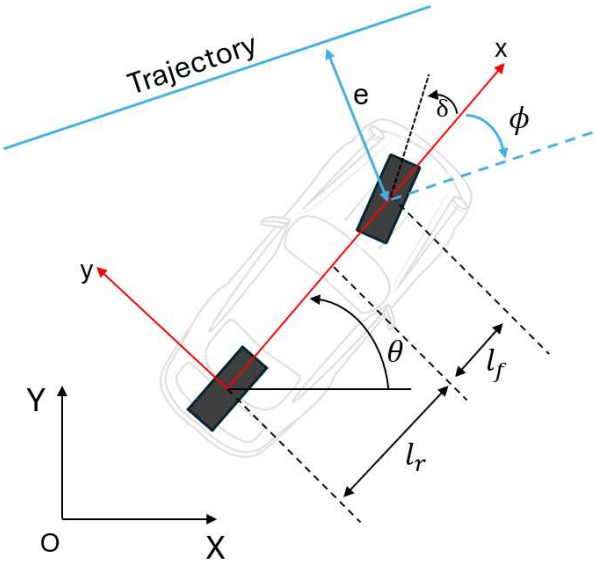


Figure 5.1: Dynamic Model and Stanley controller

The vehicle is controlled through the traction force at the front wheel and the steering angle. Therefore, the control input is defined as

$$u = \begin{bmatrix} F_{xf} & \delta \end{bmatrix}^T, \tag{5.2}$$

where F_{xf} is the longitudinal traction/braking force at the front axle and δ is the steering angle.

The dynamic bicycle model is then expressed as

$$\begin{cases} \dot{v}_x = \omega v_y + \frac{1}{m}\left(F_{xf}\cos\delta - F_{yf}\sin\delta\right) \\ \dot{v}_y = -\omega v_x + \frac{1}{m}\left(F_{xf}\sin\delta - F_{yf}\cos\delta + F_{yr}\right) \\ \dot{\omega} = \frac{1}{I_{zz}}\left(l_f\left(F_{xf}\sin\delta + F_{yf}\cos\delta\right) - l_r F_{yr}\right) \\ \dot{X} = v_x\cos\theta - v_y\sin\theta \\ \dot{Y} = v_x\sin\theta + v_y\cos\theta \\ \dot{\theta} = \omega \end{cases} \tag{5.3}$$

where m is the vehicle mass, I_{zz} is the yaw moment of inertia, while l_f and l_r denote the distances from the center of mass to the front and rear axles, respectively.

This model highlights the strong coupling between longitudinal and lateral dynamics. In particular, the yaw rate ω affects both $\dot{v}_x$ and $\dot{v}_y$, and steering influences not only the lateral motion but also the longitudinal acceleration due to the trigonometric coupling terms.

In order to close the dynamic model, the lateral tire forces at the front and rear axle, denoted as F_{yf} and F_{yr}, must be specified. In this work, we adopt a linear tire model (linearized Pacejka approximation), which is commonly used in control-oriented formulations and is sufficiently accurate for moderate slip angles, as typically encountered in urban driving scenarios[40].

The longitudinal traction force at the front axle is related to the motor torque τ by

$$F_{xf} = \frac{\tau}{2R_w}, \tag{5.4}$$

where R_w is the wheel radius and the factor 2 accounts for the fact that the front axle is composed of two wheels.

The lateral tire forces are modeled as proportional to the corresponding slip angles:

$$F_{yf} = K_f \alpha_f, \qquad F_{yr} = K_r \alpha_r, \tag{5.5}$$

where K_f and K_r are the front and rear cornering stiffness coefficients.

The slip angles are defined as

$$\begin{aligned} \alpha_f &= -\arctan\left(\tfrac{\omega l_f + v_y}{v_x}\right) + \delta, \quad \approx -\arctan\left(\tfrac{\omega l_f + v_y}{v_x}\right) + sin(\delta) \\ \alpha_r &= \arctan\left(\tfrac{\omega l_r + v_y}{v_x}\right). \end{aligned} \tag{5.6}$$

The slip angle α_f captures the difference between the front wheel orientation and the direction of motion of the front axle, while α_r describes the analogous quantity for the rear axle. The dependence on $\frac{1}{v_x}$ highlights a well-known limitation of bicycle models: they become singular at very low speeds. In practice, this is handled by enforcing a minimum velocity threshold or using hybrid kinematic/dynamic formulations. In typical urban road vehicles, the steering angle is physically bounded, usually within $\pm 40°$ [38, 40]. Therefore, it is common to assume the approximation

$$\sin(\delta) \approx \delta \tag{5.7}$$

or alternatively $\delta \approx \sin(\delta)$ when the steering appears explicitly in the slip angle formulation.

The proposed control architecture is not designed to directly generate a complete motion plan. Instead, it modifies a nominal driving behavior computed by a baseline controller, by minimally deviating from it while ensuring safety and feasibility.

In particular, the nominal steering angle is computed using the Stanley controller, a widely used path-tracking algorithm in autonomous driving due to its simplicity and robustness.

Let $\phi(t)$ denote the heading error, defined as the angular difference between the vehicle heading θ and the tangent direction of the reference path at the closest point. Let $e(t)$ denote the cross-track error, i.e., the lateral distance between the vehicle position and the reference path Fig.5.1. Then the desired steering angle is given by

$$\delta_{\text{des}}(t) = \phi(t) + \arctan\left(\frac{k\, e(t)}{k_s + v_x(t)}\right), \tag{5.8}$$

where $k > 0$ is a gain tuning the aggressiveness of lateral correction and $k_s > 0$ is a positive constant introduced to avoid singularities at low speeds.

It is worth noting that the Stanley controller implicitly introduces a speed-dependent behavior: at higher velocity the cross-track correction is less aggressive, improving stability and reducing oscillations.

The nominal longitudinal behavior is generated through a simple cruise control law, which aims at reaching and maintaining a desired speed limit v_{lim} imposed by road regulations. The desired traction force is computed as

$$F_{xf,\text{des}} = g\,(v_{\text{lim}} - v_x)\,, \tag{5.9}$$

where $g > 0$ is a proportional gain.

Although this controller does not explicitly account for vehicle-to-vehicle interaction, it provides a suitable baseline reference input that can be corrected by the safety layer based on Control Barrier Functions.

In practical implementations, the force $F_{xf,\text{des}}$ would also be saturated according to the vehicle drivetrain limits and braking constraints. In this work, such limitations are instead handled through viability-based dynamic bounds introduced in later sections.

5.2 Two-Stage QP

Starting from nominal commands generated by the lateral and longitudinal baseline controllers (Stanley and cruise control, respectively), the proposed framework computes a corrected control input that minimally deviates from the nominal one while enforcing safety and traffic-rule constraints encoded through High-Order Control Barrier Functions. In particular, let

$$u_{\text{des}}(t) = \begin{bmatrix} F_{xf,\text{des}}(t) & \delta_{\text{des}}(t) \end{bmatrix}^T \tag{5.10}$$

denote the nominal input, where F_{xf} is the front longitudinal tire force and δ is the steering angle.

The ideal objective of the safety filter is to compute the closest admissible input satisfying all safety constraints and actuator limitations, namely

$$
\begin{aligned}
u_i^* = \arg\min_{u} \quad & \|u - u_{des}\|^2 \\
\text{subject to} \quad & A_i(x_i)u_i + b_i(x_i) \geq 0
\end{aligned}
\tag{5.11}
$$

Within the standard HOCBF-QP paradigm, when the system dynamics are control-affine, HOCBF constraints can be expressed as linear inequalities in the decision variable ($A_i(x_i)u_i + b_i(x_i) \geq 0$), yielding a convex quadratic program that can be solved efficiently in real time.

However, the dynamic bicycle model adopted in this work is not control-affine with respect to the original input pair (F_{xf}, δ), since the steering angle enters the dynamics through trigonometric terms such as $\sin(\delta)$ and $\cos(\delta)$ (5.3). As a consequence, the HOCBF conditions cannot be directly cast as linear constraints, and the optimization problem in (5.11) becomes a nonlinear program. Such a formulation is generally too computationally demanding and numerically fragile for high-frequency onboard execution.

To overcome this limitation while preserving the computational advantages of QP-based safety filters, we reformulate the problem as a cascade of two convex optimization programs. Furthermore, solving simultaneously for both inputs would require explicitly addressing the strong nonlinear coupling between steering and vehicle dynamics. By decoupling the computation into two sequential stages, the approach yields smaller optimization problems with a more favorable structure, which can be solved sequentially at each control cycle with improved numerical robustness and reduced computational burden.

5.2.1 Stage 1: steering optimization via an affine reparametrization

Let us introduce a reparametrization of the steering angle:

$$
z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos\delta \\ \sin\delta \end{bmatrix}, \qquad z_{\text{des}} = \begin{bmatrix} \cos\delta_{\text{des}} \\ \sin\delta_{\text{des}} \end{bmatrix}. \tag{5.12}
$$

With this change of variables, the vehicle dynamics can be rewritten so that they become affine in z when F_{xf} is treated as a known quantity (e.g., the optimal traction computed at the previous time step, or a warm-start value): Using this parametrization, the dynamic bicycle model can be rewritten as

$$
\begin{cases}
\dot{v}_x = \omega v_y + \frac{1}{m}\left(F_{xf} z_1 - F_{yf} z_2\right) \\
\dot{v}_y = -\omega v_x + \frac{1}{m}\left(F_{xf} z_2 - F_{yf} z_1 + F_{yr}\right) \\
\dot{\omega} = \frac{1}{I_{zz}}\left(l_f\left(F_{xf} z_2 + F_{yf} z_1\right) - l_r F_{yr}\right) \\
\dot{X} = v_x \cos\theta - v_y \sin\theta \\
\dot{Y} = v_x \sin\theta + v_y \cos\theta \\
\dot{\theta} = \omega
\end{cases}
\tag{5.13}
$$

This step eliminates the trigonometric non-affinity in the steering input.

The steering-stage optimization is then formulated as

$$\begin{aligned} & z^* = argmin\|z - z_{des}\|^2 \\ & A_i(x)z + b_i(x) \geq 0 \qquad i = 1, \ldots, n \\ & z_1 \leq \overline{z_1}_j \qquad\qquad\qquad\qquad j \in \{1, \ldots, m\} \\ & z_2 \leq \overline{z_2}_j \qquad\qquad\qquad\qquad j \in \{1, \ldots, m\} \\ & z^T z = 1 \end{aligned} \tag{5.14}$$

Where $A_i(x)z + b_i(x) \geq 0$ represent the HOCBFs constraints in the affine form (they will be described later) $\overline{z_1}_j$ $\overline{z_2}_j$ represent the actuators bounds and the last constraint enforces consistency with the trigonometric parametrization. Importantly, (5.14) has linear inequality constraints and a single quadratic equality constraint, it can be solved efficiently, and it remains substantially lighter than a general nonlinear program.

Once z^* is found, the steering command is recovered as

$$\delta^* = \text{atan2}(z_2^*, z_1^*). \tag{5.15}$$

5.2.2 Stage 2: traction optimization with fixed steering

After computing δ^* (equivalently z^*), we fix the steering input and optimize the traction force. With $z = z^*$ treated as known, the dynamics become affine in F_{xf}: is affine with respect to the input F_{xf}. Hence, the optimization problem for computing F_{xf}^* is

$$\begin{aligned} & F_{xf}^* = argmin\|F_{xf} - F_{xf_{des}}\|^2 \\ & A_i(x)F_{xf} + b_i(x) \geq 0 \qquad i = 1, \ldots, n \\ & \underline{F}_{xf} \leq F_{xf} \leq \overline{F}_{xf} \end{aligned} \tag{5.16}$$

where $[\underline{F}_{xf}, \overline{F}_{xf}]$ includes both static actuator bounds and the dynamic feasibility bounds derived later from viability arguments.

The adopted two-stage optimization structure is intentionally designed to respect both the physical actuation characteristics of the vehicle and the driving intuition observed in urban scenarios. In particular, the order of the stages is not arbitrary: steering is solved first because most safety constraints of geometric nature such as lane boundary enforcement and obstacle avoidance are predominantly satisfied through lateral deviation and heading regulation, which are directly achieved via steering inputs. Resolving such conflicts through steering is typically more energy-efficient and results in smoother velocity profiles compared to aggressive braking maneuvers, thereby improving overall comfort and motion consistency. From a computational perspective, fixing the longitudinal force (e.g., by using the previously computed value) allows the resulting model to recover an affine structure in the decision variable of the steering stage, enabling HOCBF constraints to be expressed as linear inequalities and preserving the convex QP formulation.

Conversely, intersection handling and STOP/priority behaviors are naturally enforced through longitudinal constraints that require a sufficient reduction of vehicle speed before entering designated regions. These requirements are therefore more appropriately

addressed through the traction/braking input F_{xf}, while steering should remain unconstrained as much as possible in order to maintain lane alignment and avoid unnecessary lateral oscillations near intersections. This separation also improves interpretability: steering primarily ensures lateral safety and lane keeping, whereas longitudinal control governs speed adaptation and stopping decisions.

Although the two-stage decomposition introduces an approximation with respect to the ideal joint optimization, since the coupling between steering and traction is treated sequentially rather than simultaneously, it provides a favorable compromise between theoretical rigor and real-time applicability. Both stages remain quadratic programs. Moreover, computational efficiency is further improved by dynamically activating only the HOCBF constraints that are relevant at each time instant (e.g., those associated with nearby obstacles, lane boundaries, or priority zones). Feasibility is enhanced by the fact that the baseline controllers provide a natural initial guess, and the safety filter applies only the minimal correction required to satisfy constraints. Finally, actuator bounds and viability-based limits (introduced later) prevent solutions that would be safe in the abstract model but physically unrealizable due to saturation effects.

5.3 Viability Algorithm

Standard HOCBF-based QP formulations often implicitly rely on the assumption that the admissible control set U is sufficiently large to guarantee feasibility of the barrier inequalities. In practice, however, real vehicles are subject to strict actuation limits, yielding bounded inputs of the form

$$F_{xf} \in [F_{xf,\min}, F_{xf,\max}], \qquad \delta \in [\delta_{\min}, \delta_{\max}], \tag{5.17}$$

which may be insufficient to ensure forward invariance of the safe set, particularly in the presence of disturbances, modeling uncertainties, or tight geometric constraints.

To address this limitation, a viability-based module is integrated into the proposed control architecture. The goal of this component is to compute, in real time, dynamic admissible bounds on the longitudinal acceleration (and, equivalently, on the traction/braking force), such that the vehicle state is constrained to remain within a subset of the state space from which future satisfaction of safety and traffic constraints remains achievable.

This issue becomes evident, for instance, in braking scenarios near an obstacle. In such situations, the barrier condition may impose a minimum required deceleration to prevent violation of the safety constraint. We denote by $a_{\text{req}} > 0$ the magnitude of the longitudinal deceleration that would be necessary, at the current state, to satisfy the HOCBF inequality and preserve forward invariance of the safe set. This quantity is state-dependent and generally increases as the vehicle approaches the obstacle or travels at higher speeds.

However, the vehicle is subject to physical actuation limits. Let $a_{\max} > 0$ denote the maximum achievable braking deceleration, which is determined by actuator capabilities,

tire road friction conditions, and vehicle dynamic constraints. Consequently, the admissible longitudinal acceleration is bounded by

$$\dot{v}_x \geq -a_{\max}. \tag{5.18}$$

Whenever $a_{\text{req}} > a_{\max}$, the deceleration required by the barrier condition exceeds the physically available braking authority. In this case, the corresponding HOCBF constraint becomes infeasible and the quadratic program can no longer guarantee safety.

Therefore, instead of applying a posteriori saturation of the control input which may violate the barrier condition and compromise formal guarantees the proposed approach explicitly constrains the optimization to viable controls, i.e., control actions that ensure the system remains within a region where future constraint satisfaction is still possible.

5.3.1 Real-time viability bounds for longitudinal acceleration

Instead of explicitly computing the full viability kernel, which would be computationally intractable in real time, we adopt a discrete-time approximation inspired by the method proposed in [41], adapted here to the longitudinal motion of an autonomous vehicle.

The objective is to compute, at each control cycle, the admissible and viable interval of longitudinal acceleration

$$a_x \in [a_{x,m}^V, a_{x,M}^V], \tag{5.19}$$

such that applying any acceleration within these bounds guarantees that the next state remains viable and that future feasible inputs continue to exist.

Discrete-time forward propagation Let Δt denote the control sampling time. Assuming constant acceleration over $[0, \Delta t]$, the longitudinal dynamics are propagated as

$$v_x(\Delta t) = v_x + a_x \Delta t, \qquad X(\Delta t) = X + v_x \Delta t + \tfrac{1}{2} a_x \Delta t^2.$$

The admissible acceleration must satisfy four requirements:

- **Velocity feasibility:** the next velocity must remain within physical speed limits $v_x \in [v_{x,m}, v_{x,M}]$.
- **Position feasibility:** the next position must not violate spatial constraints (e.g., obstacle clearance, stop lines, road bounds).
- **Trajectory feasibility:** the entire trajectory segment $(X(t), v_x(t))$ for $t \in [0, \Delta t]$ must remain inside the feasible set $\mathcal{F}$.
- **Future braking feasibility:** from the predicted state at $t = \Delta t$, it must still be possible to apply admissible braking to avoid violating downstream constraints.

These conditions define a discrete approximation of the viability kernel. Formally, we require:

$$
\begin{aligned}
&X(0) = X, \qquad v_x(0) = v_x, \\
&(X(t), v_x(t)) \in \mathcal{F} \qquad \forall t \in [0, \Delta t], \\
&v_{x,m}^V(\Delta t) \leq v_x(\Delta t) \leq v_{x,M}^V(\Delta t), \\
&a_{x,m}^V \leq a_x \leq a_{x,M}^V.
\end{aligned}
\tag{5.20}
$$

Here:

- $a_{x,m}^V$ and $a_{x,M}^V$ are the minimum and maximum admissible longitudinal accelerations ensuring viability;
- $v_{x,m}^V(\Delta t)$ and $v_{x,M}^V(\Delta t)$ denote the velocity bounds at the next time step that preserve future feasibility;
- $\mathcal{F}$ represents the feasible set defined by physical limits (speed, acceleration) and environmental constraints (obstacles, stop regions, intersections).

The resulting bounds $a_{x,m}^V$ and $a_{x,M}^V$ are then mapped into traction force limits through

$$
F_{xf} \in [m a_{x,m}^V,\ m a_{x,M}^V], \tag{5.21}
$$

and directly incorporated as linear constraints in the traction-stage quadratic program described in Section 5.2.

The condition (5.20) are integrated in the viability algorithm. The algorithm is executed at each control cycle and computes the admissible acceleration interval by evaluating all active constraints. In particular, each constraint produces a candidate upper and lower bound on the acceleration, and the final admissible interval is obtained by selecting the most restrictive bounds:

$$
a_{x,m}^V = \max_k a_{x,m}^{(k)}, \qquad a_{x,M}^V = \min_k a_{x,M}^{(k)}, \tag{5.22}
$$

where k indexes all active constraints (velocity limit, position limit, stopping distance, obstacle proximity, etc.).

The resulting interval is then guaranteed to satisfy all constraints simultaneously. As soon as,

$$
a_{x,m}^V = a_{x,M}^V, \tag{5.23}
$$

the current state is on the boundary the viability region, meaning that only the maximum control input can prevent violation of the constraints. In practice, this condition can be interpreted as an emergency situation and should trigger a fallback behavior (e.g., maximum braking).

For clarity and reproducibility, the complete pseudocode of the viability procedure is reported in Appendix 8.2.

A key advantage of the proposed approach is that the viability computation does not require the explicit construction of the full viability kernel $\mathcal{V}$. The exact computation of $\mathcal{V}$ would involve solving a backward reachability problem over the state space, which is computationally prohibitive for real-time autonomous driving applications.

Instead, we adopt a discrete-time local approximation: at each control cycle, we directly compute the admissible interval of longitudinal acceleration that guarantees that the state reached after a short horizon Δt remains feasible and preserves future braking capability.

This one-step recursive feasibility condition avoids state-space discretization or value-function computation, reducing the viability check to simple algebraic bounds on the propagated dynamics. As a result, the acceleration limits can be updated online with negligible computational overhead.

This makes the approach suitable for real-time implementation and scalable to complex urban scenarios, where the set of active constraints changes dynamically. In addition, the resulting viability bounds are directly compatible with QP-based optimization, since they appear as simple linear inequalities in the traction force.

In conclusion, the viability module represents an essential component of the proposed control architecture, enabling the integration of realistic vehicle limitations into the HOCBF-based safety filter and ensuring that the resulting control inputs remain physically implementable.

The output of the viability computation can be visualized as a viability map in the (v_x, X) space, which identifies the admissible acceleration region depending on the current state.

An example of such a map is shown in Fig. 5.2

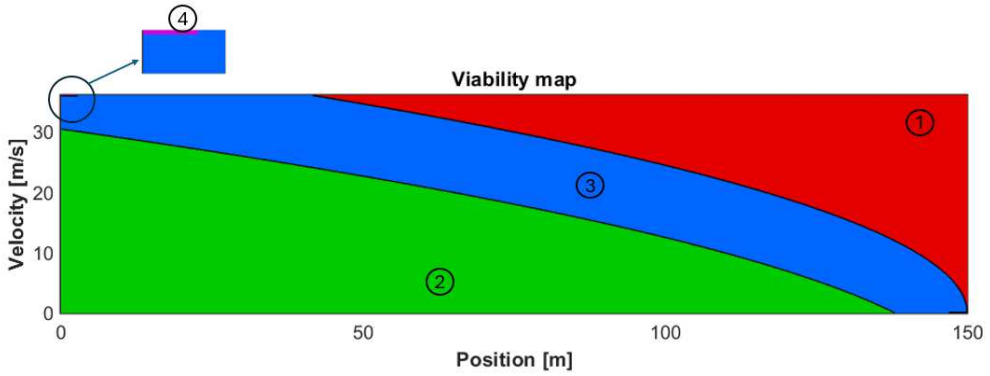


Figure 5.2: Example of a viability map along the longitudinal direction of a vehicle, considering acceleration, velocity, and viability constraints. The map, computed with a maximum absolute acceleration of $a_{\max} = 6\,\mathrm{m/s^2}$, illustrates how different regions are dominated by distinct active constraints: The red area (1) represents non-viable states where no feasible acceleration exists. The green area (2) corresponds to acceleration-limited states, where the maximum absolute acceleration is reached. The blue area (3) represents viability-limited states, where future viability constraints (e.g., position bounds) restrict the feasible acceleration range. Finally, the violet area (4) shows velocity-limited states, where further acceleration would exceed the velocity bound.

5.4 Use of HOCBFs

This subsection formalizes how High Order Control Barrier Functions are employed to encode traffic rules and safety requirements within a unified constraint-based architecture.

Urban autonomous driving involves a heterogeneous set of constraints:

- **Always-active structural constraints**, such as staying within road boundaries;
- **Situation-dependent constraints**, such as obstacle avoidance (only when obstacles are detected);
- **Event-triggered constraints**, such as stop/right-of-way enforcement near intersections.

The architecture therefore maintains a dynamic set of active HOCBF constraints. At each control cycle, perception modules provide the environment state (road geometry, obstacle positions and velocities, intersection status), and only the relevant barrier inequalities are activated. This reduces computational load and avoids unnecessary conservatism.

In the following, we detail the explicit barrier functions used for obstacle avoidance, road boundaries / lane keeping, and STOP and right-of-way behaviors.

5.4.1 Obstacle avoidance constraint

To safely navigate among road users, the vehicle must avoid collisions with both static and moving obstacles. Obstacles are approximated by ellipses, which provide a flexible and computationally convenient representation, particularly suitable for elongated objects such as cars and bicycles. Pedestrians can be captured as circles (special case of ellipses with equal semi-axes).

Let $x_{ve} = (X, Y) \in \mathbb{R}^2$ denote the vehicle position in the global frame, and let $x_o(t) = (X_o(t), Y_o(t))$ denote the obstacle center. The collision-free region around the obstacle is described by the barrier function

$$h_o(x,t) = \frac{(x_{ve,1} - x_{o,1}(t))^2}{a^2} + \frac{(x_{ve,2} - x_{o,2}(t))^2}{b^2} - 1, \tag{5.24}$$

where $a, b > 0$ are the ellipse semi-axes (chosen according to the obstacle footprint plus a safety margin). The safety requirement is

$$h_o(x,t) \geq 0, \tag{5.25}$$

meaning that the vehicle stays outside the obstacle ellipse.

Since h_o depends on position, under the vehicle dynamics it typically has relative degree 2, hence we enforce the HOCBF condition with $h = h_o$. In compact form, we obtain:

$$L_f^2 h_o(x,t) + L_g L_f h_o(x,t)\, u + \frac{\partial^2 h_o(x,t)}{\partial t^2} + L_f h_o(x,t) + \frac{\partial h_o(x,t)}{\partial t} + \lambda_{2,o}\, \psi_{1,o}(x,t) \geq 0, \tag{5.26}$$

where $\psi_{1,o} = \dot{h}_o + \lambda_{1,o} h_o$ and $\lambda_{1,o}, \lambda_{2,o} > 0$.

Driving-style interpretation. The gains $\lambda_{1,o}$ and $\lambda_{2,o}$ affect how aggressively the vehicle reacts to an obstacle. Larger values typically yield more permissive behavior (the system can remain closer to the boundary while still satisfying the inequality), while smaller values enforce earlier and smoother avoidance maneuvers. In a thesis

context, these parameters can be interpreted as behavioral knobs that trade off clearance, reactivity, and comfort.

5.4.2 Road boundaries and lane keeping

To ensure compliance with road geometry, the vehicle must remain within the lane boundaries. Let $x_{L_i} \in \mathbb{R}^2$ denote the closest point on the i-th lane boundary line to the vehicle position, where $i \in \{l, r\}$ denotes left/right boundary. We define the barrier function

$$h_{L_i}(x,t) = \|x_{ve} - x_{L_i}\|^2 - d_{L_i}^2, \qquad i \in \{l, r\}, \tag{5.27}$$

where $d_{L_i} > 0$ is the desired minimum distance from the boundary. The safety condition $h_{L_i} \geq 0$ enforces that the vehicle remains at least d_{L_i} away from each boundary.

As in the obstacle case, h_{L_i} is position-based and thus relative degree 2; we enforce the corresponding HOCBF inequality

$$L_f^2 h_{L_i}(x,t) + L_g L_f h_{L_i}(x,t)\, u + \frac{\partial^2 h_{L_i}(x,t)}{\partial t^2} + L_f h_{L_i}(x,t) + \frac{\partial h_{L_i}(x,t)}{\partial t} + \lambda_{2,L}\, \psi_{1,L_i}(x,t) \geq 0, \tag{5.28}$$

where $\psi_{1,L_i} = \dot{h}_{L_i} + \lambda_{1,L} h_{L_i}$. In contrast with obstacle avoidance, lane keeping is treated as an always-on structural constraint; therefore, $\lambda_{1,L}$ and $\lambda_{2,L}$ are typically fixed to ensure consistent behavior independently of the selected driving style.

5.4.3 STOP behavior and right-of-way enforcement

Urban driving requires respecting traffic rules at intersections: stop signs, traffic lights, and priority rules. In the proposed framework, these behaviors are encoded through an event-triggered HOCBF that constrains the vehicle to reduce its speed and come to a full stop before entering a priority/attention zone.

When an intersection (or priority region) is detected, we build an elliptical attention region around it, centered at $q_S = (X_S, Y_S) \in \mathbb{R}^2$, with semi-axes $e, f > 0$. Let $\pi(t) \in \{0, 1\}$ be an external logical variable provided by perception/infrastructure sensing, where:

- $\pi(t) = 1$ indicates that the vehicle must stop (e.g., another vehicle has right-of-way, red light, stop sign);
- $\pi(t) = 0$ indicates that the vehicle may proceed (e.g., free intersection / green light).

The corresponding barrier function is defined as

$$h_S(x,t) = \begin{cases} \frac{(x_{ve,1} - x_{S,1})^2}{e^2} + \frac{(x_{ve,2} - x_{S,2})^2}{f^2} - 1, & \text{if } \mathrm{dist}_c < c\,\mathrm{dist}_{sf}(v_{ve}) \text{ and } \pi(t) = 1, \\ 0, & \text{if } \mathrm{dist}_c \geq c\,\mathrm{dist}_{sf}(v_{ve}) \text{ or } \pi(t) = 0, \end{cases} \tag{5.29}$$

where dist_c is the current distance to the stop/priority zone, $\mathrm{dist}_{sf}(v_{ve})$ is a speed-dependent safety distance, and $c \geq 1$ is a confidence factor modulating conservatism.

This construction yields an intuitive behavior:

- if a stop is required ($\pi = 1$) and the vehicle is close enough to the zone, then h_S becomes active and forces deceleration;
- otherwise the constraint is inactive (set to 0) and does not affect the optimization.

The stop constraint is enforced using the HOCBF condition (relative degree 2), resulting in

$$L_f^2 h_S(x,t) + L_g L_f h_S(x,t)\, u + \frac{\partial^2 h_S(x,t)}{\partial t^2} + L_f h_S(x,t) + \frac{\partial h_S(x,t)}{\partial t} + \lambda_{2,S}\, \psi_{1,S}(x,t) \geq 0, \tag{5.30}$$

with $\psi_{1,S} = \dot{h}_S + \lambda_{1,S} h_S$.

Confidence factor and driving style. In this case, $\lambda_{1,S}, \lambda_{2,S}$ primarily influence the stopping "sharpness" and are not used to encode driving style. Instead, the parameter c serves as the main knob: lower values (down to $c = 1$) correspond to a more aggressive behavior, where the vehicle commits to the stop constraint only when it is closer to the intersection; higher values yield earlier braking and more conservative gap acceptance.

5.5 Integration into the overall architecture

The barrier constraints (5.26), (5.28), and (5.30) form the core rule-based safety layer. They are integrated into the two-stage optimization described in Section 5.2 as follows:

- **Steering QP:** obstacle avoidance and lane keeping constraints are enforced to compute δ^* (via the z-parameterization).
- **Traction QP:** obstacle avoidance, lane keeping, and STOP/right-of-way constraints are enforced to compute F_{xf}^*, together with viability-based force bounds (Section 5.3).

5.5.1 Stage 1: Lateral Safety (Steering Optimization)

The first stage focuses on geometric constraints that are primarily resolved through lateral motion: obstacle avoidance and lane keeping. Since the stopping behavior is inherently longitudinal, the STOP/right-of-way constraint h_S is excluded from this stage to prevent unnecessary lateral deviations near intersections.

Exploiting the variable change $z = [\cos\delta, \sin\delta]^T$ introduced in (5.12), the vehicle dynamics become affine in z. The optimization problem is formulated as a QCQP (Quadratically Constrained Quadratic Program) to compute the optimal steering orientation z^*:

$$\begin{aligned} z^* = \arg\min_{z \in \mathbb{R}^2} \quad & \|z - z_{des}\|^2 & \\ \text{s.t.} \quad & A_{obs}(x) z \leq b_{obs}(x) & \text{(Obstacle Avoidance Eq. (5.26))} \\ & A_{lane}(x) z \leq b_{lane}(x) & \text{(Lane Keeping Eq. (5.28))} \\ & z^T z = 1 & \text{(Consistency Constraint)} \end{aligned} \tag{5.31}$$

5.5.2 Stage 2: Longitudinal Safety (Traction Optimization)

The second stage computes the optimal traction/braking force F_{xf}^*. Unlike the steering stage, this optimization incorporates all active safety constraints, including the event-triggered STOP/right-of-way constraint. Crucially, it integrates the viability-based dynamic bounds derived in Section 5.3 to ensure that the computed force is physically realizable given the actuator limits and the current state.

With the steering fixed at δ^*, the dynamics are affine in F_{xf}, resulting in the following convex QP:

$$
\begin{aligned}
F_{xf}^* = \arg\min_{F_{xf}\in\mathbb{R}} \quad & \|F_{xf} - F_{xf,des}\|^2 \\
\text{s.t.} \quad & A_{obs}(x)F_{xf} \le b_{obs}(x) && \text{(Obstacle Avoidance Eq. (5.26))} \\
& A_{lane}(x)F_{xf} \le b_{lane}(x) && \text{(Lane Keeping Eq. (5.28))} \\
& A_{stop}(x)F_{xf} \ge b_{stop}(x) && \text{(STOP Eq. (5.30))} \\
& ma_{x,m}^{\mathcal{V}} \le F_{xf} \le ma_{x,M}^{\mathcal{V}} && \text{(Viability Bounds)}
\end{aligned}
\tag{5.32}
$$

The viability constraints $[a_{x,m}^{\mathcal{V}}, a_{x,M}^{\mathcal{V}}]$ impose state-dependent limits on the longitudinal force, guaranteeing that the solution does not violate the actuator saturation limits or lead to an inevitable collision state in the future.

5.6 Simulation Results

This section presents a detailed validation of the proposed constraint-based control architecture. The objective of the simulation campaign is to evaluate the effectiveness of the HOCBF-based framework in enforcing road rules, avoiding collisions with static and dynamic obstacles, and ensuring physically feasible maneuvers through the integration of viability-based input bounds.

The simulations are structured in two stages. First, the proposed approach is compared against a state-of-the-art trajectory replanning strategy based on dynamic spline generation in Matlab. Second, a realistic urban driving scenario is reproduced in the Webots simulator to assess the closed-loop behavior under dynamic conditions,viability constraints, intersections, moving vehicles, and constrained road geometries.

5.6.1 Matlab simulation

All experiments were executed on a laptop equipped with an Intel i7-10510U processor and 8 GB of RAM. The control architecture was initially developed and tested in Matlab in order to validate its feasibility and verify the numerical stability of the two-stage optimization framework described in Section 5.2.

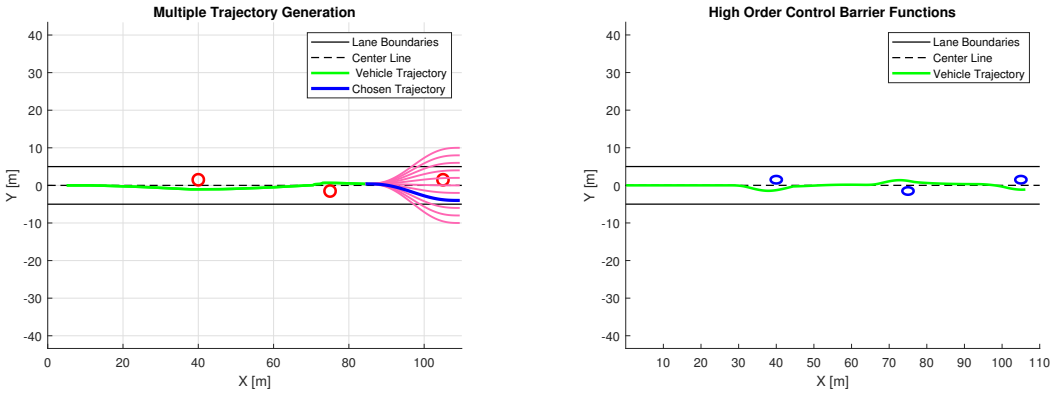
The optimization problems described in (5.31) and (5.32) were solved online at each

control cycle. In particular, the first stage corresponds to a QCQP due to the constraint $z^T z = 1$, while the second stage is a standard convex QP. Both problems were found to be solvable within a few milliseconds, making the approach suitable for real-time onboard implementation.

In order to evaluate the proposed method against an established baseline, we compare it with the real-time dynamic trajectory planning strategy proposed in [12]. This approach generates multiple candidate trajectories using cubic splines constructed from the road geometry. Each candidate spline is evaluated using a cost function that penalizes unsafe behavior and favors smooth motion, and the best trajectory is selected online.

This baseline represents a relevant benchmark since it reflects a widely adopted paradigm in autonomous driving: explicitly generating and selecting feasible trajectories rather than shaping the input through constraint enforcement.

Fig. 5.3 illustrates a representative scenario in which the road is partially obstructed by obstacles A key observation is that both approaches successfully compute collision-free behavior within approximately 10 ms. Moreover, the computation time remains approximately constant as the number of obstacles increases, which is essential in urban environments where the density of agents may vary significantly. Both methods require a simplified geometric representation of obstacles. In the spline-based planner, obstacles are modeled as circles, while in the proposed approach they are modeled as ellipses. Elliptical modeling provides a more accurate representation for road vehicles, which are typically longer than wide. This choice improves the realism of the safety margin and results in smoother and less conservative maneuvers in overtaking scenarios.



(a) **(b)**

Figure 5.3: Picture (a) illustrates the approach based on the generation of multiple cubic splines. A cost function is then used to evaluate the different trajectories by penalizing or favoring certain conditions, ultimately selecting the optimal spline. Figure (b), on the other hand, represents the HOCBF-based architecture, where the vehicle successfully avoids obstacles by solving two optimization problems, ensuring safety.

While both methods succeed in producing safe trajectories, significant differences emerge when analyzing the generated speed profiles. Fig. 5.4 reports the longitudinal speed comparison between the spline-based replanning approach and the proposed HOCBF-based method.

The spline-based planner exhibits sharper oscillations in the velocity profile, with frequent

accelerations and decelerations. This behavior is typical of sampling-based trajectory selection approaches, where small changes in the cost function or obstacle configuration may cause the selected spline to switch between candidates, resulting in non-smooth longitudinal behavior.

Conversely, the proposed approach yields a smoother velocity evolution. This is a direct consequence of the optimization objective, which minimizes the deviation from the nominal cruise control reference while only applying corrections when required by the constraints. Such smoothness is desirable in real-world driving, as it improves passenger comfort, reduces actuator stress, and contributes to energy efficiency.

In addition, the HOCBF-based approach tends to maintain a higher minimum velocity compared to the spline-based method, indicating that the vehicle avoids unnecessary braking maneuvers. This is particularly advantageous in urban driving, where excessive deceleration can reduce traffic flow and increase energy consumption.

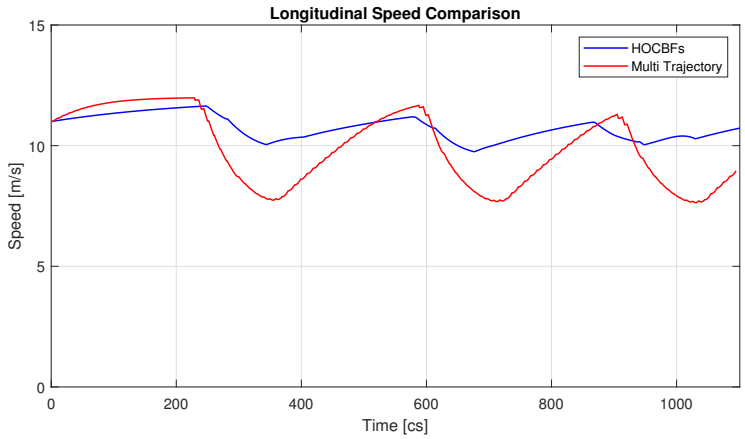


Figure 5.4: Differences in speed profiles between the two proposed approaches.

A critical limitation of the spline-based planner is that it lacks a structured safety fallback when the road is fully blocked. In such situations, the trajectory generation procedure attempts to construct a collision-free geometric path within the admissible road region. If no feasible spline satisfying the obstacle and road constraints can be found, the planner may fail to return a valid trajectory. Since the method is primarily geometric and does not explicitly encode dynamic braking feasibility, this may lead to undefined behavior or to late reactions that exceed the vehicle's deceleration capabilities.

In contrast, the proposed framework always retains the possibility of generating a safe stop maneuver. This property follows from two key elements: the HOCBF constraints, which continuously enforce forward invariance of the safe set, and the viability-based input bounds, which guarantee that the applied acceleration remains within a region from which future feasible braking actions exist.

Even when obstacle avoidance through steering becomes impossible (e.g., the road ahead is completely occupied), the optimization problem remains feasible because a pure braking action satisfying the viability bounds is always admissible. As a result, the controller can systematically reduce the longitudinal velocity while respecting actuation limits, ultimately bringing the vehicle to a safe stop without violating dynamic constraints.

5.6.2 Webots simulation

After validating the feasibility of the proposed architecture in Matlab, a high-fidelity simulation environment was implemented in Webots. In this environment, the vehicle successfully performed navigation tasks such as reaching a destination, maintaining lane position, overtaking slower vehicles, and stopping at intersections. At each time step, the two optimization problems were solved online, confirming that the proposed architecture is compatible with real-time execution.

A crucial contribution of this work is the integration of the viability algorithm into the HOCBF framework. To demonstrate its impact, we compare two configurations: HOCBF-based architecture without viability bounds and HOCBF-based architecture with viability bounds

Fig. 5.5 shows the resulting longitudinal acceleration profiles. In the case where viability is not enforced, the computed braking action violates the admissible acceleration bounds, requiring unrealistic deceleration. This leads to an inevitable collision, as shown in Fig. 5.6 Conversely, when viability bounds are included, the acceleration remains within the admissible interval $[a^{\mathcal{V}}_{x,m}, a^{\mathcal{V}}_{x,M}]$. As a result, the vehicle anticipates the braking maneuver and successfully avoids collision while remaining within physically realizable limits.

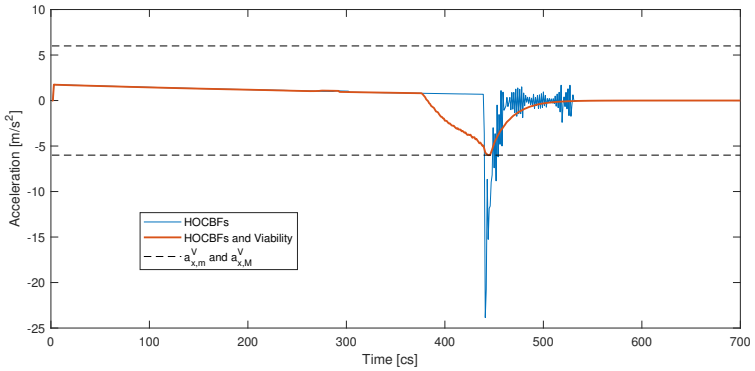
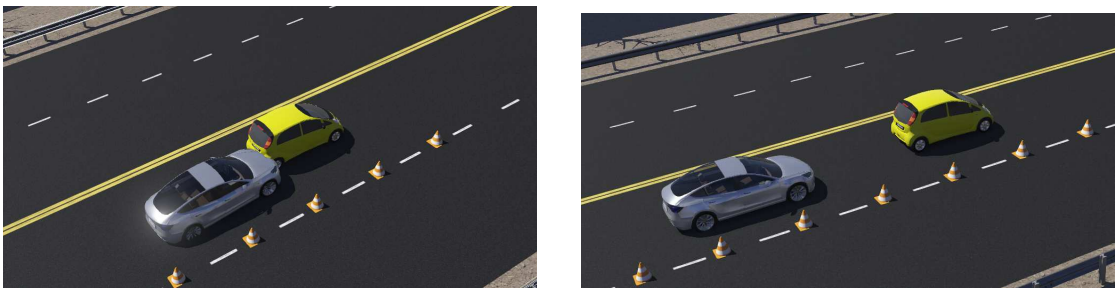


Figure 5.5: Acceleration Graphs between using only HOCBFs and combining HOCBFs with Viability



Without Viability

With Viability

Figure 5.6: Differences between using only HOCBFs and combining HOCBFs with Viability .

This experiment highlights a key conceptual point: although HOCBFs provide formal guarantees of forward invariance, such guarantees rely on the assumption that the required control input is feasible. In practical driving, actuator saturation and braking limits can invalidate this assumption. The viability algorithm restores feasibility by en-

suring that the control action remains within a region where future constraint satisfaction is still possible.

The simulation campaign confirms that the proposed architecture provides a robust and computationally efficient solution for urban autonomous driving. The HOCBF-based constraints successfully enforce collision avoidance and lane keeping, while the two-stage optimization ensures real-time feasibility. Most importantly, the integration of viability-based bounds is shown to be essential in preventing unrealistic braking commands and ensuring safety under physical actuator limitations.

Overall, these results demonstrate that the proposed approach is not only theoretically grounded but also practically viable for real-time implementation in realistic autonomous driving scenarios.

The simulation video is available online and provides additional qualitative evidence of the system performance https://doi.org/10.5281/zenodo.17566403 .

5.7 Conclusions and Future Work

This chapter presented a constraint-based control architecture for urban autonomous driving built upon High-Order Control Barrier Functions and a two-stage optimization strategy. The framework was validated through extensive simulations in both Matlab and Webots, confirming real-time feasibility and numerical stability. The Matlab simulations demonstrated that the two-stage optimization scheme, consisting of a steering QCQP followed by a traction QP, can be solved within a few milliseconds per control cycle. The comparison with a spline-based real-time trajectory planner highlighted several qualitative differences. While both methods successfully generated collision-free behaviors with comparable computation times, the proposed architecture produced smoother longitudinal speed profiles and avoided unnecessary braking maneuvers. This behavior results from the direct constraint enforcement mechanism, which minimally modifies the nominal cruise reference instead of switching between precomputed geometric candidates. A fundamental advantage of the proposed approach emerges in fully obstructed scenarios. Unlike purely geometric trajectory generation methods, the architecture always retains a structured safety fallback. The combination of HOCBF constraints and viability-based acceleration bounds guarantees that, even when steering avoidance is no longer feasible, a physically admissible braking maneuver remains available. This ensures that the vehicle can systematically reduce its speed and reach a safe stop without violating actuation limits. The Webots simulations further validated the robustness of the framework in a higher-fidelity environment. The vehicle successfully performed lane keeping, overtaking, intersection handling, and safe stopping while solving the two optimization problems online. Most importantly, the comparison between configurations with and without viability bounds demonstrated that viability is essential to prevent unrealistic braking commands and restore feasibility under hard acceleration constraints. Despite the encouraging results, several limitations remain. The current validation is entirely simulation-based and relies on ideal state estimation. The dynamic model is simplified and does not explic-

itly account for complex tire nonlinearities or parameter uncertainty. Moreover, obstacle representations are geometric and deterministic, without incorporating perception uncertainty or prediction errors in dynamic traffic scenarios. Future work will focus on three main directions. First, experimental validation on a real vehicle platform will be pursued to assess robustness under sensing noise, delays, and actuator dynamics. Second, the integration of uncertainty-aware safety margins and disturbance-robust barrier formulations will be investigated to improve reliability in real-world traffic. Third, the extension to multi-agent scenarios with dynamic obstacle prediction and cooperative traffic interactions will be explored. In summary, the results confirm that the proposed HOCBF-based architecture, enhanced by viability reasoning, provides a computationally efficient and structurally robust solution for urban autonomous driving under hard physical constraints, bridging the gap between formal safety guarantees and practical feasibility.

Chapter 6

Viability via Flatness

This chapter builds upon the results presented in the previous one. In the earlier analysis, the HOCBF-based methodology was developed and validated only for a specific case study, namely an autonomous vehicle model. Moreover, the viability argument was constructed with respect to only one of the two control inputs, thereby restricting the admissible control choices and limiting the generality of the approach. These limitations motivated the extension of the framework to a broader class of nonlinear robotic systems.

Ensuring safety in nonlinear robotic systems operating under hard input constraints remains a central challenge, particularly in safety-critical applications such as autonomous driving. Recent advances in learning-based approaches, including deep neural networks and reinforcement learning [13, 42], have demonstrated remarkable empirical performance in complex driving scenarios. By leveraging large datasets, these methods can learn sophisticated policies and adapt to highly diverse environments. However, their limited interpretability, sensitivity to distributional shifts, and lack of formal safety guarantees restrict their deployment in safety-critical contexts [43]. As a result, purely learning-based controllers are often complemented with explicit safety layers, reintroducing the need for principled constraint-handling techniques at the control level.

Among model-based strategies, Model Predictive Control represents a well-established framework for autonomous driving, as it explicitly incorporates system dynamics and constraints while optimizing predicted behavior over a finite horizon [14]. MPC has been successfully applied to trajectory tracking, obstacle avoidance, and constrained motion planning [35]. Despite its flexibility and theoretical appeal, nonlinear MPC formulations for realistic driving scenarios typically lead to high-dimensional optimization problems that must be solved repeatedly in real time. When detailed vehicle dynamics, multiple safety constraints, and uncertainty are considered, the associated computational burden can significantly limit real-time applicability.

In light of the aforementioned limitations, we deliberately retain the High-Order Control Barrier Function and viability-based framework, while addressing the feasibility issue in a structured manner. In particular, the loss of feasibility induced by hard input constraints is tackled through an architecture that blends two complementary ideas.

On the one hand, we exploit the differential flatness property to reformulate the control design problem in the space of flat outputs. For systems admitting a flat representation, the original nonlinear dynamics can be locally transformed into a chain of integrators driven by a virtual input. This canonical representation enables a structured and computationally efficient formulation of HOCBF constraints, allowing high-relative-degree safety specifications to be imposed directly on the integrator-chain dynamics in a convex optimization setting.

On the other hand, we incorporate a viability-inspired mechanism to preserve feasibility in the presence of hard actuator bounds. Rather than solving the fully constrained problem in the original nonlinear coordinates which would typically result in a nonconvex and computationally demanding online optimization we compute conservative bounds on the admissible virtual input such that at least one safe evolution remains feasible. These bounds are then embedded into the safety filter as dynamically updated box constraints, ensuring that the feasible set of the online optimization problem remains nonempty even under realistic input saturations.

6.1 Integrator Representation of Flat Systems

Consider a nonlinear control system

$$\dot{x} = f(x, u), \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m. \tag{6.1}$$

By Definition 7 (see Chapter 2), system (6.1) is said to be differentially flat if there exists an output $y \in \mathbb{R}^m$, called flat output, such that both the state and the input can be expressed as algebraic functions of y and a finite number of its time derivatives.

A fundamental consequence of differential flatness is that, in flat coordinates, the system can be equivalently represented in Brunovský normal form, i.e., as a set of decoupled chains of integrators. Let k denote the highest derivative of the flat output required to explicitly reconstruct the physical input u. We introduce a virtual control input $u_v \in \mathbb{R}^m$ and the flat output dynamics can be rewritten as a chain of integrators:

$$y^{(k)} = u_v. \tag{6.2}$$

This representation is particularly attractive because it provides a linear, controllable, and structured system in the flat output space, regardless of the complexity of the original nonlinear dynamics (6.1).

To express (6.2) as a first-order system, we define the extended state

$$x_e := \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \\ \vdots \\ y^{(k-1)} \end{bmatrix} \in \mathbb{R}^{km}. \tag{6.3}$$

More explicitly, the system can be written in the linear time-invariant form

$$\dot{x}_e = A_e x_e + B_e u_v, \tag{6.4}$$

with

$$A_e := \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \qquad B_e := \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I \end{bmatrix}, \tag{6.5}$$

where each block I denotes the identity matrix in $\mathbb{R}^{m\times m}$.

It is important to remark that the existence of the mappings $\varphi_x(\cdot)$ and $\varphi_u(\cdot)$ in (2.27) requires local invertibility of the flatness transformation.

Throughout this thesis, we assume that the system evolves within a region of the state space where the flatness map is locally invertible, the system possesses full relative degree with respect to the chosen flat output, and the derivatives of the flat output required for state and input reconstruction exist and remain bounded.

These assumptions are standard in the differential flatness literature and are satisfied by a broad class of nonlinear control systems. In particular, fully actuated mechanical systems, controllable linear systems, and several classes of underactuated robotic systems (such as wheeled mobile robots and quadrotors) admit flat outputs for which the flatness transformation is locally invertible and the relative degree is well defined [18, 44, 45].

Given a virtual input signal $u_v(t)$ and the corresponding flat output evolution $y(t)$ obtained by integrating (6.2), the physical input can be recovered using the reconstruction map. More precisely, since $u_v = y^{(k)}$, we obtain

$$u(t) = \varphi_u\left(y(t), \dot{y}(t), \ldots, y^{(k)}(t)\right) = \varphi_u\left(y(t), \dot{y}(t), \ldots, u_v(t)\right). \tag{6.6}$$

This step is purely algebraic and does not require solving a dynamic inversion problem. Therefore, the computational complexity of mapping the safety-filtered virtual input $u_v^\star$ into a physically executable command $u^\star$ is typically negligible compared to an optimization stage.

The main reason for working in the integrator space is that high-order safety constraints become straightforward to handle. Indeed, many constraints in robotics (obstacle avoid-

ance, curvature limitations, acceleration bounds) have relative degree larger than one when expressed in terms of the original nonlinear state x. In contrast, when expressed in terms of the integrator-chain state x_e, these constraints often become polynomial functions of the derivatives of y, and their time derivatives are affine in the virtual input u_v.

6.1.1 Physical Constraints in Differentially-Flat Trajectory Planning

While the integrator-chain representation significantly simplifies safety filtering, it introduces a fundamental issue: the virtual input u_v does not coincide with the physical actuator command applied to the system. Therefore, enforcing safety constraints in the flat-output space alone is insufficient unless the resulting trajectories are compatible with the physical limitations of the original system. Infact real robotic systems are inherently subject to strict actuation bounds, including torque limits, steering constraints, and saturation effects. These limitations can be expressed in the general form

$$u(t) \in U := \{u \in \mathbb{R}^m \mid u_{\min} \preceq u \preceq u_{\max}\}, \tag{6.7}$$

where $\preceq$ denotes elementwise inequality.

Assume that the system is differentially flat with flat output $y \in \mathbb{R}^m$ then, the physical input can be expressed as (6.6) and the physical admissibility condition (6.7) can therefore be rewritten as

$$u_{\min} \preceq \varphi_u\left(y, \dot{y}, \ddot{y}, \ldots, y^{(k)}\right) \preceq u_{\max}. \tag{6.8}$$

By integrating the state-dependent input limits (6.8) into the standard safety-filtering framework, we obtain the following constrained optimization problem:

$$\begin{aligned} u_v^\star = &\arg\min_{u_v \in \mathbb{R}^m} \|u_v - u_{v,\text{des}}\|^2 \\ \text{s.t.} \quad &\begin{cases} A_{\text{hocbf}}(x_e) u_e \geq b_{\text{hocbf}}(x_e) \\ u_{\min} \preceq \varphi_u\left(y, \dot{y}, \ldots, y^{(k-1)}, u_v\right) \preceq u_{\max} \end{cases} \end{aligned} \tag{6.9}$$

While the High-Order Control Barrier Function constraints ($A_{\text{hocbf}}(x_e)u_e \geq b_{\text{hocbf}}(x_e)$) remain affine with respect to the virtual input u_v in the integrator space, the physical feasibility requirements introduce a structural complexity becouse in general, the reconstruction mapping $\varphi_u(\cdot)$ is a nonlinear function of its arguments. Consequently, the optimization problem (6.9) no longer conforms to the standard convex QP formulation typically associated with CBF-based safety filters. Instead, it takes the form of a Nonlinear Program (NLP). Nevertheless, for systems with a relatively simple structure as vehciles the additional computational burden remains limited in practice.

This observation suggests that the source of non-convexity does not lie in the safety constraints themselves, but in the nonlinear mapping that translates the virtual input

into the physical control action. Therefore, instead of constraining the physical input through the nonlinear relation $\varphi_u(\cdot)$, we seek a representation of physical admissibility that is expressed directly in the virtual-input space. If a subset of virtual inputs can be identified such that every element of this subset yields a physically admissible control action, then physical feasibility can be guaranteed without destroying the affine structure of the safety constraints.

Let us define the exact set of physically admissible virtual inputs as

$$\mathcal{U}_v(x_e) = \{u_v \in \mathbb{R}^m \mid u_{\min} \preceq \varphi_u(x_e, u_v) \preceq u_{\max}\}. \tag{6.10}$$

The set $\mathcal{U}_v(x_e)$ is, in general, nonlinear and not necessarily convex, since it is implicitly defined through the nonlinear reconstruction mapping $\varphi_u(\cdot)$. As a consequence, enforcing the exact constraint $u_v \in \mathcal{U}_v(x_e)$ would typically require solving a nonlinear program online.

To preserve the quadratic structure of the safety-filtering problem, we replace $\mathcal{U}_v(x_e)$ with the box-constrained set

$$\tilde{\mathcal{U}}_v(x_e) = \{u_v \in \mathbb{R}^m \mid u_{v,\min}(x_e) \preceq u_v \preceq u_{v,\max}(x_e)\}. \tag{6.11}$$

The bounds $u_{v,\min}(x_e)$ and $u_{v,\max}(x_e)$ are constructed and chosen conservatively so that the inclusion

$$\tilde{\mathcal{U}}_v(x_e) \subseteq \mathcal{U}_v(x_e)$$

holds for all admissible x_e. Therefore, $\tilde{\mathcal{U}}_v(x_e)$ represents an inner approximation of the exact physically admissible virtual-input set any $u_v \in \tilde{\mathcal{U}}_v(x_e)$ is guaranteed to generate a physically feasible control action for the original system.

In the next section, we introduce a viability-inspired method based on discrete motion templates, which provides a systematic procedure to compute such bounds and to enforce them online in combination with HOCBF constraints. This approach is inspired by viability theory and can be interpreted as a natural extension of the methodology introduced in the previous chapter. While the earlier framework was tailored to a specific system and primarily addressed a single physical input, the present formulation generalizes the idea to the broader class of differentially-flat systems that can be transformed into chains of integrators. In this setting, feasibility considerations can be incorporated at the level of the virtual integrator inputs in a systematic and unified manner.

In particular, by selecting state-dependent bounds that prevent the system from exiting an inner approximation of the capture basin of the safe set, the online optimization problem can be designed to remain always feasible. This mechanism ensures that at least one admissible recovery maneuver remains available, thereby preserving long-term safety rather than enforcing only instantaneous constraint satisfaction.

6.2 Incorporating Viability-based Bounds

Consider the integrator-chain representation of the flat output dynamics

$$y^{(k)} = u_v, \tag{6.12}$$

with extended state

$$x_e = \begin{bmatrix} y & \dot{y} & \dots & y^{(k-1)} \end{bmatrix}^T. \tag{6.13}$$

Exact computation of viability kernels is known to be computationally intractable also for high dimenional linear systems. To obtain a tractable approximation, we adopt a template-based strategy. The selection of motion templates is inherently system dependent and reflects the specific structure of the differentially flat model under consideration. However, when specializing the framework to dynamic vehicle models, it is natural to adopt templates that are compatible to physically vehicle meaningful maneuvers.

In particular given the current extended state $x_e(t_0)$ at time t_0, we define a family of candidate future trajectories parameterized by the virtual input. For an integrator chain of order k, the general solution of (6.12) over a finite horizon $T > 0$ can be written as

$$y(t) = \sum_{i=0}^{k-1} \frac{y^{(i)}(t_0)}{i!}(t - t_0)^i + \int_{t_0}^{t} \frac{(t-\tau)^{k-1}}{(k-1)!} u_v(\tau)\, d\tau. \tag{6.14}$$

The key difficulty is that $u_v(\cdot)$ is, in principle, an infinite-dimensional decision variable. Instead of optimizing over all possible functions, we restrict attention to a finite set of structured input profiles, called motion templates. These templates represent physically meaningful maneuvers such as constant-derivative polynomial profiles (e.g., constant acceleration, jerk, or snap), exponentially or sigmoidally decaying inputs or bounded trajectories induced by standard safe control profiles.

In particular, we consider templates of the form

$$y(t) = \sum_{i=0}^{k-1} \frac{y^{(i)}(t_0)}{i!}(t - t_0)^i + \xi_k(t - t_0; u_v), \quad t \in [t_0, t_0 + T], \tag{6.15}$$

where $\xi_k(\cdot)$ is a polynomial or bounded excitation term induced by a constant or piecewise constant virtual input.

For instance, assuming that the virtual input is constant over the horizon, i.e.

$$u_v(\tau) = \bar{u}_v, \qquad \tau \in [t_0, t_0 + T], \tag{6.16}$$

the resulting trajectory is explicitly given by

$$y(t) = \sum_{i=0}^{k-1} \frac{y^{(i)}(t_0)}{i!}(t - t_0)^i + \frac{\bar{u}_v}{k!}(t - t_0)^k. \tag{6.17}$$

This expression corresponds to a polynomial of degree k, which generalizes classical uniformly accelerated motion ($k = 2$) and constant jerk motion ($k = 3$).

To connect this approach explicitly to viability theory, recall that the capture basin of a target set $\mathcal{Q} \subseteq \mathcal{C}_e$ is the set of states from which the system can reach $\mathcal{Q}$ while staying inside $\mathcal{C}_e$. In the present context, the target set $\mathcal{Q}$ can be interpreted as a "recoverable" region, such as a sufficiently slow state or a state with sufficient distance from obstacles.

Instead of computing $\mathrm{Capt}(\mathcal{C}_e, \mathcal{Q})$ exactly, we approximate it by checking whether the system can reach $\mathcal{Q}$ by applying a predefined template input.

For example, in collision avoidance problems, a natural recoverable set is defined by

$$\mathcal{Q} := \{x_e \mid \text{relative speed toward the obstacle is non-positive}\}, \tag{6.18}$$

which corresponds to stopping (or moving away) before impact.

If the braking template can guarantee that the system reaches $\mathcal{Q}$ before violating the obstacle constraint, then the current state is inside an approximation of the capture basin.

Therefore, the viability module can be implemented by the following procedure:

1. At each control step, compute candidate template trajectories using a set of virtual inputs till the most extreme.

2. Check whether the resulting trajectories satisfy the safety constraints over the horizon $[t_0, t_0 + T]$.

3. If a violation occurs, tighten the admissible range of u_v accordingly.

This procedure yields conservative bounds that ensure that the online HOCBF-QP remains feasible.

It is important to note that the proposed method provides an approximation rather than an exact computation of viability kernels. The resulting bounds may be conservative, since restricting to a finite set of templates does not capture all possible admissible control signals. Nevertheless, this conservativeness is often acceptable in practice, as safety-critical applications prioritize robustness and feasibility over optimality. Furthermore, the choice of templates can be enriched to reduce conservativeness, for instance by including piecewise-constant virtual inputs or multi-phase maneuvers.

Now is possible to show how the viability bounds can be integrated with HOCBF constraints into a single quadratic program, yielding a computationally efficient safety filter with improved feasibility properties.

We formalize the proposed safety filter as an online optimization problem that integrates High-Order Control Barrier Function constraints with the viability-based virtual input bounds introduced in Section 6.2.

6.3 Case Study: Dynamic Bicycle Model

This section instantiates the general viability-aware HOCBF framework introduced in this section on a representative and practically relevant system: the dynamic bicycle model, widely adopted in autonomous driving applications.

We consider a four-wheeled ground vehicle modeled through the classical front-steered bicycle approximation (see Fig.6.1), the model is slightly different from the previous section to better adapt it to this type of solution [46].

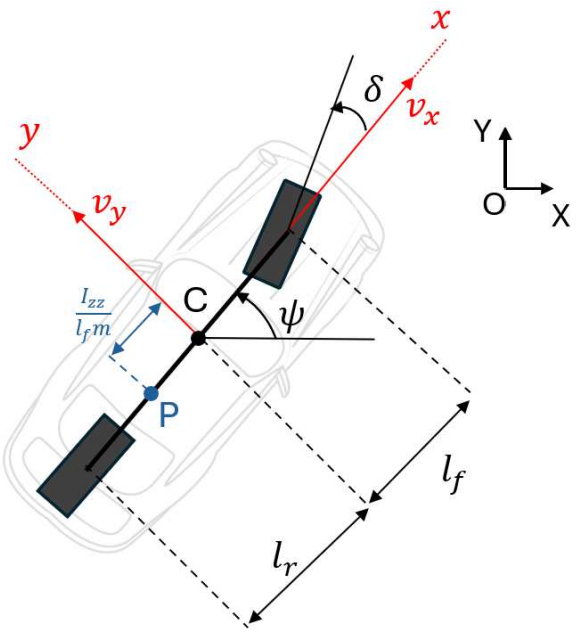


Figure 6.1: Dynamic Bicycle Model

The vehicle state vector is defined as

$$x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T \in \mathbb{R}^6, \tag{6.19}$$

where:

- $x_1 = X$: vehicle position along the global inertial X-axis;
- $x_2 = Y$: vehicle position along the global inertial Y-axis;
- $x_3 = \psi$: yaw angle (vehicle heading) with respect to the inertial frame;
- $x_4 = v_x$: longitudinal velocity expressed in the vehicle body-fixed frame;
- $x_5 = v_y$: lateral velocity expressed in the vehicle body-fixed frame;
- $x_6 = \omega$: yaw rate.

The dynamic bicycle model is given by a set of nonlinear equations:

$$\begin{cases} \dot{X} = v_x \cos\psi - v_y \sin\psi, \\ \dot{Y} = v_x \sin\psi + v_y \cos\psi, \\ \dot{\psi} = \omega, \\ m\dot{v}_x = F_{xf}\cos\delta - F_{yf}\sin\delta + F_{xr} - F_{\text{drag}} + mv_y\omega, \\ m\dot{v}_y = F_{xf}\sin\delta + F_{yf}\cos\delta + F_{yr} - mv_x\omega, \\ I_{zz}\dot{\omega} = l_f F_{yf}\cos\delta + l_f F_{xf}\sin\delta - l_r F_{yr}, \end{cases} \tag{6.20}$$

where m denotes the vehicle mass and I_{zz} is the yaw moment of inertia about the vertical axis. The geometric parameters l_f and l_r define the distances from the center of mass to the front and rear axles, respectively, thus determining the lever arms through which tire forces generate yaw moments. The steering input is represented by the front wheel angle δ. The interaction between the tires and the road is captured through the longitudinal forces F_{xf} and F_{xr} and the lateral forces F_{yf} and F_{yr} acting at the front and rear axles. In addition, F_{drag} models the resistive effects due to aerodynamic drag and rolling resistance, contributing to the longitudinal force balance.

We assume that the traction/braking actuation is applied symmetrically on the front and rear wheels, resulting in equal longitudinal forces at the two axles:

$$F_{xf} = F_{xr}. \tag{6.21}$$

The longitudinal force is modeled as a torque-limited wheel actuation:

$$F_{xf} = \frac{t_r T_{\max}}{2R_w}, \tag{6.22}$$

where:

- $t_r \in [-1, 1]$ is the normalized traction/braking command,
- $T_{\max}$ is the maximum available wheel torque,
- R_w is the wheel radius.

The command t_r therefore captures both acceleration ($t_r > 0$) and braking ($t_r < 0$) with saturation at the physical actuator limit.

The lateral forces are modeled through a linearized tire model, which is commonly adopted for moderate lateral accelerations. Specifically, we define the front and rear slip angles as

$$\alpha_f = -\frac{v_y + l_f\omega}{v_x} + \delta, \qquad \alpha_r = -\frac{v_y - l_r\omega}{v_x}, \tag{6.23}$$

where the sign convention is chosen such that positive steering δ induces positive lateral force.

Under the linear tire assumption, the lateral forces are proportional to the slip angles:

$$F_{yf} = K_f\alpha_f, \qquad F_{yr} = K_r\alpha_r, \tag{6.24}$$

where K_f and K_r denote the cornering stiffness coefficients of the front and rear tires, respectively.

By substituting (6.23) into (6.24), we obtain the explicit expressions

$$F_{yf} = K_f \left(-\frac{v_y + l_f \omega}{v_x} + \delta \right), \qquad F_{yr} = K_r \left(-\frac{v_y - l_r \omega}{v_x} \right). \tag{6.25}$$

A well known condition for the dynamic bicycle model is that (6.25) becomes singular as $v_x \to 0$. This reflects the fact that slip angles are not well-defined at zero longitudinal velocity.

The longitudinal resistive force is modeled through a standard polynomial drag approximation:

$$F_{\text{drag}} = f_0 + f_1 v_x + f_2 v_x^2, \tag{6.26}$$

where f_0, f_1, f_2 are drag coefficients that capture constant rolling resistance, linear viscous effects, and quadratic aerodynamic drag, respectively.

The physical control input of the dynamic bicycle model is defined as

$$u_{\text{real}} = \begin{bmatrix} t_r \\ \delta \end{bmatrix}. \tag{6.27}$$

In realistic implementations, both components are subject to strict bounds:

$$t_r \in [t_{r,\min}, t_{r,\max}], \qquad \delta \in [-\delta_{\max}, \delta_{\max}], \tag{6.28}$$

where typically $t_{r,\min} = -1$, $t_{r,\max} = 1$, and $\delta_{\max}$ is determined by the steering mechanism geometry.

These actuation constraints play a central role in the feasibility of safety filters. In particular, steering saturation directly limits the achievable curvature, while traction saturation limits the maximum braking and acceleration capabilities.

The numerical parameters adopted throughout this chapter are summarized in Table 6.1. These parameters correspond to a medium-size passenger vehicle and yield a realistic dynamic response in the considered operating conditions.

Table 6.1: PHYSICAL PARAMETERS OF THE VEHICLE BICYCLE MODEL

Parameter	**Value**	**Parameter**	**Value**
m	990 kg	I_{zz}	1800 kgm^2
l_f	1.25 m	l_r	1.3 m
K_f	140,000 N/rad	K_r	150,000 N/rad
R_w	0.3 m	$T_{\max}$	30,000 Nm
f_0	0.1	f_1	5
f_2	0.25		

6.3.1 Differential Flatness of the Dynamic Bicycle Model

The flatness of the dynamic bicycle model has been established in several works in the literature, and it is known that the vehicle dynamics can be parameterized by the planar coordinates of a suitable reference point located on the vehicle body axis.

Instead of selecting the center of mass position (X, Y) as flat output, we define a shifted point P located along the longitudinal body axis at a distance l_p from the center of mass. The use of such a point yields a convenient algebraic decoupling of the lateral and yaw dynamics.

Let (x_p, y_p) denote the planar coordinates of point P. The point is defined in global coordinates as

$$\begin{aligned} x_p &= X - l_p \cos\psi, \\ y_p &= Y - l_p \sin\psi, \end{aligned} \tag{6.29}$$

where

$$l_p = \frac{I_{zz}}{l_f m}. \tag{6.30}$$

Differentiating (6.29) and using the standard bicycle kinematics yields

$$\dot{x}_p = v_x, \qquad \dot{y}_p = v_y - l_p\omega, \tag{6.31}$$

where v_x and v_y are body-frame velocities and ω is the yaw rate.

We now define the flat output as

$$\mathcal{Y} := \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \end{bmatrix}. \tag{6.32}$$

Hence, the flat output corresponds to the planar position of point P. Specifying a sufficiently smooth trajectory $\mathcal{Y}(t)$ determines the full vehicle evolution, provided that suitable regularity conditions are satisfied and the longitudinal velocity remains non-zero.

the vehicle dynamics can be written as

$$\begin{cases} \dot{x}_p = v_x, \\ \dot{y}_p = v_y - l_p\omega, \\ \dot{v}_x = \frac{1}{m}\left(F_{xf}\cos\delta - F_{yf}\sin\delta + F_{xr} - F_{\text{drag}} + m v_y \omega\right), \\ \dot{v}_y = \frac{1}{m}\left(F_{xf}\sin\delta + F_{yf}\cos\delta + F_{yr} - m v_x \omega\right), \\ \dot{\omega} = \frac{1}{I_{zz}}\left(l_f F_{yf}\cos\delta + l_f F_{xf}\sin\delta - l_r F_{yr}\right). \end{cases} \tag{6.33}$$

The flatness property implies that the full vehicle state can be expressed as an algebraic function of the flat output and a finite number of its derivatives.

From (6.32)–(6.31), we immediately obtain

$$x_p = y_1, \qquad y_p = y_2, \qquad v_x = \dot{y}_1. \tag{6.34}$$

By differentiating y_2 and using the lateral and yaw dynamics, the remaining states v_y and ω can be reconstructed algebraically as

$$v_y = \frac{l_f m \dot{y}_1 \ddot{y}_2 + K_r(l_f + l_r)\dot{y}_2}{K_r(l_f + l_r)(I_{zz} - l_f l_r m) + (l_f m \dot{y}_1)^2}, \tag{6.35}$$

$$\omega = -\frac{l_f^2 m^2 \dot{y}_1 \ddot{y}_2 + K_r(l_f + l_r) l_f m \dot{y}_2}{K_r(l_f + l_r)(I_{zz} - l_f l_r m) + (l_f m \dot{y}_1)^2}. \tag{6.36}$$

The above expressions explicitly depend on $\dot{y}_1$, $\dot{y}_2$, and $\ddot{y}_2$. The reconstruction is purely algebraic and does not require integration of the nonlinear vehicle dynamics.

6.3.2 Input reconstruction and relative degree considerations

Beyond the algebraic reconstruction of the vehicle state, differential flatness requires that the physical control inputs u_{real} can also be expressed as functions of the flat output and a finite number of its time derivatives.

Differentiating y_1 twice yields

$$\ddot{y}_1 = \dot{v}_x = f_1(X, u), \tag{6.37}$$

where $f_1(\cdot)$ depends explicitly on the traction command t_r and on the steering angle δ through the longitudinal force balance here the relative degree is 2.

For the second output, differentiating twice gives

$$\ddot{y}_2 = \dot{v}_y - l_p \dot{\omega}.$$

Substituting the lateral and yaw dynamics shows that $\ddot{y}_2$ still depends on the state variables but does not provide an explicit algebraic dependence allowing inversion with respect to the physical inputs.

A third differentiation yields

$$y_2^{(3)} = f_2(X, u), \tag{6.38}$$

where $f_2(\cdot)$ depends explicitly on the steering angle δ and on the traction command through the tire forces here the relative degree is three.

Since the total relative degree equals the system order, the system is differentially flat with flat output $\mathcal{Y} = [y_1, y_2]^T$.

Consequently, the physical inputs can be reconstructed algebraically as

$$u_{\text{real}} = \beta\left(y_1, y_2, \dot{y}_1, \dot{y}_2, \ddot{y}_1, \ddot{y}_2, y_2^{(3)}\right), \tag{6.39}$$

for a suitable smooth mapping $\beta(\cdot)$ obtained by inverting the relations

$$\ddot{y}_1 = f_1(X, u), \qquad y_2^{(3)} = f_2(X, u).$$

Importantly, the steering and traction inputs depend on higher-order derivatives of the

flat output, in particular on the third derivative of y_2. This structural property has direct implications for safety filtering and constraint enforcement. The explicit expressions of f_1 and f_2, obtained by symbolic differentiation and substitution of the dynamic bicycle model, are lengthy but purely algebraic. For completeness, the derivation of equations (8.1), (8.2), and (8.3) is reported in Appendix 8.3.

Since the control inputs appear only after differentiating the flat outputs multiple times, safety constraints expressed in terms of y_1 or y_2 naturally require high-order Control Barrier Functions. In particular, constraints on longitudinal velocity or inter-vehicle distance lead to second- or third-order derivative conditions. This motivates embedding the flat-output dynamics into a higher-order integrator representation.

For control design purposes, we represent each planar flat coordinate as an integrator chain. The minimal representation consistent with the relative degrees would involve a second-order chain for y_1 and a third-order chain for y_2. However, in order to ensure sufficient smoothness and to obtain continuous acceleration and jerk profiles, we adopt a uniform fourth-order representation.

Specifically, we model the flat output dynamics as

$$\frac{d^4}{dt^4}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = u_e, \qquad u_e \in \mathbb{R}^2, \tag{6.40}$$

where u_e is a virtual control input corresponding to the snap (fourth derivative) of the planar trajectory.

Equivalently, defining the extended state

$$x_e := \begin{bmatrix} y_1 & y_2 & \dot{y}_1 & \dot{y}_2 & \ddot{y}_1 & \ddot{y}_2 & \dddot{y}_1 & \dddot{y}_2 \end{bmatrix}^\top \in \mathbb{R}^8, \tag{6.41}$$

the integrator dynamics become

$$\dot{x}_e = \begin{bmatrix} \dot{y}_1 & \dot{y}_2 & \ddot{y}_1 & \ddot{y}_2 & \dddot{y}_1 & \dddot{y}_2 & u_{e,1} & u_{e,2} \end{bmatrix}^\top. \tag{6.42}$$

The choice of a quadruple-integrator structure is particularly convenient for two reasons: firstly it guarantees sufficiently smooth trajectories with continuous acceleration and jerk profiles, which are desirable for vehicle motion planning and passenger comfort. Secondly it provides a natural interface for viability-based braking templates and high-order safety filters formulated at the snap level.

6.4 HOCBF-based Safety Constraints for the Extended Integrator Model

Once the dynamic bicycle model has been rewritten in the flat output space as a chain of integrators, safety and feasibility constraints can be imposed directly on the extended integrator state. Now we construct a set of High-Order Control Barrier Functions that

encode the safety requirements and physical model constraint of the bicycle model considered in this work.

In particular, we focus on three key classes of constraints:

1. obstacle avoidance constraints, ensuring collision-free motion;
2. curvature constraints, encoding steering limitations and vehicle maneuverability bounds;
3. acceleration constraints, enforcing comfort and feasibility limits.

These constraints are formulated directly in terms of the extended integrator state which evolves according to the quadruple integrator dynamics. We emphasize that, in this representation, the control variable is the snap input u_e. This choice yields safety constraints that are affine in u_e once sufficient derivatives are taken, thus preserving convexity of the online safety filter.

6.4.1 Obstacle avoidance constraint

Let the environment contain a set of circular obstacles. For simplicity, we consider a single obstacle centered at $x_{\text{ob}} \in \mathbb{R}^2$ with radius $R_{\text{ob}} > 0$. Obstacle avoidance requires the vehicle reference point P to remain outside the obstacle region.

Defining the position of the integrator system as

$$p := \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = x_{e,1:2}, \tag{6.43}$$

we introduce the candidate barrier function

$$h_o(x_e) = \|p - x_{\text{ob}}\|^2 - R_{\text{ob}}^2. \tag{6.44}$$

The function $h_o(x_e)$ depends only on the position p. For the quadruple integrator dynamics, the control u_e appears explicitly only after four time derivatives. Hence, the obstacle constraint has relative degree $r_o = 4$ with respect to the snap input u_e. Consequently, obstacle avoidance must be enforced through a fourth-order HOCBF inequality.

Using the recursive HOCBF construction, we define

$$\psi_{o,0}(x_e) := h_o(x_e), \tag{6.45}$$

and

$$\psi_{o,j}(x_e) := \dot{\psi}_{o,j-1}(x_e) + \lambda_{o,j}\psi_{o,j-1}(x_e), \qquad j = 1, 2, 3, 4, \tag{6.46}$$

where $\lambda_{o,j} > 0$ are design parameters.

The obstacle avoidance HOCBF constraint is then

$$\psi_{o,4}(x_e, u_e) \geq 0. \tag{6.47}$$

Since the integrator dynamics are linear, the expression (6.47) can always be written in affine form

$$A_o(x_e)u_e \geq b_o(x_e), \tag{6.48}$$

thus yielding a linear inequality in the QP decision variable u_e.

6.4.2 Curvature constraint and steering feasibility

Beyond obstacle avoidance, vehicle motion must respect steering limitations. In particular, the maximum steering angle $\delta_{\max}$ imposes an upper bound on the curvature that the vehicle can physically achieve.

For a planar trajectory $p(t) = [y_1(t), y_2(t)]^T$, the curvature is defined as

$$\kappa = \frac{\dot{y}_1\ddot{y}_2 - \dot{y}_2\ddot{y}_1}{\left(\dot{y}_1^2 + \dot{y}_2^2\right)^{3/2}}. \tag{6.49}$$

The curvature expression (6.49) depends on the first and second derivatives of the flat output and is therefore directly computable from the integrator state x_e.

Under the kinematic bicycle approximation, curvature and steering are related by

$$\kappa = \frac{\tan(\delta)}{l_f + l_r}. \tag{6.50}$$

Therefore, the steering saturation $\delta \in [-\delta_{\max}, \delta_{\max}]$ implies the curvature bound

$$|\kappa(t)| \leq \kappa_{\max}, \qquad \kappa_{\max} := \frac{\tan(\delta_{\max})}{l_f + l_r}. \tag{6.51}$$

To encode this constraint in HOCBF form, we define the curvature barrier function

$$h_c(x_e) := \kappa_{\max}^2 - \kappa^2. \tag{6.52}$$

The curvature κ depends on $\dot{y}$ and $\ddot{y}$. Under quadruple integrator dynamics, the snap input affects $\ddot{y}$ after two integrations. Hence, $h_c(x_e)$ has relative degree $r_c = 2$ with respect to u_e. As a consequence, curvature feasibility can be enforced via a second-order HOCBF.

Following the standard construction, we define

$$\psi_{c,0}(x_e) := h_c(x_e), \tag{6.53}$$

$$\psi_{c,1}(x_e) := \dot{\psi}_{c,0}(x_e) + \lambda_{c,1}\psi_{c,0}(x_e), \tag{6.54}$$

$$\psi_{c,2}(x_e) := \dot{\psi}_{c,1}(x_e) + \lambda_{c,2}\psi_{c,1}(x_e), \tag{6.55}$$

and enforce the constraint

$$\psi_{c,2}(x_e, u_e) \geq 0. \tag{6.56}$$

Again, due to the integrator structure, (6.56) is affine in u_e.

6.4.3 Acceleration constraint

In addition to curvature bounds, the vehicle must respect maximum admissible acceleration limits. These constraints may represent either physical limitations (tire-road friction, actuator constraints) or comfort-related restrictions .

Let $\dot{p} = [\dot{y}_1, \dot{y}_2]^T$ denote the planar velocity of point P and $\ddot{p} = [\ddot{y}_1, \ddot{y}_2]^T$ its planar acceleration. A natural scalar measure of acceleration magnitude is

$$a(t) = \|\ddot{p}(t)\|. \tag{6.57}$$

Adopting an acceleration projection along the direction of motion. Following the same formulation, we define

$$a(t) = \frac{\dot{y}_1\ddot{y}_1 + \dot{y}_2\ddot{y}_2}{\sqrt{\dot{y}_1^2 + \dot{y}_2^2}}. \tag{6.58}$$

The expression (6.58) corresponds to the tangential acceleration component, i.e., the component of $\ddot{p}$ along the instantaneous velocity direction. This is particularly meaningful in driving applications, since longitudinal acceleration and braking strongly influence safety and comfort.

Let $a_{\max} > 0$ be the maximum admissible acceleration magnitude. We then define the acceleration barrier function

$$h_a(x_e) := a_{\max}^2 - a(x_e)^2. \tag{6.59}$$

Since $a(x_e)$ depends on $\dot{y}$ and $\ddot{y}$, the acceleration constraint has relative degree $r_a = 2$ with respect to the snap input u_e. Thus, it can be enforced using a second-order HOCBF constraint.

We define

$$\psi_{a,0}(x_e) := h_a(x_e), \tag{6.60}$$

$$\psi_{a,1}(x_e) := \dot{\psi}_{a,0}(x_e) + \lambda_{a,1}\psi_{a,0}(x_e), \tag{6.61}$$

$$\psi_{a,2}(x_e) := \dot{\psi}_{a,1}(x_e) + \lambda_{a,2}\psi_{a,1}(x_e), \tag{6.62}$$

and enforce

$$\psi_{a,2}(x_e, u_e) \geq 0. \tag{6.63}$$

6.4.4 Lateral acceleration constraint

In addition to tangential acceleration bounds, the vehicle must also respect maximum admissible lateral acceleration limits. This constraint captures handling and comfort requirements and, in practice, is tightly related to tire–road friction.

We define the lateral acceleration magnitude as

$$a_n(t) = \frac{|\dot{y}_1 \ddot{y}_2 - \dot{y}_2 \ddot{y}_1|}{\sqrt{\dot{y}_1^2 + \dot{y}_2^2}}. \tag{6.64}$$

The expression (6.64) corresponds to the normal (centripetal) acceleration component, i.e., the component of $\ddot{p}$ orthogonal to the instantaneous velocity direction. This quantity is particularly meaningful in driving applications since it captures the lateral load experienced during cornering.

Let $a_{n,\max} > 0$ be the maximum admissible lateral acceleration magnitude. To avoid the non-smooth absolute value in (6.64), we enforce the equivalent squared constraint and define the lateral-acceleration barrier function

$$h_{a_n}(x_e) := a_{n,\max}^2 - a_n(x_e)^2. \tag{6.65}$$

Since $a_n(x_e)$ depends on $\dot{y}$ and $\ddot{y}$, the lateral acceleration constraint has relative degree $r_{a_n} = 2$ with respect to the snap input u_e. Thus, it can be enforced using a second-order HOCBF constraint.

We define

$$\psi_{a_n,0}(x_e) := h_{a_n}(x_e), \tag{6.66}$$

$$\psi_{a_n,1}(x_e) := \dot{\psi}_{a_n,0}(x_e) + \lambda_{a_n,1}\psi_{a_n,0}(x_e), \tag{6.67}$$

$$\psi_{a_n,2}(x_e) := \dot{\psi}_{a_n,1}(x_e) + \lambda_{a_n,2}\psi_{a_n,1}(x_e), \tag{6.68}$$

and enforce

$$\psi_{a_n,2}(x_e, u_e) \geq 0. \tag{6.69}$$

6.4.5 Combined HOCBF constraint set

The above constraints define a collection of admissible snap inputs u_e ensuring obstacle avoidance, curvature feasibility, and acceleration feasibility.

By stacking the HOCBF inequalities, we obtain a polyhedral constraint set of the form

$$A_{\text{hocbf}}(x_e)u_e \geq b_{\text{hocbf}}(x_e), \tag{6.70}$$

where $A_{\text{hocbf}}(x_e) \in \mathbb{R}^{N\times 2}$ and $b_{\text{hocbf}}(x_e) \in \mathbb{R}^N$, with N equal to the total number of HOCBF inequalities.

The resulting constraints are linear in the optimization variable u_e, thus allowing the online safety filter to remain a convex quadratic program.

6.5 Viability-based Emergency Braking via Constant-Snap Templates

The HOCBF constraints introduced in Section 6.4 enforce forward invariance of the safe set whenever the associated optimization problem remains feasible. However, feasibility is not guaranteed in general, especially in dynamic driving scenarios where the vehicle approaches obstacles at high speed under strict actuator limitations.

To address this issue, we incorporate a viability-inspired module that dynamically restricts the admissible snap input u_e based on the existence of an emergency maneuver capable of preventing constraint violation. In the present case study, we focus on the most fundamental recovery maneuver in autonomous driving: emergency braking.

The main idea is to compute, at each control step, a conservative bound on the snap input ensuring that the vehicle remains inside an approximation of the capture basin of the obstacle-avoidance safe set. Such a bound is then used to tighten the admissible range of the QP decision variable u_e, guaranteeing that a safe stopping trajectory remains feasible.

At each control step t_0, consider the flat-output state of the quadruple integrator

$$p(t_0) = p_0, \quad \dot{p}(t_0) = v_0, \quad \ddot{p}(t_0) = a_0, \quad \dddot{p}(t_0) = j_0. \tag{6.71}$$

To assess imminent collision risk, we first predict the vehicle motion under the nominal assumption of zero snap,

$$p^{(4)}(t) = 0, \qquad t \geq t_0. \tag{6.72}$$

Under this assumption, the predicted trajectory evolves as

$$p(t) = p_0 + v_0\tau + \frac{1}{2}a_0\tau^2 + \frac{1}{6}j_0\tau^3, \qquad \tau := t - t_0. \tag{6.73}$$

Let the obstacle be centered at x_{ob} with radius R_{ob}. The predicted time-to-collision t_{col} is defined as the smallest strictly positive solution of

$$\|p(t) - x_{\text{ob}}\|^2 - R_{\text{ob}}^2 = 0. \tag{6.74}$$

If no solution exists within a prescribed horizon, the system is considered locally safe under nominal evolution. Otherwise, t_{col} represents the time at which collision would occur if no corrective snap input were applied. This quantity provides a natural finite horizon for viability assessment.

When t_{col} is finite, we evaluate whether there exists a constant snap maneuver that preserves obstacle clearance over the interval $[t_0, t_0 + t_{\text{col}}]$.

Define the relative position vector

$$r_0 := p_0 - x_{\text{ob}}, \tag{6.75}$$

and the outward unit direction

$$n := \frac{r_0}{\|r_0\|}. \tag{6.76}$$

We restrict the candidate emergency maneuver to constant snap inputs aligned with the radial direction,

$$u_e(t) = s_n\, n, \qquad \tau \in [0, t_{\text{col}}], \tag{6.77}$$

where $s_n \in [s_{\min}^{viab}, s_{\max}^{viab}]$ is a scalar snap amplitude within admissible bounds.

Under (6.77), the predicted motion becomes

$$p(t) = p_0 + v_0\tau + \frac{1}{2}a_0\tau^2 + \frac{1}{6}j_0\tau^3 + \frac{1}{24}s_n n\tau^4. \tag{6.78}$$

The maneuver is declared feasible if

$$\|p(t) - x_{\text{ob}}\| \geq R_{\text{ob}} + \delta_{\text{safe}}, \qquad \forall \tau \in [0, t_{\text{col}}], \tag{6.79}$$

where $\delta_{\text{safe}} > 0$ is a prescribed clearance margin.

Among all admissible s_n satisfying (6.79), we select the smallest value in magnitude that ensures feasibility. This choice minimizes conservativeness while preserving the existence of a collision-free quartic trajectory.

The procedure described above does not compute the full viability kernel of the obstacle-avoidance constraint. Instead, it constructs a sufficient condition for the existence of a safe trajectory within the restricted class of constant-snap motions aligned with the obstacle radial direction.

Consequently, the resulting constraint defines an inner approximation of the local capture basin under bounded snap inputs. If a feasible $s_n^\star$ exists, then there exists at least one admissible snap trajectory that prevents collision over the predicted horizon. If no such $s_n^\star$ can be found within $[s_{\min}, s_{\max}]$, the state lies outside this approximate viability region, indicating that constraint infeasibility is unavoidable under the constant-snap assumption.

The existence of the constant-snap viability maneuver implies that the projected snap along direction n must satisfy

$$n^T u_e \geq s_n^\star. \tag{6.80}$$

Constraint (6.80) is linear in u_e and can therefore be directly incorporated as an additional inequality in the QP.

By restricting the admissible snap inputs to those satisfying (6.80), the optimization problem is confined to a subset of inputs that preserve the existence of a collision-free constant-snap trajectory. Importantly, this filtering operation does not replace the HOCBF constraint, but complements it by safeguarding feasibility under actuator saturation.

6.6 Final Quadratic Program Formulation for Safe Vehicle Control

We present the final online optimization problem used to synthesize a safe control action for the dynamic bicycle model. The formulation integrates the HOCBF constraints introduced in Section 6.4 with the viability-based braking bound derived in Section 6.5. The resulting controller is a quadratic program solved at each control step, yielding a minimally invasive modification of the nominal snap command.

Recall that the flat output dynamics are modeled as a quadruple integrator

$$\frac{d^4}{dt^4}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = u_e, \qquad u_e \in \mathbb{R}^2,$$

where the virtual control input represents the planar snap of the flat output trajectory.

At each sampling time t_k, a nominal controller provides a desired snap input $u_{\text{des}}(t_k) \in \mathbb{R}^2$ which is typically obtained from a trajectory tracking law in the integrator space. In the absence of constraints, applying u_{des} would yield the desired tracking performance.

However, since u_{des} may violate safety or feasibility constraints, we compute a corrected snap input $u_e^\star$ through an optimization-based safety filter.

6.6.1 Quadratic cost function

The safety filter is designed to minimally deviate from the nominal command. Thus, we adopt the quadratic objective

$$u_e^\star = \arg\min_{u_e \in \mathbb{R}^2} \|u_e - u_{\text{des}}\|^2 \tag{6.81}$$

where $\|\cdot\|$ denotes the Euclidean norm. This objective ensures that the nominal behavior is preserved whenever it is already safe, while enforcing the smallest possible modification when constraints become active.

The safety requirements are encoded through the barrier functions defined in Section 6.4. Each barrier function yields an HOCBF constraint of the form

$$\psi_{i,r_i}(x_e, u_e) \geq 0, \qquad i \in \{o, c, a\}, \tag{6.82}$$

where $r_o = 4$ and $r_c = r_a = 2$.

As discussed earlier, due to the integrator-chain structure, each HOCBF inequality can be written as a linear constraint in the decision variable u_e:

$$A_i(x_e)u_e \geq b_i(x_e), \qquad i \in \{o, c, a\}. \tag{6.83}$$

These constraints guarantee forward invariance of the safe set as long as the QP remains

feasible. We incorporate the viability-based braking bound derived in Section 6.5. Let $\hat{r}(t_k)$ denote the unit vector from the obstacle to the vehicle at time t_k, and let $s_{\text{brake}}(t_k)$ be the emergency braking snap computed from the constant snap template.

Then, the viability requirement can be expressed as (6.80)

Collecting the above components, the proposed viability-aware safety filter for the dynamic bicycle model is given by the quadratic program

$$u_e^\star = \arg\min_{u_e \in \mathbb{R}^2} \|u_e - u_{\text{des}}\|^2 \tag{6.84}$$

$$\text{s.t.} \quad \begin{cases} \psi_{o,4}(x_e, u_e) \geq 0, \\ \psi_{c,2}(x_e, u_e) \geq 0, \\ \psi_{a,2}(x_e, u_e) \geq 0, \\ \psi_{a_n,2}(x_e, u_e) \geq 0, \\ n^T u_e \geq s_n^\star. \end{cases} \tag{6.85}$$

In the implementation, the feasibility-preserving viability constraint is embedded in the bound set. In particular, when the braking module detects an imminent collision, the upper bound is tightened so that

$$u_{e,\max} \leftarrow u_{e,\max}^{\text{viab}}, \tag{6.86}$$

where $u_{e,\max}^{\text{viab}}$ is obtained from the constant snap braking computation.

Finally, the physical bicycle input $u_{\text{real}} = [t_r, \delta]^T$ is reconstructed through the flatness map and it is then applied to the original bicycle model (6.20).

6.7 Comparative Simulation Study: HOCBF-Based Control and Nonlinear MPC

This section presents a comprehensive validation of the proposed viability-aware High-Order Control Barrier Function framework applied to the nonlinear dynamic bicycle model. The objective of this analysis is twofold. First, we assess tracking accuracy, strict constraint satisfaction, and physical feasibility of the HOCBF-based controller. Second, we provide a direct and systematic comparison with a nonlinear Model Predictive Control formulation implemented on the same vehicle model and subject to identical physical and safety constraints.

All simulations were conducted in MATLAB on an Intel i7 platform with 8GB RAM. The same dynamic bicycle model, identical physical parameters, and equivalent constraint definitions were adopted for both control architectures in order to ensure a fair and consistent comparison.

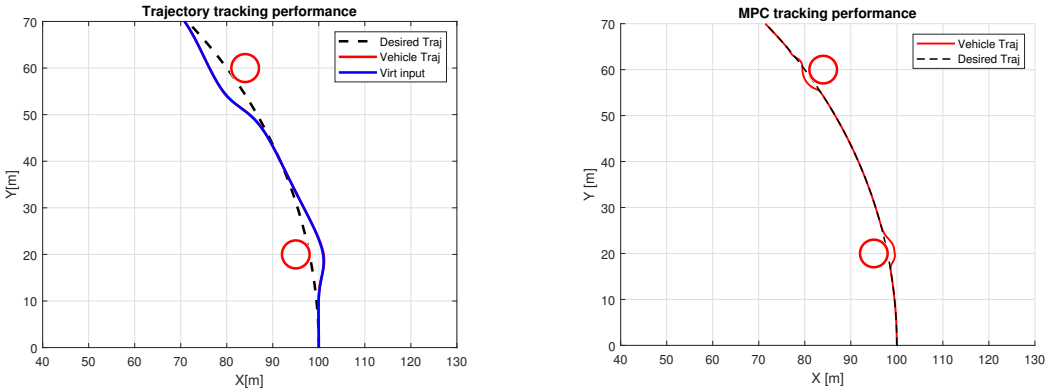
The simulated environment consists of a planar driving scenario including two static circular obstacles positioned along a nominal reference trajectory. The vehicle is initialized

with nonzero forward velocity and is required to track the reference path while satisfying safety, physical, and actuation constraints at all times.

The enforced constraints include obstacle avoidance through a fourth-order HOCBF, steering-induced curvature feasibility derived from steering angle limitations, longitudinal and lateral acceleration bounds reflecting both physical and comfort constraints, actuator saturation limits, and a viability-based longitudinal braking constraint. The scenario is intentionally constructed such that the reference trajectory approaches the obstacles closely, thereby activating the safety filters and testing the robustness of the constraint-handling mechanisms.

6.7.1 Performance

Figure 6.2 shows the nominal reference trajectory together with the closed-loop trajectories generated by the HOCBF-based controller and the nonlinear MPC. Both controllers successfully avoid the obstacles while maintaining close adherence to the reference path. In the HOCBF-based architecture, the trajectory of the nonlinear bicycle model overlaps almost perfectly with the flat integrator trajectory, confirming the consistency of the differential flatness mapping and the validity of the cascade optimization structure. Deviations from the reference appear only when obstacle constraints become active and correspond to the minimal modification required to guarantee forward invariance of the safe set. The MPC solution achieves comparable geometric tracking performance. However, in regions where constraints become active, the MPC trajectory exhibits sharper curvature transitions, reflecting the finite-horizon nature of the optimization and the sensitivity to cost-function weighting.



(a) HOCBFs and Viability trajectory.

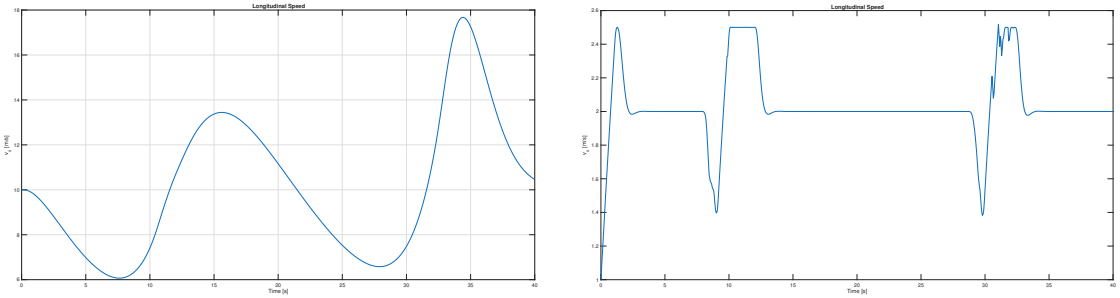
(b) MPC trajectory.

Figure 6.2: Comparison of trajectory tracking performance between HOCBF-based control and nonlinear MPC.

Figure 6.3 reports the longitudinal velocity evolution obtained with both control strategies. The HOCBF-based controller produces a smooth and progressively decreasing velocity profile when approaching obstacles. As the vehicle enters regions of reduced admissible acceleration, the longitudinal speed decreases in a continuous and predictable manner. This behavior is directly influenced by the viability-based tightening of acceleration bounds, which ensures that a feasible braking trajectory always exists within

physical limits.

In contrast, the MPC solution exhibits more pronounced oscillatory behavior in the velocity profile, particularly when the obstacle enters and exits the prediction horizon. Although constraints are respected, the repeated finite-horizon re-optimization introduces local speed adjustments that result in less regular velocity evolution.



(a) HOCBFs and Viability Longitudinal Velocity. **(b)** MPC velocity.

Figure 6.3: Comparison of longitudinal velocity profiles between HOCBF-based control and nonlinear MPC.

To evaluate physical feasibility and passenger comfort, both longitudinal and lateral accelerations are analyzed in Figure 6.4. The HOCBF-based controller maintains accelerations strictly within the prescribed physical and comfort bounds at all times. No overshoot or high-frequency oscillations are observed. Lateral acceleration remains compatible with steering geometry limitations, and longitudinal deceleration during obstacle avoidance remains physically feasible and smooth.

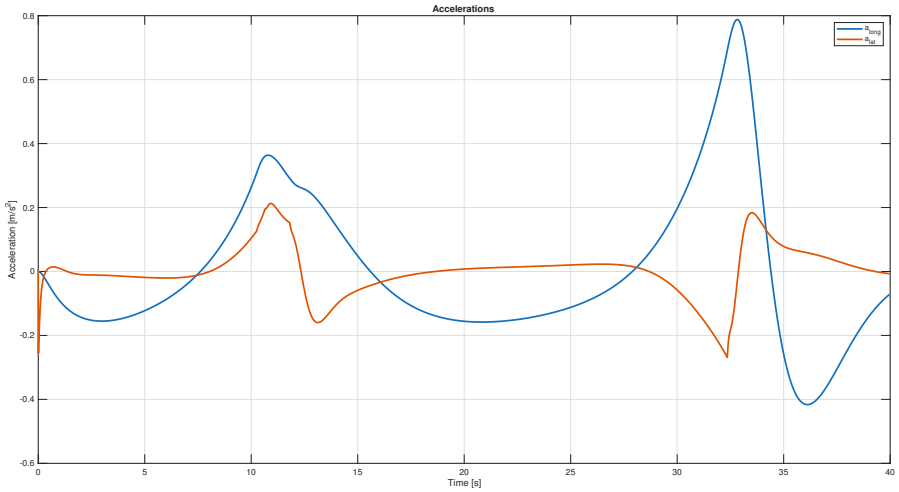


Figure 6.4: Acceleration HOCBF viability based control.

Figure 6.5 shows the curvature evolution together with the maximum admissible curvature derived from steering angle saturation. The curvature generated by the HOCBF-based controller never exceeds the admissible bound, confirming that steering feasibility is guaranteed at all times. The MPC solution also respects curvature limits but approaches saturation more abruptly when reacting to obstacles, resulting in sharper transitions.

The sampling time adopted for the proposed architecture is $dt = 0.01$ s, while the MPC controller operates with $dt = 0.05$ s. Despite the higher update frequency, the proposed method remains computationally efficient. The average computation time per control

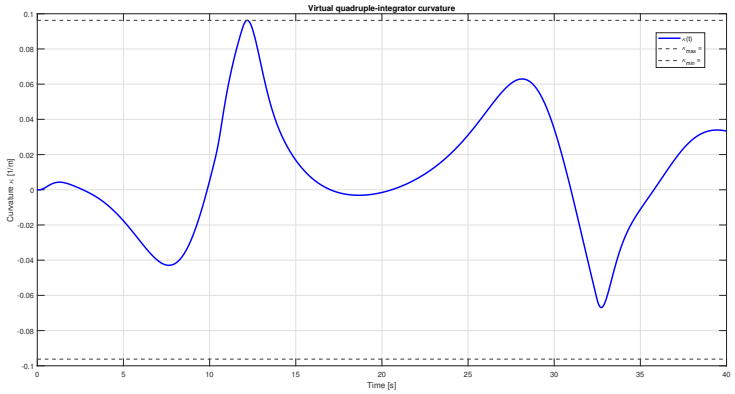


Figure 6.5: Curvature profile and maximum admissible bound.

step for the HOCBF-based approach is

$$T_{\text{HOCBF}} = [0.002] \text{ s},$$

whereas for the nonlinear MPC it is

$$T_{\text{MPC}} = [0.20] \text{ s}.$$

The HOCBF-based controller solves convex quadratic programs with fixed structure and exhibits nearly constant computation time. In contrast, MPC computation time increases with prediction horizon length and depends on initialization quality, making worst-case execution time more difficult to characterize.

We also compares the control architecture built in this chapter with a baseline HOCBF-based QP that does not include the viability-based braking constraint. This comparison highlights the critical role of the viability module in ensuring feasibility and safety under actuator limitations.

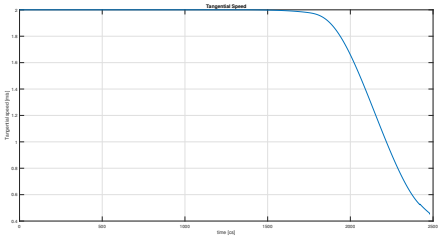
In the considered example, the viability module yields the symmetric bound

$$s_{\min}^{\text{viab}} = -0.1, \qquad s_{\max}^{\text{viab}} = +0.1, \tag{6.87}$$

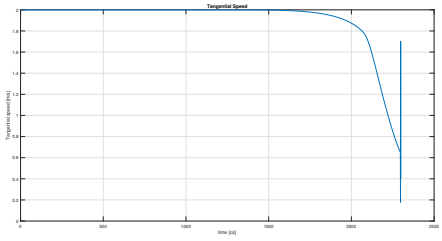
and the resulting closed-loop solution remains strictly within this interval.

Figure 6.6 summarizes the qualitative differences between the two architectures. Under HOCBF+Viability, the viability constraint becomes active sufficiently early, shaping the snap profile before the vehicle reaches the critical region. As a consequence, the longitudinal speed decreases more gradually, avoiding sudden deceleration. Importantly, since the snap never exceeds the admissible viability band (6.87), the corresponding physical inputs (traction/braking and steering) remain unsaturated. This leads to a consistent tracking of the commanded deceleration and guarantees non-collision in the presence of actuator bounds.

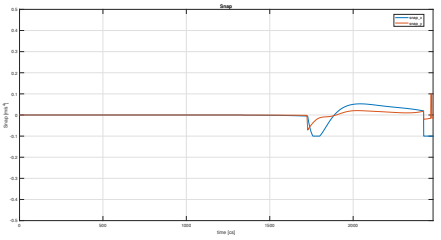
In contrast, in the HOCBF-only case the controller reacts later and attempts to re-



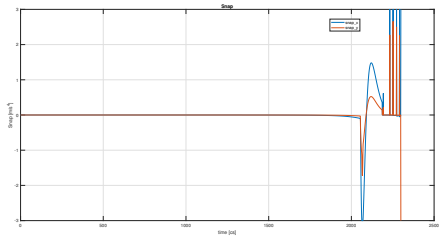
(a) HOCBF+Viability longitudinal speed profile.



(b) HOCBF-only longitudinal speed profile.



(c) HOCBFs+Viability snap profile.



(d) HOCBF-only snap profile.

Figure 6.6: Comparison between HOCBF+Viability and HOCBF-only. The viability constraint enforces the projected snap bound ± 0.1, leading to an earlier yet smoother deceleration that never requires unrealizable snap peaks and avoids saturation of the real vehicle inputs. Without viability filtering, the controller demands excessively large snap values, producing steep commanded speed reductions and saturating the physical inputs, which may prevent stopping in time in real execution.

cover safety by applying significantly larger snap magnitudes. While such aggressive snap commands may appear effective in the idealized integrator model, they exceed the viability-consistent range and translate into unrealistically steep speed reductions. In practice, these snap demands induce saturation of the actual vehicle inputs, so the real system cannot realize the requested deceleration profile. As a result, the vehicle fails to stop in time and collision avoidance is no longer guaranteed in the physical closed loop.

Overall, the comparison highlights that the viability module acts as a feasibility preserving filter: it enforces conservative, realizable snap profiles that avoid late, impulsive corrections and prevent actuator saturation, thereby bridging the gap between integratOor-level snap commands and the true vehicle actuation limits.

The comparison with nonlinear MPC also highlights the different nature of the guarantees provided by the two approaches. MPC handles dynamics, constraints, and performance objectives in a unified finite-horizon optimization problem, which makes it highly flexible and well suited to problems in which preview information and trajectory optimality are central. However, recursive feasibility and safety depend on the prediction horizon, terminal ingredients, cost tuning, and solver convergence. In dense or rapidly changing scenarios, these aspects may require careful design and can increase the computational burden. The proposed viability-aware HOCBF formulation follows a different philosophy. Safety constraints are enforced pointwise through barrier inequalities, while the viability module restricts the admissible virtual inputs so that future recoverability is not lost. This results in smaller optimization problems with predictable computation time and an explicit mechanism to avoid actuator-infeasible safety demands. The price paid for this structure is reduced planning expressiveness: the controller does not optimize

a full future trajectory and the evasive behavior is shaped indirectly through barrier gains, safety margins, and recovery templates. Nevertheless, this also gives the framework a degree of modularity, since the emergency or recovery templates used by the viability module can be customized according to the specific vehicle or robotic platform. In this way, platform-dependent actuation limits, maneuvering capabilities, and safety priorities can be encoded without changing the overall viability-based safety architecture. Consequently, while MPC offers greater freedom in optimizing maneuver geometry, the proposed method emphasizes formal local safety enforcement, recursive feasibility under bounded inputs, and real-time computational regularity.

6.7.2 Experimental Validation on a Scaled Vehicle Platform

To further validate the approach beyond numerical simulation, the proposed controller was implemented on a scaled autonomous vehicle platform.

Figure 6.7: The AgileX LIMO mobile platform, featuring a multimodal chassis and integrated ROS/ROS2 compatibility for autonomous navigation.

The experimental setup is organized as follows. The controller runs on an external PC and communicates with the AgileX LIMO platform through ROS1. The vehicle state is measured using a VICON motion-capture system, while obstacle conditions are generated by the control software in order to reproduce the same safety logic adopted in simulation. This setup allows the proposed controller to be tested in a real-time experimental loop without modifying the control architecture. During the experimental validation, the main vehicle states and control inputs were recorded, as shown in Fig. 6.8. The real-world results confirm smooth trajectory tracking, satisfaction of the imposed constraints, and real-time feasibility. In particular, the control inputs generated by the proposed method remain within the admissible bounds of the platform, without persistent saturation or abrupt variations. Moreover, the robot states and velocities exhibit smooth and continuous behavior, showing that the safety filter does not introduce undesirable oscillations or aggressive accelerations. A video of the experiment is available at https://zenodo.org/records/20159266 . These results are especially relevant in the

experimental setting, as they demonstrate that the proposed HOCBF-based safety filter with viability constraints can be implemented on real hardware while preserving dynamically smooth and physically admissible motions. Furthermore, the hardware validation confirms the key property already observed in simulation: safety constraints are enforced without requesting actuator-infeasible commands. The ROS1–VICON integration also demonstrates that the controller can be executed online in a real experimental loop. Compared with MPC-based approaches, the proposed HOCBF-based method solves a fixed-size convex QP with predictable runtime and guarantees forward invariance and recursive feasibility by construction, whereas MPC generally requires iterative nonlinear optimization. This makes the proposed approach particularly suitable for real-time safety-critical applications.

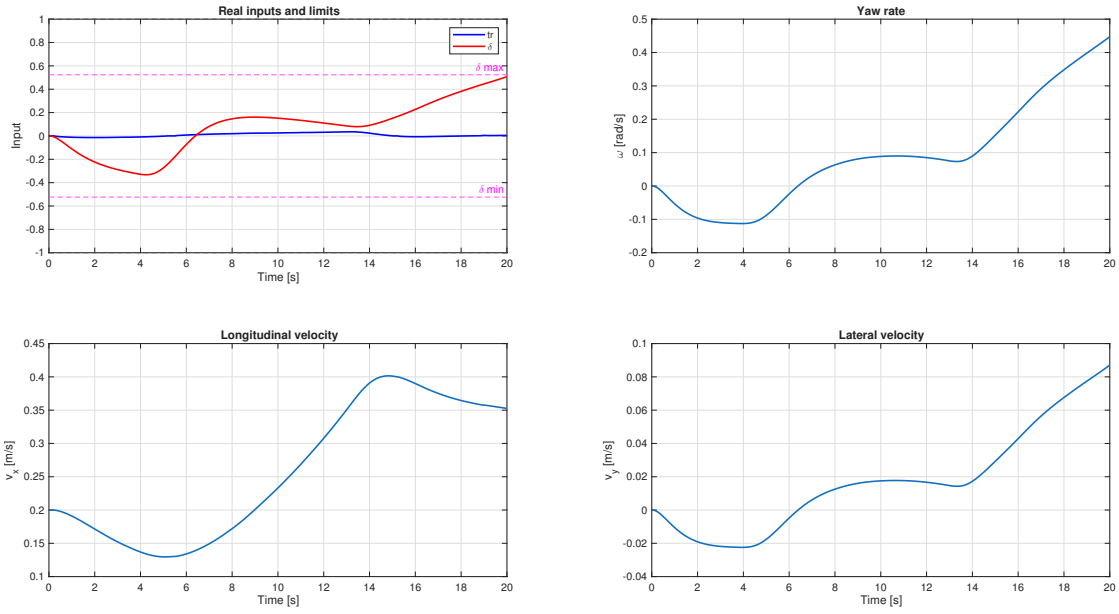


Figure 6.8: LIMO inputs and state.

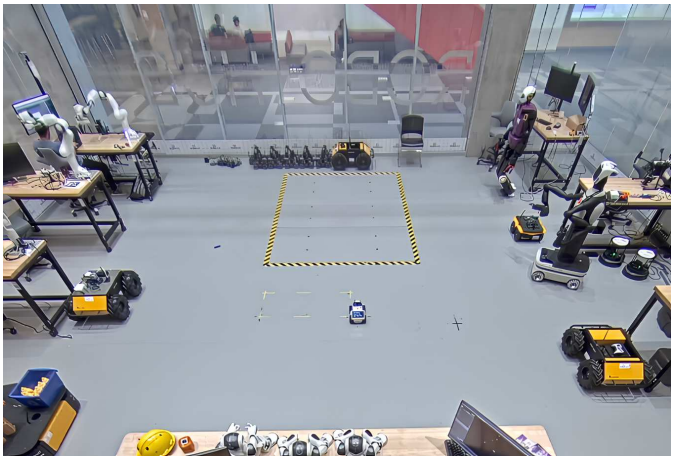


Figure 6.9: Integration of the LIMO with our architecture inside the University of Waterloo Robohub's.

6.8 Conclusions and Future Work

This chapter presented a viability-aware framework for safety-critical control of differentially flat nonlinear systems under hard actuation constraints. The main objective was to overcome a structural limitation of classical HOCBF-based approaches, namely the potential loss of feasibility when actuator saturation prevents the enforcement of safety constraints. By exploiting differential flatness, the original nonlinear dynamics were reformulated in the flat output space as an integrator chain. This representation enables the systematic construction of high-order Control Barrier Functions for Cartesian safety constraints while preserving an affine dependence on the virtual input. As a result, the online safety filter can be formulated as a small convex quadratic program with predictable computational cost. The key contribution lies in the integration of viability-inspired reasoning within the HOCBF framework. Instead of enforcing physical input constraints directly through the nonlinear flatness reconstruction map, conservative bounds on the admissible virtual input are computed using predefined recovery templates, such as constant-snap emergency braking profiles. By dynamically tightening these bounds near the boundary of the capture basin, the controller is prevented from entering states from which future safety enforcement would become impossible. This mechanism preserves convexity of the online optimization problem while significantly improving robustness to infeasibility. The methodology was validated on the dynamic bicycle model, reformulated as a quadruple integrator in planar coordinates. HOCBF constraints were designed to enforce obstacle avoidance, curvature feasibility, and acceleration limits, while viability-based bounds guaranteed recoverability under strict actuation constraints. Despite its effectiveness, the framework presents some limitations. The viability mechanism relies on a finite set of motion templates and therefore provides a conservative approximation of the true viability kernel. The flatness-based representation is locally valid and may exhibit singularities at low longitudinal velocity. Moreover, the emergency braking strategy is derived from a one-dimensional projection along the obstacle line-of-sight and may introduce conservativeness in scenarios where combined steering maneuvers would be preferable. Finally, accurate state estimation and sufficiently smooth higher-order derivatives are assumed. Future research directions include enriching the viability module with multi-phase braking and steering templates to reduce conservativeness, integrating learning-based approximations of viability bounds for faster online evaluation, incorporating robustness margins against model uncertainty and disturbances, extending the framework to multi-agent traffic scenarios, and performing full experimental validation on a real vehicle platform. In summary, the integration of differential flatness, HOCBF-based safety filtering, and viability-inspired feasibility constraints provides a computationally efficient and structurally robust methodology for safety-critical control under hard input limitations.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

This thesis addressed the problem of safety-critical control for autonomous vehicles in urban environments under realistic constraints, with a specific focus on reconciling two requirements that are often in tension: formal safety guarantees and feasibility under hard actuation limits and real-time computation. The main message emerging from this work is that safety enforcement at the control layer cannot be treated solely as a local, instantaneous constraint satisfaction problem. For deployment in realistic urban scenarios, safety mechanisms must be designed together with explicit feasibility and recoverability considerations. Rather than proposing a monolithic solution, the thesis developed a progressive control architecture, where each contribution extends the previous one by addressing a concrete limitation encountered when moving toward increasingly realistic settings. The first contribution focused on kinematic safety using Control Barrier Functions. Traffic rules and driving behaviors, including lane keeping, overtaking across dashed markings, obstacle avoidance, and intersection right-of-way or stop compliance, were encoded as barrier constraints within a single convex quadratic program. This formulation demonstrated modularity, formal forward invariance guarantees for the designed safe sets, and real-time computational suitability. The second contribution integrated infrastructure-assisted information into the same optimization-based safety layer through pairwise barrier functions. Occupancy information provided by smart infrastructure, such as blind or partially occluded intersections, was used to dynamically reshape the safe set. The resulting method preserved convexity and real-time tractability, while reducing conservativeness when additional information became available. The third contribution addressed the transition from kinematic to dynamic vehicle models, where higher relative degree constraints and actuator saturation can lead to infeasibility of barrier conditions. By combining High-Order Control Barrier Functions with a viability-inspired module that computes state-dependent admissible acceleration bounds, the proposed approach prevents late braking behaviors and keeps the system within regions where future constraint satisfaction remains possible. This restores recursive feasibility and preserves safety under hard input limitations. The final contribution exploited differential flatness to transform every flat system (in this case a

nonlinear vehicle) into an integrator-chain representation in the flat-output space, enabling small convex online optimization problems with predictable computation time. Viability-inspired bounds were incorporated at the level of virtual inputs through recovery templates, such as constant-snap emergency braking, improving robustness to infeasibility while maintaining computational efficiency. The framework was validated in simulation and experimentally on a scaled vehicle platform, demonstrating real-time capability and robustness. Overall, the thesis shows that barrier-based safety filtering and viability reasoning are complementary. Barrier functions provide efficient online enforcement of safety constraints, while viability-inspired mechanisms ensure recursive feasibility under strict actuation limits. The resulting methodology provides a principled and modular foundation for safety-critical control architectures oriented toward real-world autonomous driving deployment.

7.2 Future Works

Despite the encouraging results, several research directions remain open, especially when targeting full-scale deployment in realistic traffic environments. A first natural step is the implementation and validation of the proposed framework on a full-scale autonomous vehicle. This would allow the assessment of robustness under real sensing pipelines, communication delays, actuator dynamics, and unmodeled disturbances. Systematic benchmarking of computational latency and worst-case execution time on embedded automotive hardware would also be required. A second important direction concerns real infrastructure-to-vehicle integration. While infrastructure-assisted safety was validated in simulation, future work should implement real V2I and I2V communication channels, explicitly accounting for latency, packet loss, synchronization issues, and cybersecurity constraints. Barrier formulations that incorporate bounded communication delays and intermittent updates would further increase reliability. Another relevant extension involves uncertainty-aware safety constraints. The current framework largely assumes reliable state estimation and deterministic obstacle representations. Incorporating perception noise, prediction uncertainty for dynamic agents, and model uncertainty through robust or stochastic barrier formulations would strengthen safety guarantees in real traffic conditions. Further developments can also target richer recovery strategies within the viability-inspired module. Current feasibility guarantees rely on conservative bounds derived from a limited set of recovery templates. Extending the approach to multi-phase braking, combined lateral and longitudinal recovery maneuvers, and less conservative viability approximations would improve performance in highly constrained scenarios. Multi-agent extensions represent another key research direction. Urban driving inherently involves interactions among multiple vehicles. Extending the architecture to vehicle-to-vehicle cooperation requires interaction-aware constraints, improved prediction models, and coordinated safety mechanisms. Pairwise barrier functions and viability-based feasibility reasoning provide a promising foundation for such cooperative frameworks. In summary, this thesis establishes a coherent pathway from kinematic safety enforcement to dynamic, feasibility-aware, and computationally scalable safety-critical control. Future work will focus on bridging the remaining gap toward full real-world

deployment, strengthening robustness to uncertainty, enabling cooperative interaction, and validating the approach in infrastructure-enabled and multi-agent environments.

Chapter 8

Appendix

8.1 Implementation and Reproducibility

This section summarizes the main implementation details required to reproduce the numerical and experimental results presented in the thesis. The control algorithms were implemented with a fixed sampling time of

$$\Delta t = 0.01 \text{ s}.$$

All MATLAB and Webots simulations were executed on the same laptop equipped with an Intel i7-10510U processor and 8 GB of RAM.

Table 8.1: Implementation details for the main validation environments.

Environment	Software/Interface	Solver	Sampling time
MATLAB simulations	MATLAB, default solver settings	`quadprog` for CBF and HOCBF quadratic programs; `fsolve` for the flatness-based reconstruction step	0.01 s
Webots simulations	Webots R2023 with ROS2 Humble interface	`scipy.optimize` with SLSQP	0.01 s
Scaled-vehicle experiments	AgileX LIMO platform with ROS1 interface connected to the external control PC	Optimization solved on the external PC; resulting control inputs sent to the platform through ROS1	0.01 s

In the MATLAB implementation, the CBF-based and HOCBF-based safety filters were solved using `quadprog` with default MATLAB settings. The nonlinear algebraic equations required by the flatness-based reconstruction were solved using `fsolve`. In the Webots implementation, the same control structure was interfaced with ROS2 Humble, and the optimization problems were solved in Python using the SLSQP method available in

`scipy.optimize`. For the scaled-vehicle tests, the AgileX LIMO platform ran a ROS1 interface and received the control commands computed on the external PC.

The physical and controller parameters used in the dynamic bicycle simulations are reported in the corresponding chapters, including the vehicle parameters in Table 6.1 and the viability-related bounds in the sections dedicated to the HOCBF and flatness-based formulations.

8.2 Viability Algorithm

This appendix reports the complete pseudocode of the discrete-time viability procedure. The algorithm computes, at each control cycle, the admissible longitudinal acceleration interval ensuring that the next state remains viable and that future constraint satisfaction is guaranteed under bounded acceleration.

All environment-dependent constraints (e.g., stop lines, obstacle clearance, intersection priority regions) are embedded into the time-varying admissible position interval

$$X \in [X_{\min}, X_{\max}],$$

so that the viability computation is performed on the one-dimensional longitudinal subsystem.

The first routine computes the acceleration bounds induced by the discrete-time viability inequalities. These bounds ensure that, from the predicted state at $t = \Delta t$, it remains possible to brake within the admissible acceleration limits without violating the longitudinal position constraints.

Algorithm 1 Acceleration Bounds from Longitudinal Viability

Require: $X, v_x, X_{\min}, X_{\max}, a_{\max}, \Delta t$
Ensure: $[a_{x,m}^{\text{viab}}, a_{x,M}^{\text{viab}}]$

$a \leftarrow \Delta t^2$
$a_1 \leftarrow -\dfrac{v_x}{\Delta t}$
▷ Upper viability inequality w.r.t. $X_{\max}$
$b \leftarrow \Delta t(2v_x + a_{\max}\Delta t)$
$c \leftarrow v_x^2 - 2a_{\max}(X_{\max} - X - \Delta t v_x)$
$\Delta \leftarrow b^2 - 4ac$
if $\Delta \geq 0$ **then**
    $a_3 \leftarrow \dfrac{-b + \sqrt{\Delta}}{2a}$
    $a_{x,M}^{\text{viab}} \leftarrow \max(a_1, a_3)$
else
    $a_{x,M}^{\text{viab}} \leftarrow a_1$
end if
▷ Lower viability inequality w.r.t. $X_{\min}$
$b \leftarrow 2\Delta t v_x - a_{\max}\Delta t^2$
$c \leftarrow v_x^2 - 2a_{\max}(X + \Delta t v_x - X_{\min})$
$\Delta \leftarrow b^2 - 4ac$
if $\Delta \geq 0$ **then**
    $a_2 \leftarrow \dfrac{-b - \sqrt{\Delta}}{2a}$
    $a_{x,m}^{\text{viab}} \leftarrow \min(a_1, a_2)$
else
    $a_{x,m}^{\text{viab}} \leftarrow a_1$
end if
return $[a_{x,m}^{\text{viab}}, a_{x,M}^{\text{viab}}]$

The interval $[a_{x,m}^{\text{viab}}, a_{x,M}^{\text{viab}}]$ represents the candidate acceleration bounds obtained from future braking feasibility only. These bounds do not yet account for direct velocity or acceleration limits.

The second routine intersects the viability bounds with the physical acceleration limits and with the velocity feasibility constraints derived from the discrete-time propagation model

$$v_x(\Delta t) = v_x + a_x \Delta t.$$

Each constraint produces a candidate upper and lower bound, and the final admissible interval is obtained by selecting the most restrictive ones.

Algorithm 2 Intersection with Physical Acceleration and Velocity Limits

Require: $a_{x,m}^{\text{viab}}, a_{x,M}^{\text{viab}}, a_{\max}, v_{x,\min}, v_{x,\max}, v_x, \Delta t$
Ensure: $[a_{x,m}^V, a_{x,M}^V]$
1: $LB \leftarrow \{a_{x,m}^{\text{viab}}, -a_{\max}, (v_{x,\min} - v_x)/\Delta t\}$
2: $UB \leftarrow \{a_{x,M}^{\text{viab}}, +a_{\max}, (v_{x,\max} - v_x)/\Delta t\}$
3: $a_{x,m}^V \leftarrow \max(LB)$
4: $a_{x,M}^V \leftarrow \min(UB)$
5: **return** $[a_{x,m}^V, a_{x,M}^V]$

The resulting interval $[a_{x,m}^V, a_{x,M}^V]$ defines the set of accelerations that simultaneously satisfy:

- physical acceleration limits,
- velocity feasibility at the next time step,
- future braking feasibility with respect to longitudinal position constraints.

The complete viability procedure, executed at each control cycle, is reported below. It sequentially evaluates the viability-induced bounds and intersects them with the physical limits to obtain the final admissible acceleration interval.

Algorithm 3 Computation of the Viable Longitudinal Acceleration Interval

Require: $X, v_x, X_{\min}, X_{\max}, v_{x,\min}, v_{x,\max}, a_{\max}, \Delta t$
Ensure: $[a_{x,m}^V, a_{x,M}^V]$, feasible flag
1: $[a_{x,m}^{\text{viab}}, a_{x,M}^{\text{viab}}]$
2: $\quad \leftarrow$ accBoundsFromViability$(X, v_x, X_{\min}, X_{\max}, a_{\max}, \Delta t)$
3: $[a_{x,m}^V, a_{x,M}^V]$
4: $\quad \leftarrow$ combineBounds$(a_{x,m}^{\text{viab}}, a_{x,M}^{\text{viab}}, a_{\max}, v_{x,\min}, v_{x,\max}, v_x, \Delta t)$
5: feasible $\leftarrow (a_{x,m}^V < a_{x,M}^V)$
6: **return** $[a_{x,m}^V, a_{x,M}^V]$, feasible

If the resulting interval is empty, i.e.,

$$a_{x,m}^V \geq a_{x,M}^V,$$

the current state lies outside the viability region and no admissible acceleration exists that can prevent future constraint violation.

Conversely, when

$$a_{x,m}^V = a_{x,M}^V,$$

the system operates on the boundary of the viability region, meaning that only one admissible acceleration (typically maximum braking) can prevent violation of the constraints. In practice, this condition corresponds to an emergency scenario and may trigger a fallback behavior.

The procedure is executed at each control cycle. All environment-dependent constraints (e.g., stop lines, obstacle clearance, intersection priority regions) are embedded in the time-varying position interval $X \in [X_{\min}, X_{\max}]$, so that the viability computation remains one-dimensional and suitable for real-time implementation.

8.3 Appendix: Detailed Derivation of Flat-Output Derivatives

In this appendix we report the explicit expressions required for the reconstruction of the physical inputs from the flat outputs and their higher-order derivatives.

For compactness of notation, we introduce the following auxiliary terms:

$$\mathrm{N} = l_f m\, \dot{y}_1\, \ddot{y}_2 + K_r(l_f + l_r)\, \dot{y}_2,$$

$$\mathrm{D} = K_r(l_f + l_r)\big(I_{zz} - l_f l_r m\big) + \big(l_f m\, \dot{y}_1\big)^2.$$

Longitudinal flat-output second derivative The second derivative of the longitudinal flat coordinate is directly related to the longitudinal dynamics:

$$\ddot{y}_1 = \dot{v}_x = \frac{F_{xf}\cos\delta - F_{yf}\sin\delta + F_{xf} - F_{\mathrm{drag}} + m v_y \omega}{m}. \tag{8.1}$$

This equation provides the first nonlinear relation used for input reconstruction.

Third derivative of the lateral flat coordinate The third derivative of the lateral flat output is obtained by differentiating the flatness relations and substituting the yaw dynamics. After straightforward but lengthy algebraic manipulations, one obtains:

$$\begin{aligned} &\frac{2l_f^2 m^2 \dot{y}_1 \Big(l_f^2 m^2 \ddot{y}_2 \dot{y}_1 + K_r l_f m (l_f + l_r)\dot{y}_2\Big)\ddot{y}_1}{\Big(l_f^2 m^2 (\dot{y}_1)^2 + K_r(l_f + l_r)(I_{zz} - l_f l_r m)\Big)^2} \\ &\qquad - \frac{l_f^2 m^2\, \dddot{y}_2 \dot{y}_1 + l_f^2 m^2 \ddot{y}_2 \ddot{y}_1 + K_r l_f m (l_f + l_r)\ddot{y}_2}{l_f^2 m^2 (\dot{y}_1)^2 + K_r(l_f + l_r)(I_{zz} - l_f l_r m)} \\ &= \frac{l_f F_{yf}\cos\delta + l_f F_{xf}\sin\delta - l_r F_{yr}}{I_{zz}}. \end{aligned} \tag{8.2}$$

This equation explicitly shows that the physical inputs enter only at the third differen-

tiation level, confirming the relative degree three of the lateral flat coordinate.

Optional consistency relation For completeness, a consistency relation obtained from the differentiated flat-output kinematics can also be written as

$$\ddot{y}_2 - \frac{I_{zz}}{l_f m}\frac{\Big(l_f^2 m^2(\ddot{y}_1\ddot{y}_2 + \dot{y}_1\dddot{y}_2) + K_r(l_f + l_r)l_f m\ddot{y}_2\Big)\mathrm{D} - 2l_f^2 m^2\dot{y}_1\ddot{y}_1\mathrm{N}}{\mathrm{D}^2} = \frac{F_{xf}\sin\delta + F_{yf}\cos\delta + F_{yr} - mv_x\omega}{m}. \tag{8.3}$$

This relation is not strictly required for input reconstruction, which can be performed using the longitudinal equation and the third-derivative equation above, but can be employed as an additional consistency condition in a nonlinear least-squares formulation.

Although the fourth derivative of the flat outputs can be derived by straightforward but lengthy symbolic differentiation, it involves no additional conceptual steps beyond the algebraic substitutions already presented. Since it would only result in significantly less readable expressions without adding further theoretical insight, it is omitted for the sake of clarity.

Bibliography

[1] F. Bernabei and C. Secchi, "Smart infrastructure and autonomous vehicles: Ensuring safety and efficiency in urban traffic with control barrier functions," *Mechatronics*, 2024.

[2] F. Bernabei, C. Secchi, and G. Notomista, "Viable and computationally efficient safety for differentially flat dynamical systems with application to autonomous vehicles," *IEEE Robotics and Automation Letters*, 2026, submitted.

[3] F. Bernabei and C. Secchi, "A constraint-based control architecture for urban autonomous vehicles," in *International Conference on Methods and Models in Automation and Robotics*, 2024, young Author Best Paper Award.

[4] ——, "A constraint-driven control system for autonomous vehicles navigating in urban environments," in *European Control Conference (ECC)*, 2026, submitted.

[5] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[6] S. Fallah, M. Dianati, A. Stevens, D. Oxtoby, U. Montanaro, S. Dixit, and A. Mouzakitis, "Towards connected autonomous driving: Review of use-cases," *Vehicle System Dynamics*, vol. 57, no. 6, pp. 779–814, 2019.

[7] B. Paden, M. Čap, 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.

[8] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[9] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2020.

[10] D. Gruyer, S. Glaser, and M. Revilloud, "Motion planning and control for autonomous vehicles: A review," *Annual Reviews in Control*, vol. 47, pp. 1–15, 2019.

[11] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari, "Proud - public road urban driverless test: Architecture and results," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2014, pp. 648–654.

[12] M. Wang, L. Zhang, Z. Wang, Y. Sai, and Y. Chu, "A real-time dynamic trajectory planning for autonomous driving vehicles," in *2019 3rd Conference on Vehicle Control and Intelligence (CVCI)*. IEEE, 2019, pp. 1–6.

[13] D. Coelho and M. Oliveira, "A review of end-to-end autonomous driving in urban environments," *IEEE Access*, vol. 10, pp. 75 296–75 311, 2022.

[14] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[15] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[16] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *IEEE Conference on Decision and Control (CDC)*, 2019, pp. 474–479.

[17] J.-P. Aubin, *Viability Theory*, 2nd ed. Berlin, Heidelberg: Springer, 2011.

[18] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[19] M. Nagumo, "Über die lage der integralkurven gewöhnlicher differentialgleichungen," *Proceedings of the Physico-Mathematical Society of Japan*, vol. 24, pp. 551–559, 1942.

[20] J.-P. Aubin, *Viability Theory*. Birkhäuser, 1991.

[21] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[22] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*, 2007, pp. 477–492.

[23] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," in *IFAC World Congress*, 2007, pp. 462–467.

[24] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," *IEEE Control Systems Magazine*, vol. 37, no. 2, pp. 28–53, 2017.

[25] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," in *IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 2015, pp. 54–61.

[26] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *American Control Conference*, 2016, pp. 322–328.

[27] J.-P. Aubin and A. Cellina, *Differential Inclusions*. Berlin, Heidelberg: Springer, 1984.

[28] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton–jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[29] A. Bouguerra, D. Abel, N. Perrin, and P. Mullhaupt, "Viability-based safety for nonlinear control systems," *IEEE Control Systems Letters*, vol. 3, no. 2, pp. 324–329, 2019.

[30] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," *International Journal of Robust and Nonlinear Control*, vol. 5, no. 5, pp. 345–364, 1995.

[31] A. A. Nguyen, F. Jabbari, and M. Egerstedt, "Mutualistic interactions in heterogeneous multi-agent systems," in *Proceedings of the 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 411–418.

[32] A. A. Nguyen, L. Guerrero-Bonilla, F. Jabbari, and M. Egerstedt, "Scalable, pairwise collaborations in heterogeneous multi-robot teams," *IEEE Control Systems Letters*, vol. 8, pp. 604–609, 2024.

[33] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conference (ECC)*, 2019, pp. 3420–3431.

[34] F. Ferraguti, C. T. Landi, A. Singletary, H.-C. Lin, A. Ames, C. Secchi, and M. Bonfè, "Safety and efficiency in robotics: The control barrier functions approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 139–151, 2022.

[35] 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.

[36] C. Yu, V. Cherfaoui, and P. Bonnifait, "Semantic evidential lane grids with prior maps for autonomous navigation," in *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1875–1881.

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

[38] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed. Springer, 2012.

[39] G. Genta, *Motor Vehicle Dynamics: Modeling and Simulation*. World Scientific, 2009.

[40] H. B. Pacejka, *Tire and Vehicle Dynamics*, 3rd ed. Butterworth-Heinemann, 2012.

[41] A. D. Prete, "Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 281–288, 2018.

[42] T. Okuyama, T. Yoshida, Y. Ikeda, and T. Noda, "Deep reinforcement learning for autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1–6.

[43] H.-P. Schöner, ""how good is good enough?" in autonomous driving," in *Electronic Components and Systems for Automotive Applications*, J. Langheim, Ed. Springer, 2019, pp. 119–142.

[44] J. Lévine, *Analysis and Control of Nonlinear Systems: A Flatness-based Approach*. Springer, 2009.

[45] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE ICRA*, 2011.

[46] J. Seo, J. Lee, E. Baek, R. Horowitz, and J. Choi, "Safety-critical control with nonaffine control inputs via a relaxed control barrier function for an autonomous vehicle," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 3, pp. 1036–1048, 2022.