FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

UP2Speed - Formal methods for collision avoidance of autonomous racing vehicles

Pedro Alexandre Mendes Lopes

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Luís Tiago Paiva

July 27, 2026

Abstract

Autonomous vehicles have attracted significant attention due to their potential to improve transportation efficiency and road safety. However, guaranteeing safe operation in dynamic environments remains a major challenge, requiring vehicles to avoid collisions while maintaining satisfactory trajectory tracking performance.

This dissertation investigates safety critical control strategies for autonomous racing vehicles based on Control Barrier Functions (CBFs). A control framework combining CBFs, Control Lyapunov Functions (CLFs), and Quadratic Programming (QP) is developed to simultaneously guarantee safety and trajectory tracking. In addition, High Order Control Barrier Functions (HOCBFs) and Integral High Order Control Barrier Functions (iHOCBFs) are investigated to address systems with higher relative degree.

Furthermore, machine learning techniques are employed to automatically optimize the controller parameters using simulation data, reducing the need for manual tuning. The proposed methodology is subsequently extended to multi vehicle scenarios through a formulation that explicitly accounts for the relative velocity of neighbouring vehicles.

Finally, the developed controller is experimentally validated using a small scale autonomous vehicle platform equipped with computer vision based state estimation and wireless communication. The obtained results demonstrate the effectiveness of the proposed approach in achieving safe trajectory tracking and obstacle avoidance under both simulated and real world conditions.

Keywords: Autonomous Vehicles, Control Barrier Functions (CBFs), Safety Critical Control, Obstacle Avoidance, Machine Learning, Multi Vehicle Systems.

Resumo

Os veículos autónomos têm despertado um interesse crescente devido ao seu potencial para melhorar a eficiência dos sistemas de transporte e aumentar a segurança rodoviária. No entanto, garantir uma operação segura em ambientes dinâmicos continua a constituir um desafio significativo, exigindo que os veículos evitem colisões enquanto mantêm um desempenho adequado no seguimento de trajetórias.

Nesta dissertação são investigadas estratégias de controlo crítico para a segurança de veículos autónomos de competição, baseadas em Control Barrier Functions (CBFs). É desenvolvido um sistema de controlo que combina CBFs, Control Lyapunov Functions (CLFs) e Quadratic Programming (QP), permitindo garantir simultaneamente a segurança e o seguimento de trajetória. São ainda exploradas formulações de High Order Control Barrier Functions (HOCBFs) e Integral High Order Control Barrier Functions (iHOCBFs), de forma a lidar com sistemas que apresentam grau relativo superior.

Adicionalmente, são aplicadas técnicas de aprendizagem automática para otimizar automaticamente os parâmetros do controlador com base em dados obtidos em simulação, reduzindo a necessidade de ajuste manual. A metodologia proposta é posteriormente estendida a cenários com múltiplos veículos através de uma formulação que considera explicitamente a velocidade relativa dos veículos vizinhos.

Por fim, o controlador desenvolvido é validado experimentalmente numa plataforma de veículo autónomo em pequena escala, equipada com estimação de estado baseada em visão por computador e comunicação sem fios. Os resultados obtidos demonstram a eficácia da abordagem proposta na realização de seguimento de trajetória e desvio de obstáculos de forma segura, tanto em ambiente de simulação como em condições reais.

Palavras-chave: Veículos Autónomos, Control Barrier Functions (CBFs), Controlo Crítico para a Segurança, Desvio de Obstáculos, Aprendizagem Automática, Sistemas Multi-Veículo.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals[1] that address major global challenges such as health, safety, sustainability, innovation, and economic development.

Figure 1: United Nations Sustainable Development Goals [1].

Autonomous driving technologies have the potential to significantly improve transportation systems by reducing the number of traffic accidents, increasing mobility efficiency, and enabling the development of safer and smarter cities. However, achieving these benefits requires reliable perception, planning, and control systems capable of operating safely in dynamic environments.

This dissertation contributes to this objective through the development of safety-critical control strategies for autonomous vehicles based on Control Barrier Functions, machine learning-assisted parameter optimization, and experimental validation on a real vehicle platform. By improving the safety, reliability, and robustness of autonomous navigation systems, this work supports several of the United Nations Sustainable Development Goals, particularly those related to health and safety, technological innovation, and sustainable transportation.

The specific Sustainable Development Goals addressed in this work are:

SDG 3 Ensure healthy lives and promote well-being for all at all ages.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.

SDG 11 Make cities and human settlements inclusive, safe, resilient, and sustainable.

[1] https://sdgs.un.org/goals

SDG	Target	Contribution	Performance Indicators and Metrics
3	3.6	Development of safety-critical control algorithms capable of reducing the likelihood of vehicle collisions through formal safety guarantees and real-time obstacle avoidance.	Collision-free operation, successful obstacle avoidance, minimum safety distance maintained, trajectory tracking performance.
9	9.5	Advancement of autonomous vehicle technologies through the development of novel CBF-based control architectures, machine learning-assisted parameter optimization, and experimental validation on a real platform.	Successful implementation and validation of autonomous driving technologies, controller performance improvement, experimental verification of safety-critical systems.
11	11.2	Contribution to future intelligent transportation systems by enabling safe autonomous navigation in environments containing static and dynamic obstacles, including neighbouring vehicles.	Reliable autonomous navigation, safe vehicle interactions, obstacle avoidance success rate, robustness in multi-vehicle scenarios.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Luís Tiago Paiva, for his guidance, support, and valuable insights throughout the development of this dissertation. His knowledge and feedback were fundamental to the completion of this work.

I would also like to thank everyone working at the UP2Speed laboratory for providing an inspiring environment in which to learn and develop this research. I am especially grateful to my colleagues from the 5DPO robotics team, with whom I had the privilege of sharing some of the most rewarding moments of my university years. Throughout this journey, they became much more than teammates. In particular, I would like to thank Afonso Mateus and Daniel Silva for their friendship, collaboration, countless technical discussions, and all the challenges and achievements we experienced together.

Finally, I would like to thank my family, especially my mother, for her unconditional love, support, encouragement, and sacrifices throughout my academic journey. Without her, this achievement would not have been possible.

To everyone who contributed, directly or indirectly, to this work, I extend my heartfelt gratitude.

Pedro Lopes

Contents

1 Introduction **1**
- 1.1 Background and Motivation . . . 1
- 1.2 Problem Statement . . . 1
- 1.3 Research Objectives and Contributions . . . 2
- 1.4 Dissertation Organization . . . 2

2 Context **5**
- 2.1 Autonomous racing vehicles . . . 5
  - 2.1.1 Perception . . . 6
  - 2.1.2 Planning . . . 7
  - 2.1.3 Control . . . 8
- 2.2 UP2Speed Setup . . . 9
  - 2.2.1 Vehicle Hardware . . . 9
  - 2.2.2 Track Infrastructure . . . 10
  - 2.2.3 Software Architecture . . . 11

3 State of the Art **13**
- 3.1 Model Predictive Control . . . 13
- 3.2 Artificial Potential Fields . . . 14
- 3.3 Control Barrier Functions . . . 16
  - 3.3.1 Barrier Functions . . . 16
  - 3.3.2 Control Barrier Functions . . . 18
- 3.4 Quadratic Programming . . . 19
- 3.5 Control Lyapunov Functions . . . 20
- 3.6 High-Order Control Barrier Functions . . . 21
- 3.7 Integral High Order Control Barrier Functions . . . 23

4 Single-Vehicle Safety Control **27**
- 4.1 CBF with an External Controller . . . 28
  - 4.1.1 Formulation . . . 28
  - 4.1.2 Simulation Results and Discussion . . . 32
- 4.2 Lookahead-Based CBF . . . 35
  - 4.2.1 Formulation . . . 36
  - 4.2.2 Simulation Results and Discussion . . . 37
- 4.3 CLF-CBF-QP Controller . . . 42
  - 4.3.1 Formulation . . . 42
  - 4.3.2 Simulation Results and Discussion . . . 45
- 4.4 Chapter Conclusions . . . 48

5 Parameter Optimization Using Machine Learning **51**
5.1 Methodology . . . 51
5.2 Results and Discussion . . . 57
5.3 Chapter Conclusions . . . 62

6 Multi-Vehicle Safety Control **65**
6.1 Static Obstacle Formulation . . . 65
6.1.1 Formulation . . . 65
6.1.2 Simulation Results and Discussion . . . 66
6.2 Relative Velocity CBF Formulation . . . 66
6.2.1 Formulation . . . 67
6.2.2 Simulation Results and Discussion . . . 69
6.3 Chapter Conclusions . . . 70

7 Real Vehicle Implementation **73**
7.1 Experimental Methodology . . . 73
7.1.1 State Estimation . . . 73
7.1.2 Communication Framework . . . 74
7.1.3 Test Scenarios . . . 75
7.2 Experimental Results and Discussion . . . 75
7.2.1 Trajectory Tracking . . . 75
7.2.2 Obstacle Avoidance . . . 76
7.2.3 Computational Performance . . . 76
7.3 Sources of Experimental Error . . . 78
7.4 Chapter Conclusions . . . 79

8 Conclusions and Future Work **81**

References **83**

List of Figures

1 United Nations Sustainable Development Goals [1]. . . . v

2.1 The different levels of automation according to Society of Automotive Engineers (SAE) [11]. . . . 6
2.2 A typical autonomous vehicle system [12]. . . . 7
2.3 The different planning approaches: (a) Grid based; (b) Sampling based; (c) Potential field. [14] . . . 8
2.4 Experimental vehicle platform. . . . 9
2.5 Electronic components used in the vehicle. . . . 10
2.6 Vehicle before and after the hardware modifications. . . . 10
2.7 Experimental track used for the real world validation. . . . 11
2.8 Overall architecture of the experimental platform. . . . 12

3.1 Illustration of model predictive control [18] . . . 13
3.2 Illustration of artificial potential fields [23]. . . . 15
3.3 Simulation showing a passage blocked [24]. . . . 16
3.4 Illustration of the modification to the classical Artificial Potential Fields (APF) [24]. 16
3.5 Simulation of the solution for the blocked passage [24]. . . . 16
3.6 Illustration of the ACC safety constraint [26]. . . . 18

4.1 Kinematic bicycle model. . . . 27
4.2 Safety distance definition. . . . 29
4.3 Simulation track similar to the laboratory track. . . . 33
4.4 Comparison between the nominal controller and the Quadratic Programming (QP) solution. . . . 33
4.5 Steering angle comparison. . . . 33
4.6 Vehicle trajectory using the integral High Order Control Barrier Function (iHOCBF) formulation. . . . 34
4.7 Comparison between the nominal controller and the iHOCBF-QP solution. . . . 34
4.8 Lookahead point concept [29]. . . . 36
4.9 Vehicle trajectory using the lookahead-based Control Barrier Function (CBF). . . . 38
4.10 Influence of the class $\mathscr{K}$ parameter α on obstacle avoidance behaviour. . . . 39
4.11 Influence of the lookahead distance on obstacle avoidance. . . . 40
4.12 Limitations associated with excessively small and excessively large lookahead distances. . . . 40
4.13 Influence of the weighting matrix parameters on the obstacle avoidance behaviour. 41
4.14 Vehicle behaviour considering only the barrier constraints associated with the track boundaries. . . . 42
4.15 Trajectory tracking error definition [29]. . . . 43

4.16 Influence of the lookahead distance on trajectory tracking. 45
4.17 Influence of the Control Lyapunov Function (CLF) convergence parameter ε. . . 46
4.18 Influence of the slack-variable weight p_{slack}. 47
4.19 Influence of the trajectory-tracking weights. 48

5.1 Illustration of Support Vector Machine (SVM) classification in the parameter space [33]. 53
5.2 Feasible and infeasible parameter samples collected from simulations. 55
5.3 Learned SVM decision function $H(p,q)$ and the associated feasibility boundary. . 56
5.4 Gaussian Radial Basis Function (GRBF) interpolation obtained from the sum of Gaussian basis functions [36]. 56
5.5 Estimated score surface obtained through GRBF interpolation. The star denotes the maximum simulated score, while the cross indicates the optimum predicted by the optimization framework. 58
5.6 Comparison between the optimized parameter set and the default parameter set. . 59
5.7 Refined score surface obtained through local sampling around the predicted optimum. 60
5.8 Vehicle becoming trapped when using parameters learned for a different obstacle configuration. 61
5.9 Trajectory obtained using the parameters learned for the new obstacle configuration. 61
5.10 Trajectory obtained using the reference angular velocity as the nominal angular velocity in the QP cost function. 62

6.1 Evolution of the vehicle trajectories during the simulation using the static obstacle formulation. 67
6.2 Evolution of the vehicle trajectories during the simulation using the relative velocity formulation. 69
6.3 Examples of multi-vehicle interactions using the proposed CBF-based controller. 70

7.1 HSV calibration interface used for colour segmentation tuning. 73
7.2 Vehicle orientation estimation based on consecutive position measurements. . . . 74
7.3 Trajectory tracking experiment using the CLF-CBF-QP controller. 75
7.4 Real world obstacle avoidance experiments using the proposed CBF-based controller. 77
7.5 Vehicle stopped due to simultaneous activation of track boundary safety constraints. 78

List of Tables

7.1 Average execution times of the experimental platform. 77

List of Acronyms

ACC Adaptive Cruise Control
APF Artificial Potential Fields
BLE Bluetooth Low Energy
CBF Control Barrier Function
CLF Control Lyapunov Function
FEUP Faculdade de Engenharia da Universidade do Porto
GRBF Gaussian Radial Basis Function
HOCBF High Order Control Barrier Function
iHOCBF integral High Order Control Barrier Function
IMU Inertial Measurement Unit
LP Linear Program
ML Machine Learning
MPC Model Predictive Control
PID Proportional Integral Derivative
PRM Probabilistic Roadmap
QP Quadratic Programming
RRT Rapidly Exploring Random Tree
SAE Society of Automotive Engineers
SVM Support Vector Machine

Chapter 1

Introduction

1.1 Background and Motivation

Autonomous vehicles have become one of the most active research topics in recent decades due to their potential to transform transportation systems. By combining advances in sensing, computation, communication, and control technologies, autonomous vehicles improve traffic efficiency [2], reduce pollution [3], and enhance road safety [4], while freeing occupants from the driving task. Unlike human drivers, autonomous systems are not affected by fatigue, distraction, or stress, allowing them to react more consistently and rapidly to changes in the surrounding environment.

Despite the significant technological progress achieved in recent years, guaranteeing safety remains one of the main challenges in autonomous driving [5]. Autonomous vehicles must continuously operate in dynamic and uncertain environments while avoiding collisions with obstacles, other vehicles, and road boundaries. Consequently, the development of control strategies capable of enforcing safety constraints without compromising vehicle performance has become a critical research area.

Motorsport has historically played an important role in the development of automotive technologies that were later transferred to commercial vehicles. Similarly, autonomous racing platforms provide valuable testbeds for the development and validation of autonomous driving technologies, since they require vehicles to operate close to their performance limits while maintaining safe operation [6]. In this context, this dissertation is developed within the UP2Speed project at the Faculty of Engineering of the University of Porto (Faculdade de Engenharia da Universidade do Porto (FEUP)), which focuses on the development of autonomous racing vehicles capable of navigating a track while avoiding collisions with obstacles and neighbouring vehicles.

1.2 Problem Statement

Among the various approaches proposed in the literature to address safety critical control problems, Control Barrier Functions (CBFs) have attracted considerable attention due to their ability to

formally guarantee safety by enforcing the forward invariance of predefined safe sets. Unlike trajectory planning or tracking methods, CBFs focus exclusively on safety and can be integrated with existing controllers through optimization based frameworks [7], [8], [9]. In particular, the combination of CBFs with Control Lyapunov Functions (CLFs) within a Quadratic Programming (QP) formulation enables the simultaneous achievement of safety and trajectory tracking objectives.

1.3 Research Objectives and Contributions

The main objective of this dissertation is to investigate, develop, and validate safety critical control strategies based on CBFs for autonomous racing vehicles. Initially, different barrier function formulations are studied and applied to obstacle avoidance problems. Particular attention is given to High Order Control Barrier Functions (HOCBFs) and Integral High Order Control Barrier Functions (iHOCBFs), which allow safety constraints to be enforced in systems where the control inputs do not appear directly in the first derivative of the barrier function [10]. Furthermore, a lookahead based formulation is explored to improve obstacle avoidance behaviour while maintaining trajectory tracking performance.

In addition to controller design, this work also investigates the automatic tuning of controller parameters through machine learning techniques. Simulation data are used to identify feasible regions of the parameter space and to estimate controller performance, allowing the optimization process to be performed systematically instead of relying solely on manual tuning. This approach aims to improve vehicle performance while preserving the safety guarantees provided by the control framework.

The proposed methodology is further extended to multi vehicle scenarios, where neighbouring vehicles are treated as dynamic obstacles. A formulation based on relative velocity information is introduced to explicitly account for obstacle motion, reducing conservativeness and improving the quality of the resulting vehicle interactions.

Finally, the developed control framework is validated experimentally using a small scale autonomous vehicle platform equipped with computer vision based state estimation, wireless communication, and onboard actuation. The experimental results demonstrate the applicability of the proposed methods in real world conditions and provide insight into the practical challenges associated with perception delays, communication latencies, actuator limitations, and modelling inaccuracies.

1.4 Dissertation Organization

This dissertation is organized as follows. Chapter 2 provides the necessary background on autonomous driving and describes the UP2Speed experimental platform used throughout this work. Chapter 3 reviews the state of the art in safety critical control, including Model Predictive Control (MPC), Artificial Potential Fields (APF), CBFs, CLFs, and HOCBFs. Chapter 4 presents the proposed single vehicle safety control framework and its application to obstacle avoidance

and trajectory tracking. Chapter 5 introduces the machine learning methodology used for controller parameter optimization and discusses the obtained results. Chapter 6 extends the proposed framework to multi vehicle scenarios by incorporating relative velocity information into the safety formulation. Chapter 7 describes the real vehicle implementation and presents the experimental validation. Finally, Chapter 8 summarizes the main conclusions and outlines directions for future work.

Chapter 2

Context

In this chapter, the vision of what is considered autonomous cars and the different levels of autonomous driving will be presented, followed by the three main categories present in this type of system, namely perception, planning, and control. These topics will be developed in each of the following subsections:

- Subsection 2.1.1 presents what perception is, its importance, and the ways in which it can be performed.
- Subsection 2.1.2 presents what planning is, its purpose, the different ways to perform it, and its advantages and disadvantages.
- Subsection 2.1.3 presents what control is, how it acts on the system, and the most classical way of performing it.

2.1 Autonomous racing vehicles

Autonomous driving has become increasingly present on the streets, drawing growing attention from everyone, especially researchers and major automotive companies. The prospect of multiple autonomous vehicles to which people entrust their lives also increases the demand for their safety and performance.

However, firstly it is important to clarify what is meant by autonomous driving. According to the Society of Automotive Engineers (SAE), there are five levels of autonomous driving, as shown in Figure 2.1 [11]. At Level 0 are conventional vehicles, fully controlled by humans. At Levels 1 and 2, driver assistance systems begin to appear, such as cruise control and lane detection with slight steering corrections, but they still require constant driver participation and attention. At Level 3, the vehicle gains some independence, incorporating features such as the so called chauffeur system and being able to perform overtaking; however, the driver must always be ready to take control. At Level 4, the vehicle is fully autonomous in certain areas or controlled environments, eliminating the need for a driver, although human intervention remains possible if necessary or

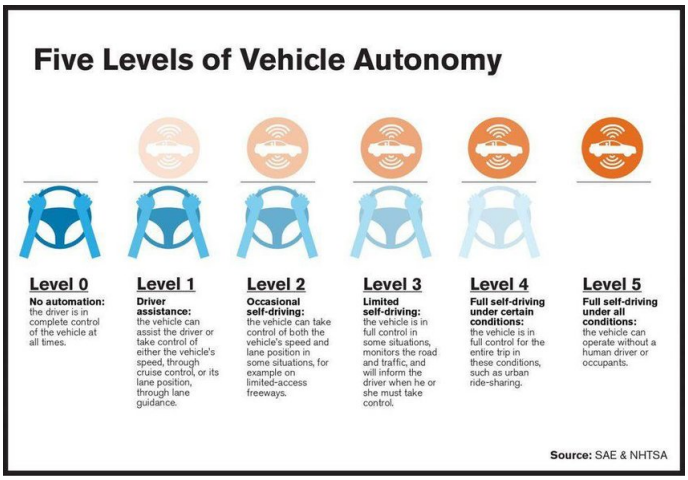


Figure 2.1: The different levels of automation according to SAE [11].

if the vehicle leaves the permitted zone. Finally, at Level 5, the highest degree of autonomous driving is reached, in which the vehicle requires no human intervention in any situation, and this is the scenario considered from this point onward.

In addition, the focus will not be on autonomous vehicles in general, but rather on autonomous racing cars, since they constitute excellent testbeds for the development of autonomous driving technologies that can later be applied to civilian vehicles. This is due to the fact that they allow one of the main factors involved to be pushed to the limit, namely speed.

With this in mind, and as previously mentioned, this work is developed as part of a broader project carried out at FEUP, named UP2Speed, which focuses on the control of autonomous racing vehicles.

This project integrates the three fundamental categories of an autonomous vehicle system, namely perception, planning, and control [6], [12], [13], as shown in Figure 2.2. Perception corresponds to the process of extracting information from the environment in which the vehicle is operating, allowing it to localize itself and identify obstacles. Planning determines the path that the vehicle should follow to move from point A to point B, using the information provided by perception to, for example, avoid obstacles. Finally, the control layer corresponds to the low level component responsible for tracking the references generated by the planning stage. It computes the actuator commands required to follow the desired trajectory while maintaining stability and satisfying vehicle constraints.

2.1.1 Perception

Perception is a fundamental component in this type of system, as it acts as the vehicle's "eyes." Without it, it would not be possible to correctly determine the vehicle's position, since the system would be entirely dependent on odometry, which is subject to various errors. In addition, it would not be possible to obtain information about static obstacles and, more importantly, about dynamic

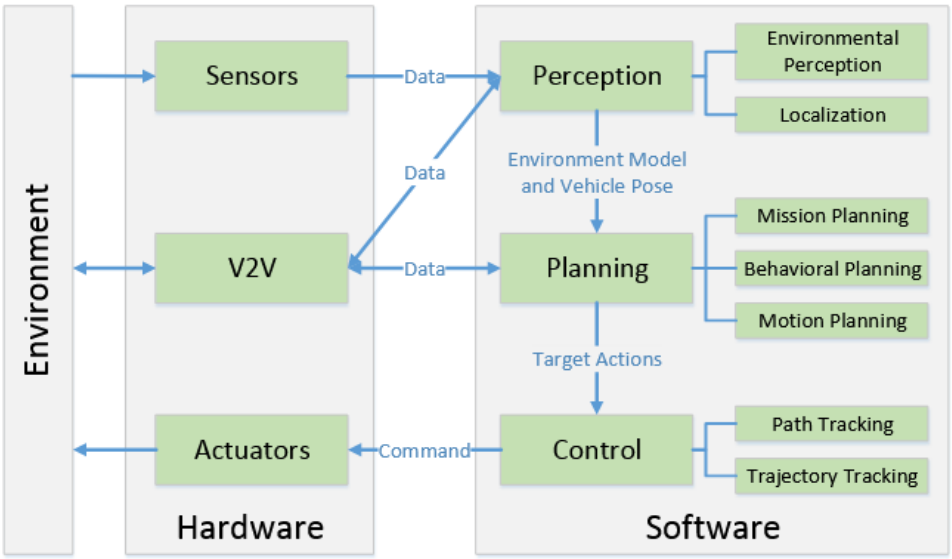


Figure 2.2: A typical autonomous vehicle system [12].

obstacles, which would prevent the vehicle from reacting to unpredictable situations such as children crossing the road.

To achieve this, perception can be performed in several ways, either on-board, where the sensors are mounted directly on the vehicle, using devices such as LiDARs, radars, and ultrasonic sensors, or off-board, where the sensing system is external to the vehicle, for example using cameras that observe the environment from outside. Although they operate in different ways, these devices allow the surrounding environment to be mapped and relevant elements to be identified, assisting the vehicle's navigation.

In the UP2Speed project, perception is performed off-board using an external overhead camera positioned above the track and pointing downward, capturing images at several frames per second as the vehicles move. After proper processing and the application of vehicle and track boundary detection techniques, the images allow information such as the position and velocity of each vehicle to be obtained.

2.1.2 Planning

Next comes planning, which tells the vehicle the path it should follow in order to reach its goal from its current position, taking into account several important factors such as path length, computational cost, smoothness, energy cost, and one of the main focuses discussed here, safety. These factors can become real challenges in dynamic environments, such as when pedestrians are crossing the street.

Planning algorithms can be divided into three types of approaches, as illustrated in Figure 2.3: grid based approaches, sampling based approaches, and potential field based approaches [14].

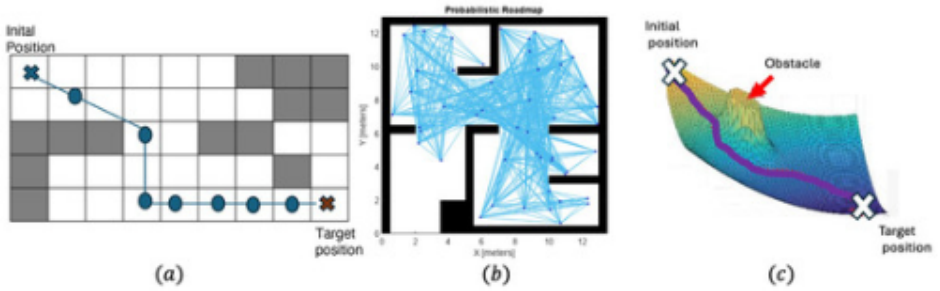


Figure 2.3: The different planning approaches: (a) Grid based; (b) Sampling based; (c) Potential field. [14]

The first type divides the environment into a grid, creating multiple cells, each of which is assigned a cost. The algorithm then selects the cells that should be traversed in order to reach the desired point, aiming to find the optimal path. Some examples are algorithms such as Dijkstra and A*, however this type of algorithm has an exponential computational cost that grows with the size and dimension of the space, making it less suitable for real time responses.

The second type works by randomly sampling the map and connecting these points in order to create a graph of the path that must be followed, where two points are not connected if there is an obstacle between them. Algorithms such as Probabilistic Roadmap (PRM) and Rapidly Exploring Random Tree (RRT) are examples of this type of planning, and they perform well in complex and high dimensional environments.

Finally, the last type transforms the map into a kind of force field in which the final goal has an attractive force while obstacles have a repulsive force, causing the vehicle to tend to move toward the goal and away from obstacles such as track boundaries and people. Compared with other techniques, this approach is computationally light and can handle dynamic obstacles and thus avoid them; however, it has some disadvantages, such as getting trapped in local minima and having difficulty dealing with narrow passages.

2.1.3 Control

It is at this stage that the planned motion is translated into actuator commands. Using the current vehicle state and the desired trajectory as inputs, the controller computes the steering, acceleration, and braking commands required to guide the vehicle along the planned path.

One of the simplest ways to do this is by using classical control [12], where feedback, that is, the output signal of the system, is used to correct the error with respect to the desired behavior, such as maintaining a certain speed. The most common way to achieve this is by using a Proportional

Integral Derivative (PID) controller, which adjusts the control signal based on the error, the integral of the error, and the derivative of the error, as shown in the following equation.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt}$$

However, this technique has some limitations: it has a delayed response to error, since it only acts after the deviation has already occurred; it is sensitive to noise; and it is susceptible to the windup phenomenon associated with the integral term, since when the actuator saturates, the integral continues to accumulate, causing overshoot.

2.2 UP2Speed Setup

Although the UP2Speed project involves other components, such as perception, the focus of this work is on the control component. Therefore, this section presents the experimental platform developed within the UP2Speed project at FEUP, which provides a controlled environment for repeatable real world testing. The following subsections describe the vehicle hardware, the experimental track, and the overall software architecture.

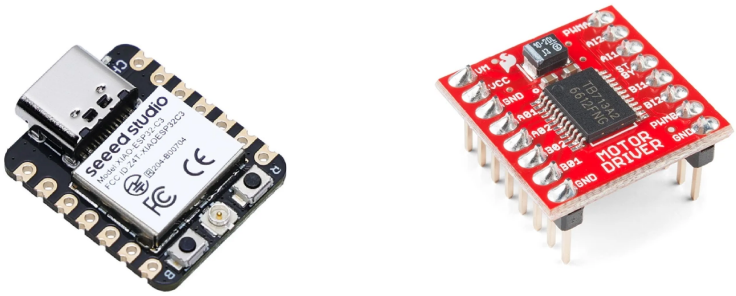
2.2.1 Vehicle Hardware

The experimental platform was developed using a small scale autonomous vehicle designed to validate the proposed safety critical control framework under real world conditions. The vehicle consists of a 1:43 scale chassis equipped with a rear wheel drive DC motor and a front steering servo, Fig. 2.4.

Figure 2.4: Experimental vehicle platform.

The onboard electronics were upgraded to an ESP32-C3 XIAO microcontroller, shown in Fig. 2.5a, which is responsible for receiving control commands and actuating the steering and

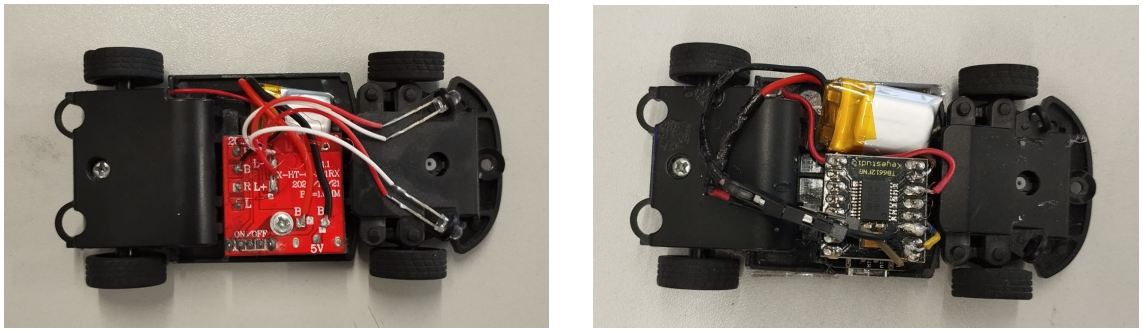
propulsion systems, and a TB6612FNG motor driver, shown in Fig. 2.5b, which enables bidirectional control of the DC motor while providing sufficient current capacity for vehicle operation.



(a) ESP32-C3 XIAO (b) TB6612FNG motor driver

Figure 2.5: Electronic components used in the vehicle.

This architecture allows computationally demanding tasks, such as image processing, state estimation, and optimization based control, to be executed on an external computer, while the vehicle performs only the low level actuation tasks.



(a) Original vehicle electronics (b) Modified vehicle electronics

Figure 2.6: Vehicle before and after the hardware modifications.

Figure 2.6 illustrate the hardware modifications introduced during the implementation process.

2.2.2 Track Infrastructure

The experimental validation was conducted on a custom built indoor track specifically designed for autonomous vehicle testing. The track boundaries were constructed using 3D printed barriers, allowing different layouts and obstacle configurations to be easily created and modified, Fig. 2.7.

A RunCam Pro 5 camera was mounted above the track in a top-down configuration, providing a global view of the experimental environment. The camera records video at resolutions of up to 4K (3840 × 2160) and supports frame rates of up to 60 fps, offering sufficient image quality and temporal resolution for reliable vehicle localization and orientation estimation. This setup enables the continuous acquisition of images used for vehicle localization and orientation estimation.

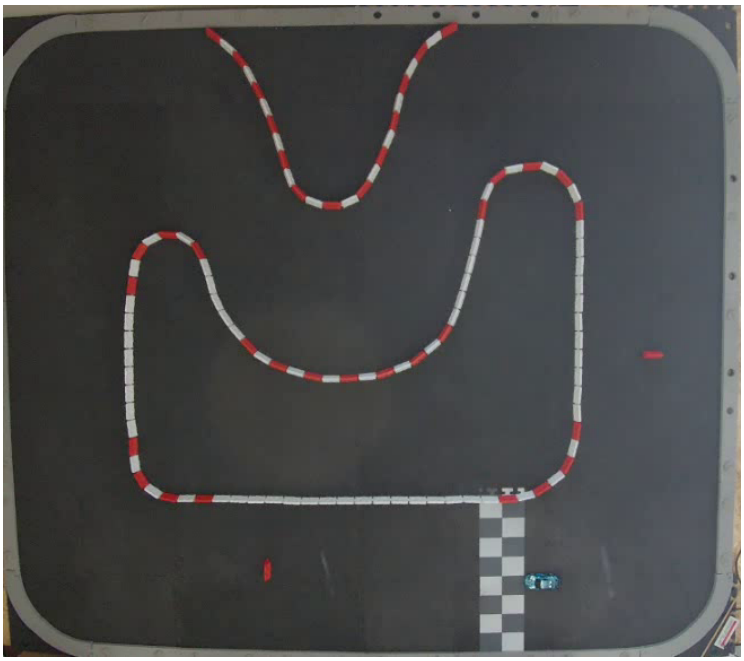


Figure 2.7: Experimental track used for the real world validation.

Static obstacles were placed at predefined locations along the track in order to evaluate the obstacle avoidance capabilities of the proposed controller. The resulting environment reproduces the main challenges addressed in simulation, including trajectory tracking, obstacle avoidance, and boundary avoidance, while maintaining a controlled and repeatable testing setup.

2.2.3 Software Architecture

The software architecture was designed to separate perception, control, and actuation into independent modules communicating through wireless interfaces.

The perception module receives images from the overhead camera and processes them using OpenCV based computer vision algorithms. Through colour segmentation and geometric processing, the position and orientation of the vehicle are estimated in real time. The estimated state is then transmitted to the controller through a UDP communication channel.

The control module executes the proposed CLF-CBF-QP framework using the estimated vehicle state. Based on the current vehicle position, the reference trajectory, and the surrounding obstacles, the controller computes the desired linear and angular velocity commands required to simultaneously satisfy trajectory tracking and safety objectives.

Finally, the computed commands are transmitted to the vehicle through Bluetooth Low Energy (BLE) communication. The ESP32-C3 microcontroller receives these commands and converts them into steering and propulsion signals that are applied to the actuators.

The overall architecture of the experimental platform is illustrated in Figure 2.8.

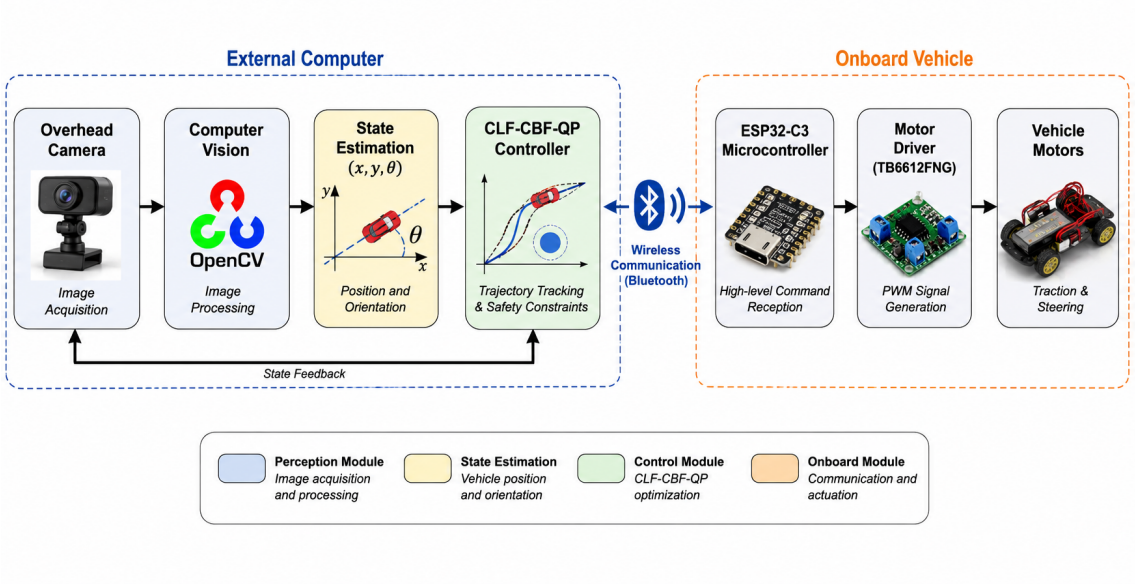


Figure 2.8: Overall architecture of the experimental platform.

Chapter 3

State of the Art

Since the focus of this work lies on the control component dedicated to obstacle avoidance, some existing techniques used for this purpose will be presented and analyzed here, including MPC, APF, and CBF.

3.1 Model Predictive Control

MPC is a feedback based control technique, shown in Figure 3.1, that iteratively solves an optimization problem and has the ability to impose constraints on the input, state, and output values of the system [15], [16], [17]. The fact that it provides valuable properties such as future prediction and the ability to enforce constraints, thereby guaranteeing system safety, makes it one of the most popular control techniques in autonomous vehicle control.

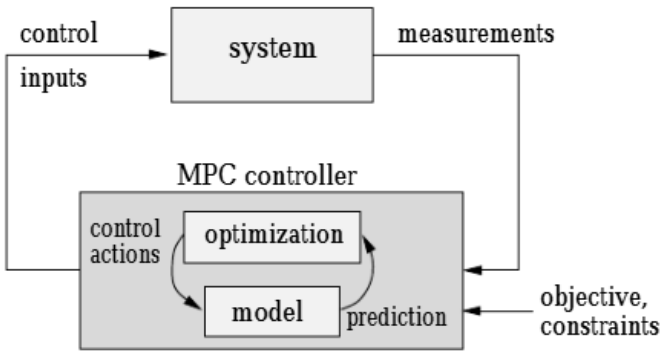


Figure 3.1: Illustration of model predictive control [18]

To apply this technique, one must first obtain the open loop model that describes the dynamics of the system being considered, in this case, a car. Then, the constraints that must be imposed are defined, such as the maximum acceleration or a minimum distance between the car and the track boundary, as well as a cost function, such as minimizing the error between the vehicle position

and the reference position. With this, the information required to implement the typical MPC controller algorithm is available [19], which can be summarized as follows :

1. First, the optimization problem based on the constraints and the cost function is solved only over a defined time window, for example for the next 10 time steps, which produces a sequence of control inputs for the system over that time horizon.

2. Next, only the control input corresponding to the first time step is applied.

3. Then, after applying the previous control command, a measurement or estimation of the system state is performed, which is why it is considered a feedback based controller.

4. Finally, the time window is shifted forward by one time step and, using the current value of the system state, the entire process is repeated, solving the optimization problem again, and so on.

Now addressing some questions that may have arisen regarding the algorithm presented above:

First: Why solve the optimization problem at every time step? In fact, the optimization problem only needs to be solved again if the measured system state differs from the predicted one. However, because the system model is not perfect and is affected by friction, modelling errors, measurement uncertainties, and changes in the environment, such discrepancies generally arise at every control interval. Therefore, the system state is measured or estimated and the optimization problem is solved again to compute a new optimal control sequence.

Second: Why use a time horizon? Optimization problems usually require a high computational cost, and in systems where real time response is important, it is desirable to reduce the duration of each control cycle as much as possible.

Third: Why not use a time horizon of only one time step? And how should its length be chosen? It would indeed be possible to use a time horizon of only one time step and thus reduce the time required to solve the optimization problem. However, this creates another issue, namely the inability to predict the future. For example, if the car needs to take a curve in the near future and this curve is not taken into account in advance, increasing the car's speed under the assumption that it is on a straight line may cause the car to be unable to make the turn and go off the track. Therefore, when choosing a time horizon, care must be taken to ensure that it is not too short, so that the future cannot be predicted, and not too long, so as not to significantly increase the optimization solving time. Moreover, predicting too far into the future may not be much more advantageous, especially in dynamic environments.

3.2 Artificial Potential Fields

Another method that is also widely used for trajectory planning and obstacle avoidance is the Artificial Potential Field, which was initially introduced by Khatib in 1985 [20]. Its popularity is due to the fact that it offers very valuable advantages, such as the ability to operate in real time,

perform obstacle avoidance, and generate smooth paths.

Its operating principle, as explained earlier, is based on the existence of a potential field that causes the vehicle to move away from obstacles and toward the final goal, thereby reaching its destination [21], [22], [23], as illustrated in Figure 3.2.

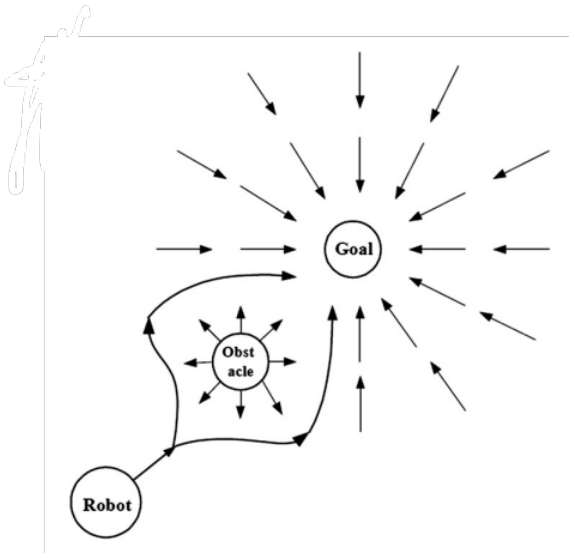


Figure 3.2: Illustration of artificial potential fields [23].

To achieve this, an attractive potential field is created around the final goal, U_A, and repulsive potential fields are created around the obstacles, U_R, with all of them being summed to generate the total potential field of the environment, U_T, as shown in the following equation:

$$U_{\mathrm{T}}(X) = U_A(X) + U_R(X)$$

By applying the gradient to these potential fields, the corresponding force fields are obtained:

$$F_A = -\nabla U_A(X),$$
$$F_R = -\nabla U_R(X),$$
$$F_T = F_A + F_R.$$

With this, by having the force vector, the vehicle can determine the direction it must follow as well as its speed, since if the magnitude of the force vector is small it means that the vehicle is close to an obstacle and therefore the speed should be lower. However, like any other control method, this one also has its imperfections, which can be summarized in the following points:

- When the final goal is close to an obstacle, the vehicle has difficulty reaching it.
- The formation of local minima at points where the attractive and repulsive forces are equal.
- Passages blocked by the proximity of two obstacles, as shown in Figure 3.3.

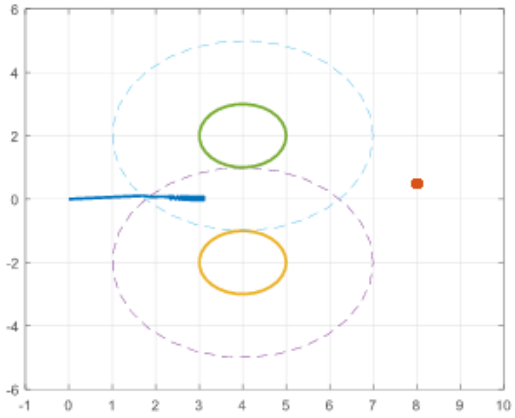


Figure 3.3: Simulation showing a passage blocked [24].

However, several methods have already been developed based on the classical APF method to overcome some of these problems, namely local minima and blocked passages, as presented in [24], where a tangential component is added to the repulsive force.

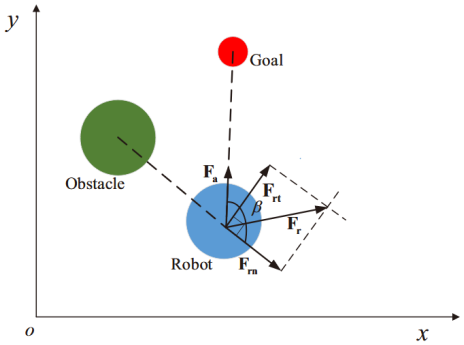


Figure 3.4: Illustration of the modification to the classical APF [24].

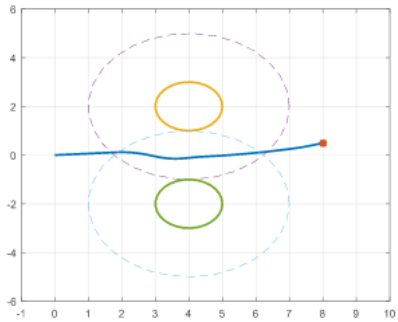


Figure 3.5: Simulation of the solution for the blocked passage [24].

3.3 Control Barrier Functions

This method, unlike those previously presented, has the distinctive feature of focusing exclusively on system safety. That is, CBFs do not include additional functionalities beyond ensuring safety, such as trajectory planning or tracking, unlike MPC, which solves a full optimization problem, as seen in Subsection 3.1.

3.3.1 Barrier Functions

As stated in [7], [8], [9], barrier functions are mathematical functions used to enforce safety constraints on a dynamical system by defining a safe set as

$$\mathscr{C} = \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}, \quad \text{where } h : \mathbb{R}^n \to \mathbb{R}.$$

Then, as originally established by Nagumo's theorem [25], if the system is initially safe, i.e., $h(\mathbf{x}) \geq 0$, and the following condition is satisfied,

$$\dot{h}(\mathbf{x}) \geq 0 \quad \text{whenever } h(\mathbf{x}) = 0,$$

for a system of the form

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

where $f : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^n$ is locally Lipschitz and $\mathbf{u} \in \mathbb{R}^q$ is the control input, then the system is guaranteed to remain inside the safe set for all future time, thereby ensuring safety.

However, this condition presents a discontinuity, since it is only required to hold on the boundary of the safe set, i.e., when $h(\mathbf{x}) = 0$ [10]. To illustrate this issue, consider an Adaptive Cruise Control (ACC) scenario involving two vehicles separated by a distance z,

$$z = x_p - x,$$

where $x \in \mathbb{R}$ denotes the position of the controlled vehicle and $x_p \in \mathbb{R}$ denotes the position of the leading vehicle. The system dynamics are defined as

$$\dot{z} = v_p - v, \quad u = \dot{v},$$

where v and v_p are the velocities of the controlled and leading vehicles, respectively, and $u \in \mathbb{R}$ is the control input acting on the acceleration of the controlled vehicle.

Assume a minimum safety distance d_0 such that $z \geq d_0$. Defining the barrier function as

$$h(\mathbf{x}) = z - d_0,$$

with state vector $\mathbf{x} = (z, v)$, the safety condition becomes

$$\dot{h}(\mathbf{x}) = \dot{z} = v_p - v \geq 0$$

whenever $h(\mathbf{x}) = 0$ (i.e., when $z = d_0$).

In other words, the velocity of the following vehicle cannot exceed that of the leading vehicle when the separation distance reaches d_0. However, when the distance is greater than d_0, the following vehicle could travel significantly faster than the leading vehicle. Consequently, as soon as the distance decreases to d_0, its velocity would need to be reduced instantaneously in order to satisfy the safety condition, which is physically impossible.

Furthermore, removing the condition $h(\mathbf{x}) = 0$ altogether would imply that $h(\mathbf{x})$ could never decrease. In the ACC example, this would mean that the distance between the vehicles could never decrease and would only be allowed to increase. As a consequence, every subset of the safe set would become invariant, which is generally an overly restrictive requirement.

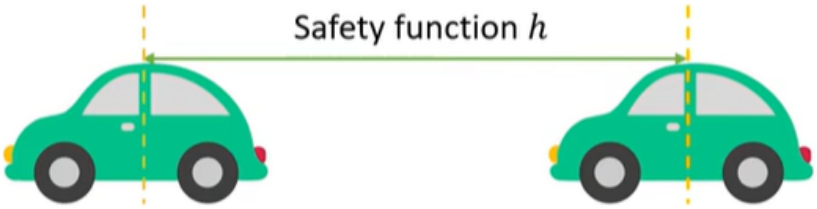


Figure 3.6: Illustration of the ACC safety constraint [26].

To address both the discontinuity and the invariance of all subsets of the safe set, the following relaxation can be introduced:

$$\dot{h}(\mathbf{x}) + \varepsilon h(\mathbf{x}) \geq 0,$$

or equivalently,

$$\dot{h}(\mathbf{x}) \geq -\varepsilon h(\mathbf{x}),$$

where $\varepsilon \geq 0$ determines how quickly the system is allowed to approach the boundary of the safe set. Under this formulation, $\dot{h}(\mathbf{x})$ may take negative values, allowing $h(\mathbf{x})$ to decrease and the system to move closer to the boundary. However, as the system approaches the boundary ($h(\mathbf{x}) = 0$), the maximum rate at which $h(\mathbf{x})$ can decrease becomes progressively smaller.

Returning to the ACC example, the condition becomes

$$v_p - v + \varepsilon(z - d_0) \geq 0.$$

This implies that the closer the following vehicle gets to the leading vehicle, the smaller the maximum admissible velocity difference becomes. Consequently, the following vehicle is forced to gradually reduce its speed as it approaches the safety boundary, resulting in a physically realizable and smoother behaviour.

3.3.2 Control Barrier Functions

Consider now the general class of control-affine systems given by

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{3.1}$$

where f and g are locally Lipschitz functions, $\mathbf{x} \in D \subset \mathbb{R}^n$, and $\mathbf{u} \in U \subset \mathbb{R}^m$ represents the set of admissible control inputs.

Let $h(\mathbf{x})$ be a barrier function associated with a given system constraint. Its time derivative is given by

$$\dot{h}(\mathbf{x}) = \frac{dh}{d\mathbf{x}}\dot{\mathbf{x}}.$$

Substituting $\dot{\mathbf{x}}$ from (3.1) yields

$$\dot{h}(\mathbf{x}) = \frac{dh}{d\mathbf{x}}\big(f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}\big),$$

which can be expanded as

$$\dot{h}(\mathbf{x}) = \frac{dh}{d\mathbf{x}}f(\mathbf{x}) + \frac{dh}{d\mathbf{x}}g(\mathbf{x})\,\mathbf{u}.$$

Using Lie derivative notation, this expression becomes

$$\dot{h}(\mathbf{x}) = L_f h(\mathbf{x}) + L_g h(\mathbf{x})\,\mathbf{u},$$

where

$$L_f h(\mathbf{x}) := \frac{dh}{d\mathbf{x}}f(\mathbf{x}), \qquad L_g h(\mathbf{x}) := \frac{dh}{d\mathbf{x}}g(\mathbf{x}).$$

Consequently, the relaxed barrier condition introduced previously can be rewritten as

$$L_f h(\mathbf{x}) + L_g h(\mathbf{x})\,\mathbf{u} + \varepsilon h(\mathbf{x}) \geq 0, \tag{3.2}$$

or, more generally,

$$L_f h(\mathbf{x}) + L_g h(\mathbf{x})\,\mathbf{u} + \alpha\big(h(\mathbf{x})\big) \geq 0, \tag{3.3}$$

where $\alpha : [0, a) \to [0, \infty)$ is an extended class $\mathcal{K}$ function, that is, a continuous and strictly increasing function satisfying $\alpha(0) = 0$.

A function $h(\mathbf{x})$ satisfying (3.3) is referred to as a *Control Barrier Function* [10]. By incorporating the control input u directly into the safety condition, CBFs provide a systematic framework for enforcing safety constraints while simultaneously allowing the controller to achieve additional objectives. In practice, the CBF condition is typically included as a constraint in an optimization problem, ensuring that the selected control action maintains the forward invariance of the safe set.

3.4 Quadratic Programming

QP is an optimization framework in which the objective function is quadratic and the constraints are linear [27]. A general QP problem can be formulated as

$$\min_{\mathbf{x}} \; f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + c^\top \mathbf{x}$$

subject to the linear constraints

$$A\mathbf{x} = \mathbf{b}$$

and

$$D\mathbf{x} \leq \mathbf{q}, \tag{3.4}$$

where Q is a symmetric positive semidefinite matrix, c is a vector of coefficients, and A, D, $\mathbf{b}$, and $\mathbf{q}$ define the equality and inequality constraints.

This framework enables the incorporation of CBFs into control-affine systems of the form (3.1). A common approach consists of minimizing the deviation between a nominal control input, or a control input generated by a path-following algorithm, and the control input obtained after enforcing the safety constraints. In this way, safety can be guaranteed while minimally affecting the nominal system behaviour.

A typical cost function is given by

$$\min_{\mathbf{u}} \frac{1}{2}(\mathbf{u}-\mathbf{u}_{\text{nom}})^{\top} W (\mathbf{u}-\mathbf{u}_{\text{nom}}) \tag{3.5}$$

where $\mathbf{u}_{\text{nom}}$ denotes the nominal control input and $W \in \mathbb{R}^{q\times q}$ is a symmetric positive definite weighting matrix that determines the relative cost associated with each component of the control input.

The CBF condition presented in (3.3) is then introduced as an inequality constraint of the optimization problem. Consequently, at each control step, the QP computes the control action that remains as close as possible to the nominal control input while guaranteeing that the safety constraints are satisfied.

3.5 Control Lyapunov Functions

CLFs are often presented alongside CBF due to their close relationship and because both can be naturally unified within a QP framework [7], [8], [9]. While CLFs are used to enforce stability by driving the system state towards a desired equilibrium, CBFs are employed to guarantee safety by ensuring that the system remains within a prescribed safe set.

An alternative way of interpreting CBFs is through a Lyapunov-based perspective. Consider the Lyapunov candidate function

$$V(\mathbf{x}) = -h(\mathbf{x}).$$

Substituting this expression into the Lyapunov stability condition

$$\dot{V}(\mathbf{x}) + \varepsilon V(\mathbf{x}) \leq 0,$$

yields

$$-\dot{h}(\mathbf{x}) - \varepsilon h(\mathbf{x}) \leq 0,$$

which can be rewritten as

$$\dot{h}(\mathbf{x}) + \varepsilon h(\mathbf{x}) \geq 0.$$

This is precisely the relaxed barrier condition introduced previously. For this reason, $h(\mathbf{x})$ is often referred to as a *Lyapunov-like barrier function*, highlighting the strong connection between Lyapunov stability theory and barrier-based safety analysis.

By combining CLF and CBF constraints, it is possible to formulate the following QP:

$$\begin{aligned} \min_{u,s} \quad & \frac{1}{2}(\mathbf{u}-\mathbf{u}_{\text{nom}})^T W(\mathbf{u}-\mathbf{u}_{\text{nom}}) + p_{slack}s^2 \\ \text{s.t.} \quad & L_f h(\mathbf{x}) + L_g h(\mathbf{x})\,\mathbf{u} + \alpha\big(h(\mathbf{x})\big) \geq 0, \\ & L_f V(\mathbf{x}) + L_g V(\mathbf{x})\,\mathbf{u} + \varepsilon V(\mathbf{x}) \leq s. \end{aligned} \tag{3.6}$$

where s is a relaxation variable introduced to prevent the CLF constraint from conflicting with the CBF constraint. This ensures that the optimization problem remains feasible even when the stability and safety objectives cannot be simultaneously satisfied. In such situations, safety is typically given priority, while the relaxation variable allows temporary violations of the CLF condition at the cost of an increased objective function value.

3.6 High-Order Control Barrier Functions

However, the CBF framework presented previously is only effective if the control input u appears explicitly in the constraint (3.3), that is, if $L_g h(\mathbf{x}) \neq 0$. Otherwise, the control input cannot influence the barrier condition, making it impossible to steer the system away from unsafe regions.

This issue arises in the Adaptive Cruise Control example illustrated in Figure 3.6, whose dynamics are given by

$$\begin{bmatrix} \dot{z} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v_p - v \\ u \end{bmatrix}.$$

Equivalently, in control-affine form,

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

with

$$f(\mathbf{x}) = \begin{bmatrix} v_p - v \\ 0 \end{bmatrix},$$

$$g(\mathbf{x}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and the barrier function defined as

$$h(\mathbf{x}) = z - d_0.$$

The gradient of the barrier function is

$$\frac{\partial h}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

and therefore

$$\dot{h}(\mathbf{x}) = \frac{\partial h}{\partial \mathbf{x}} f(\mathbf{x}) + \frac{\partial h}{\partial \mathbf{x}} g(\mathbf{x})\mathbf{u}.$$

Substituting the corresponding expressions yields

$$\dot{h}(\mathbf{x}) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v_p - v \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u},$$

resulting in

$$\dot{h}(\mathbf{x}) = v_p - v.$$

It can be observed that the control input is cancelled through the term $L_g h(\mathbf{x})$, making the barrier condition independent of the control action. Consequently, the controller cannot directly enforce the safety constraint.

However, differentiating $h(\mathbf{x})$ once more yields

$$\ddot{h}(\mathbf{x}) = \dot{v}_p - \dot{v}, \tag{3.7}$$

where the control input $\dot{v} = u$ now appears explicitly. This observation motivates the use of HOCBFs [28], which extend the CBF framework to barrier functions with higher relative degree.

HOCBFs are defined recursively through the sequence

$$\psi_i(\mathbf{x}) = \dot{\psi}_{i-1}(\mathbf{x}) + \alpha_i(\psi_{i-1}(\mathbf{x})), \qquad i \in \{1, \ldots, m\},$$

where $\alpha_i(\cdot)$, $i \in \{1, \ldots, m\}$, are class $\mathcal{K}$ functions, $\psi_0(\mathbf{x}) = h(\mathbf{x})$, and m denotes the relative degree of the barrier function $h(\mathbf{x})$, i.e., the number of differentiations required before the control input appears explicitly in the resulting expression.

The HOCBF condition is then enforced through

$$\psi_m(\mathbf{x}) \geq 0.$$

In the previous example, the control input only appears after differentiating the barrier function twice. Therefore, the relative degree is $m = 2$, leading to

$$\psi_1 = \dot{h} + \alpha_1(h),$$

and

$$\psi_2 = \dot{\psi}_1 + \alpha_2(\psi_1).$$

The corresponding HOCBF constraint becomes

$$\dot{\psi}_1 + \alpha_2(\psi_1) \geq 0.$$

Expanding the above expression yields

$$\ddot{h} + \alpha_1'(h)\dot{h} + \alpha_2\left(\dot{h} + \alpha_1(h)\right) \geq 0,$$

where the control input appears through the term $\ddot{h}$, as shown in (3.7).

More generally, for time-invariant barrier functions, HOCBFs can be expressed in the form

$$L_f^m b(\mathbf{x}) + L_g L_f^{m-1} b(\mathbf{x})\,\mathbf{u} + O(b(\mathbf{x})) + \alpha_m(\psi_{m-1}(\mathbf{x})) \geq 0, \tag{3.8}$$

where

$$O(b(\mathbf{x})) = \sum_{i=1}^{m-1} L_f^i\left(\alpha_{m-i} \circ \psi_{m-i-1}\right)(\mathbf{x}).$$

This formulation enables the application of barrier-based safety constraints to systems whose control inputs do not appear in the first derivative of the barrier function, thereby extending the applicability of CBFs to a much broader class of dynamical systems.

3.7 Integral High Order Control Barrier Functions

In many systems, multiple control inputs are available, meaning that the control vector has dimension greater than one. In the example presented so far, the control input is simply $\mathbf{u} = \dot{v}$, meaning that only the linear acceleration of the vehicle can be controlled, allowing the vehicle to accelerate or decelerate. However, by also controlling the angular acceleration, it becomes possible not only to slow down the vehicle but also to steer it away from obstacles.

Consider now the following differential vehicle model:

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \qquad \dot{v} = \mathbf{u}_2, \qquad \dot{\theta} = \phi, \qquad \dot{\phi} = \mathbf{u}_1,$$

where $(x,y) \in \mathbb{R}^2$ denotes the two-dimensional position of the system, $v \in \mathbb{R}$ is the linear velocity, $\theta \in \mathbb{R}$ is the heading angle, $\phi \in \mathbb{R}$ represents the angular velocity, and $u_1, u_2 \in \mathbb{R}$ denote the angular and linear accelerations, respectively.

The objective is to guarantee safety by preventing collisions with circular obstacles through the constraint

$$\sqrt{(x-x_0)^2+(y-y_0)^2} \geq r,$$

where $(x_0, y_0) \in \mathbb{R}^2$ denotes the obstacle centre and $r > 0$ its safety radius.

The corresponding barrier function can be defined as

$$h(\mathbf{x}) := \sqrt{(x-x_0)^2+(y-y_0)^2} - r.$$

It can be shown that, when applying (3.8) with relative degree $m = 2$, only the control input $\mathbf{u}_2$ appears explicitly:

$$L_f^2 h(\mathbf{x}) + L_{g_2} L_f h(\mathbf{x})\, u_2 + 2L_f h(\mathbf{x}) + h(\mathbf{x}) \geq 0,$$

where

$$g_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$$

represents the second column of $g(\mathbf{x})$.

To make the angular acceleration input u_1 appear explicitly, the relative degree must be increased to $m = 3$, yielding

$$\begin{aligned} L_f^3 h(\mathbf{x}) + L_{g_1} L_f^2 h(\mathbf{x})\, u_1 + L_f [L_{g_2} L_f h(\mathbf{x})]\, u_2 + L_{g_2} L_f h(\mathbf{x})\, \dot{u}_2 \\ + 3L_f^2 h(\mathbf{x}) + 3L_{g_2} L_f h(\mathbf{x})\, u_2 + 3L_f h(\mathbf{x}) + h(\mathbf{x}) \geq 0, \end{aligned} \tag{3.9}$$

where

$$g_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

represents the first column of $g(\mathbf{x})$.

In this case, both the linear and angular accelerations contribute to the safety constraint. However, since u_1 and u_2 appear with different relative degrees, namely $m = 3$ and $m = 2$, respectively, the derivative $\dot{u}_2$ also appears in (3.9). This becomes problematic when formulating the corresponding QP.

To address this issue, [10] introduces auxiliary dynamics for control inputs whose derivatives appear explicitly in the HOCBF:

$$\dot{\mathbf{u}}_j = f_j(\mathbf{u}_j) + g_j(\mathbf{u}_j) v_j, \tag{3.10}$$

where $j \in S_d$ identifies each control input whose derivatives appear explicitly in the HOCBF. The set S_d contains the indices of these inputs and $N_d = |S_d|$ denotes the total number of such inputs.

The vector

$$\mathbf{u}_j = (u_{j,1}, u_{j,2}, \ldots, u_{j,m_j}) \in \mathbb{R}^{m_j}$$

represents the auxiliary state associated with the control input u_j, with $u_{j,1} = u_j$. The quantity m_j denotes the highest derivative order of u_j appearing explicitly in the HOCBF, satisfying

$$1 \leq m_j < \bar{m},$$

where $\bar{m}$ is the maximum relative degree associated with the safety constraint $h(x) \geq 0$.

Furthermore, $\nu_j \in \mathbb{R}$ denotes the new auxiliary control input, while

$$f_j : \mathbb{R}^{m_j} \to \mathbb{R}^{m_j}, \qquad g_j : \mathbb{R}^{m_j} \to \mathbb{R}^{m_j}$$

define the dynamics of the auxiliary system.

As a consequence, control inputs whose derivatives appear in the HOCBF are treated as state variables, while the corresponding auxiliary inputs ν_j become optimization variables within the QP.

The resulting HOCBF can then be written as

$$L_f^{\bar{m}} h(\mathbf{x}) + \left[L_{\mathbf{g}_n} L_f^{\bar{m}-1} h(\mathbf{x}) \right] \mathbf{u}_n + O(h(\mathbf{x}), \mathbf{u}_a, \boldsymbol{\nu}) + \alpha_{\bar{m}}(\psi_{\bar{m}-1}(\mathbf{x}, \mathbf{u}_a)) \geq 0,$$

where $\mathbf{u}_n$ contains the control inputs whose derivatives do not appear explicitly in the HOCBF.

The vector $\mathbf{u}_a$ is defined as the concatenation of all auxiliary states associated with the control inputs whose derivatives appear in the HOCBF.

Let $S_d = \{j_1, j_2, \ldots, j_{N_d}\}$. The vector $\mathbf{u}_a$ is defined as

$$\mathbf{u}_a = \left[\mathbf{u}_{j_1}^\top, \mathbf{u}_{j_2}^\top, \ldots, \mathbf{u}_{j_{N_d}}^\top \right]^\top.$$

Similarly, the vector

$$\boldsymbol{\nu} = \left[\nu_{j_1}, \nu_{j_2}, \ldots, \nu_{j_{N_d}} \right]^\top.$$

contains all auxiliary control inputs, where each $\nu_j \in \mathbb{R}$, $j \in S_d$, is associated with the highest-order derivative of the corresponding control input u_j.

The term

$$O(h(\mathbf{x}), \mathbf{u}_a, \boldsymbol{\nu})$$

contains all remaining terms of the HOCBF expression.

Once the QP has been solved and the optimal values of $\mathbf{u}_n$ and ν_j have been obtained, the original control inputs $\mathbf{u}_j$ can be recovered through integration of (3.10). For this reason, the resulting formulation is referred to as an iHOCBFs.

Returning to the previous example, the following auxiliary dynamics can be introduced for $\dot{u}_2$:

$$\dot{u}_2 = \nu.$$

The iHOCBF constraint then becomes

$$\begin{aligned} L_f^3 h(\mathbf{x}) + L_{g_1} L_f^2 h(\mathbf{x}) u_1 + L_f \left[L_{g_2} L_f h(\mathbf{x}) \right] u_2 \\ + L_{g_2} L_f h(\mathbf{x}) \nu + 3 L_f^2 h(\mathbf{x}) \\ + 3 L_{g_2} L_f h(\mathbf{x}) u_2 + 3 L_f h(\mathbf{x}) + h(\mathbf{x}) \geq 0. \end{aligned}$$

After solving the QP and obtaining the optimal value of ν, the control input u_2 can be recovered through

$$u_2(t + \Delta t) = u_2(t) + \nu \Delta t.$$

Chapter 4

Single-Vehicle Safety Control

Based on the Control Barrier Function theory presented in the previous section, an application example is now considered. The objective is to simulate a vehicle travelling along a track while avoiding several circular obstacles placed throughout its path.

A kinematic bicycle model is adopted, Fig. 4.1, where the vehicle is controlled through its linear velocity and steering angle. Initially, path following is performed by an external controller, i.e., without incorporating a Lyapunov objective into the QP formulation. A comparison with a Lyapunov based approach will be presented later.

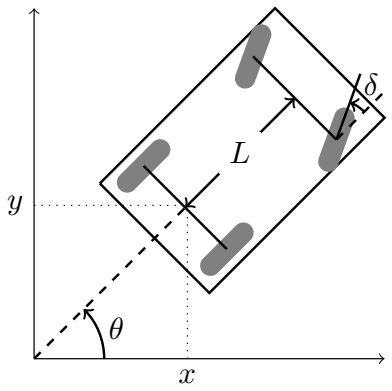


Figure 4.1: Kinematic bicycle model.

The vehicle dynamics are given by

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \dfrac{v}{L}\tan(\delta) \end{cases}$$

where $x, y \in \mathbb{R}$ denote the vehicle position in the plane, $\theta \in \mathbb{R}$ represents the heading angle, $v \in \mathbb{R}$ is the linear velocity, $\delta \in \mathbb{R}$ is the steering angle, and $L \in \mathbb{R}$ denotes the wheelbase.

However, the presence of the term $\tan(\delta)$ prevents the system from being written directly in control affine form. To overcome this issue, $\dot{\theta}$ is replaced by the angular velocity ω, allowing the system to be expressed as

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} v \\ \omega \end{bmatrix},$$

$$f(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad g(\mathbf{x}) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}.$$

After computing the optimal value of ω, the corresponding steering angle δ can be recovered through

$$\delta = \arctan\left(\frac{L\omega}{v}\right).$$

4.1 CBF with an External Controller

The next step is to define a barrier function for obstacle avoidance. For simplicity, the vehicle is approximated as a circular body with radius r_{robot}, while the obstacles are modelled as circles with radius r_{obs}.

4.1.1 Formulation

The objective is to guarantee that the distance between the vehicle and an obstacle remains greater than a predefined safety distance, i.e.,

$$dist \geq r_{safe},$$

where

$$dist = \sqrt{d_x^2 + d_y^2} = \sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}$$

and

$$r_{safe} = r_{obs} + r_{robot} + margin,$$

as illustrated in Figure 4.2.

Equivalently, the safety condition can be expressed as

$$dist - r_{safe} \geq 0,$$

or

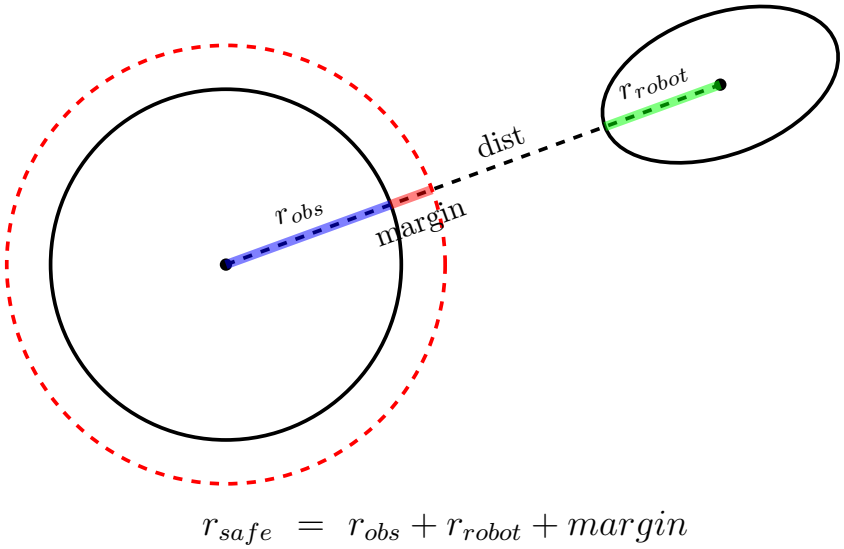


Figure 4.2: Safety distance definition.

$$dist^2 - r_{safe}^2 \geq 0,$$

which will be more convenient for the subsequent derivations. Since this condition is equivalent to requiring $h(\mathbf{x}) \geq 0$, the barrier function is defined as

$$h(\mathbf{x}) = dist^2 - r_{safe}^2 = d_x^2 + d_y^2 - r_{safe}^2.$$

Since

$$\dot{h}(\mathbf{x}) = \frac{\partial h}{\partial \mathbf{x}} \dot{\mathbf{x}},$$

its gradient is given by

$$\frac{\partial h}{\partial \mathbf{x}} = \begin{bmatrix} 2d_x & 2d_y & 0 \end{bmatrix},$$

assuming r_{safe} to be constant, as its variation is negligible compared with the variation of the vehicle obstacle distance.

Therefore,

$$\dot{h}(\mathbf{x}) = \begin{bmatrix} 2d_x & 2d_y & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}. \tag{4.1}$$

Substituting the vehicle kinematics,

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \qquad \dot{\theta} = \omega,$$

yields

$$\dot{h}(\mathbf{x}) = \begin{bmatrix} 2d_x & 2d_y & 0 \end{bmatrix} \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}.$$

Applying the CBF condition (3.2) results in

$$2d_x v\cos\theta + 2d_y v\sin\theta + \varepsilon\left(d_x^2 + d_y^2 - r_{safe}^2\right) \geq 0.$$

Finally, rewriting the constraint in the standard QP form (3.4) gives

$$\begin{bmatrix} -2(d_x\cos\theta + d_y\sin\theta) & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \leq \varepsilon\left(d_x^2 + d_y^2 - r_{safe}^2\right), \tag{4.2}$$

with the cost function

$$\min_{\mathbf{u}} \quad \frac{1}{2}(\mathbf{u} - \mathbf{u}_{\text{nom}})^T W (\mathbf{u} - \mathbf{u}_{\text{nom}}),$$

where

$$W = \begin{bmatrix} W_v & 0 \\ 0 & W_\omega \end{bmatrix},$$

and $W_v, W_\omega > 0$ denote the weights associated with the linear and angular velocity inputs, respectively.

From (4.2), it can be readily observed that the angular velocity component does not appear in the CBF constraint. Consequently, the barrier function only affects the linear velocity, meaning that the vehicle can only slow down as it approaches an obstacle rather than actively steering around it.

To address this limitation, HOCBF techniques must be employed in order to introduce the angular velocity component ω into the safety constraint. Defining

$$\psi_0(\mathbf{x}) = h(\mathbf{x}),$$
$$\psi_1(\mathbf{x}) = \dot{\psi}_0(\mathbf{x}) + \alpha_1\psi_0(\mathbf{x}).$$

and

$$\psi_2(\mathbf{x}) = \dot{\psi}_1(\mathbf{x}) + \alpha_2\psi_1(\mathbf{x}) \geq 0,$$

results in

$$\psi_2(\mathbf{x}) = \ddot{h} + (\alpha_1 + \alpha_2)\dot{h} + \alpha_1\alpha_2 h \geq 0.$$

Therefore, the second derivative of $h(\mathbf{x})$ must be computed:

$$\ddot{h}(\mathbf{x}) = \frac{d}{dt}\left[2v(d_x\cos\theta + d_y\sin\theta)\right].$$

Applying the product rule yields

$$\ddot{h}(\mathbf{x}) = 2\dot{v}(d_x\cos\theta + d_y\sin\theta) + 2v\frac{d}{dt}(d_x\cos\theta + d_y\sin\theta).$$

Since

$$\dot{d}_x = v\cos\theta, \qquad \dot{d}_y = v\sin\theta,$$

it follows that

$$\frac{d}{dt}(d_x\cos\theta + d_y\sin\theta) = v + \omega(-d_x\sin\theta + d_y\cos\theta).$$

Consequently,

$$\ddot{h}(\mathbf{x}) = 2\dot{v}(d_x\cos\theta + d_y\sin\theta) + 2v^2 + 2v\omega(-d_x\sin\theta + d_y\cos\theta).$$

The resulting HOCBF constraint is therefore

$$\begin{aligned}\psi_2(\mathbf{x}) =& 2\dot{v}(d_x\cos\theta + d_y\sin\theta) + 2v^2 + 2v\omega(-d_x\sin\theta + d_y\cos\theta)\\ &+ 2v(\alpha_1 + \alpha_2)(d_x\cos\theta + d_y\sin\theta)\\ &+ \alpha_1\alpha_2(d_x^2 + d_y^2 - r_{safe}^2) \geq 0.\end{aligned}$$

However, the presence of the linear acceleration term $\dot{v}$ introduces a nonlinearity that complicates the solution of the QP. Following a similar approach to that employed in iHOCBF formulations, an auxiliary dynamics is introduced for the linear velocity:

$$\dot{v} = \nu,$$

where ν becomes a new control input associated with the auxiliary dynamics.

The augmented state vector is then defined as

$$\mathbf{z} = \begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix},$$

while the input vector becomes

$$\mathbf{u} = \begin{bmatrix} v \\ \omega \end{bmatrix}.$$

The resulting system dynamics are

$$\begin{cases} \dot{x} = v\cos\theta, \\ \dot{y} = v\sin\theta, \\ \dot{\theta} = \omega, \\ \dot{v} = \nu. \end{cases}$$

Substituting $\dot{v}$ with ν gives

$$\ddot{h}(\mathbf{z}) = 2\nu(d_x\cos\theta + d_y\sin\theta) + 2v^2 + 2v\omega(-d_x\sin\theta + d_y\cos\theta).$$

Therefore, the iHOCBF constraint becomes

$$\begin{aligned} \psi_2(\mathbf{z}) =& 2\nu(d_x\cos\theta + d_y\sin\theta) + 2v^2 + 2v\omega(-d_x\sin\theta + d_y\cos\theta) \\ &+ 2v(\alpha_1+\alpha_2)(d_x\cos\theta + d_y\sin\theta) \\ &+ \alpha_1\alpha_2(d_x^2 + d_y^2 - r_{safe}^2) \geq 0, \end{aligned} \tag{4.3}$$

and, after solving the optimization problem and obtaining the optimal value of ν, the linear velocity is recovered through numerical integration:

$$v(t+\Delta t) = v(t) + \nu\Delta t.$$

4.1.2 Simulation Results and Discussion

Based on the previous formulations, Python simulation algorithms were developed to solve QP using both the initial CBF formulation, in which ω does not appear explicitly (4.2), and the iHOCBF formulation (4.3), allowing a comparison between the two approaches.

For the initial CBF formulation, the expected behaviour was observed: the vehicle successfully avoided collision with the obstacle solely by reducing its linear velocity, Fig. 4.3.

The Fig. 4.4a and 4.4b show the difference between the nominal control input, generated by the path following controller, and the control input obtained from the QP with collision avoidance constraints.

In the linear velocity profile, Fig. 4.4a, it can be observed that the value computed by the QP starts decreasing at approximately $2.8\,s$, corresponding to the instant at which the vehicle approaches the obstacle. In contrast, the value of ω in Fig. 4.4b remains almost identical to the nominal control input.

The variation of the steering angle δ can also be observed in Fig. 4.5. For the nominal controller, δ is simply a scaled version of ω, since the linear velocity remains constant. However, in the QP solution, the reduction in linear velocity after $2.8\,s$ causes the resulting steering angle to

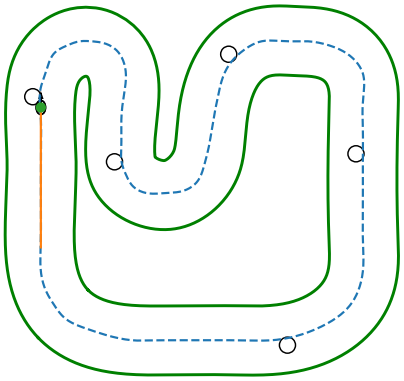
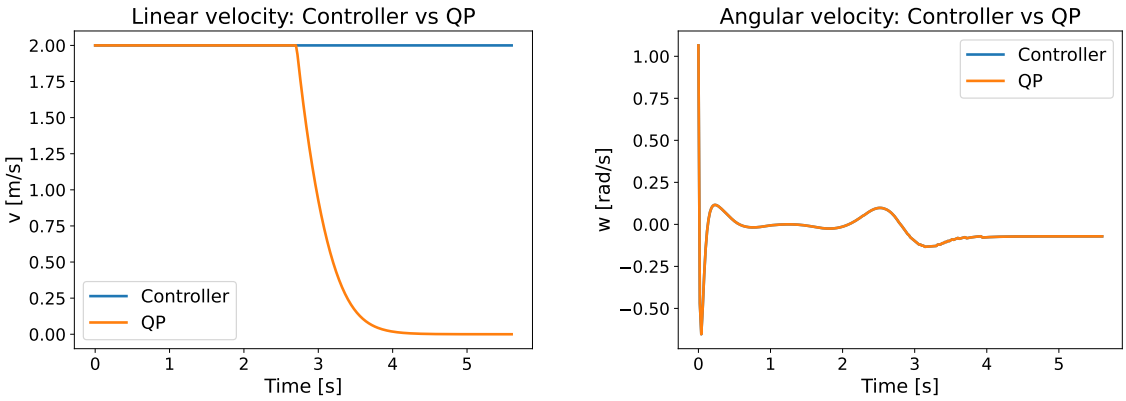


Figure 4.3: Simulation track similar to the laboratory track.



(a) Linear velocity

(b) Angular velocity

Figure 4.4: Comparison between the nominal controller and the QP solution.

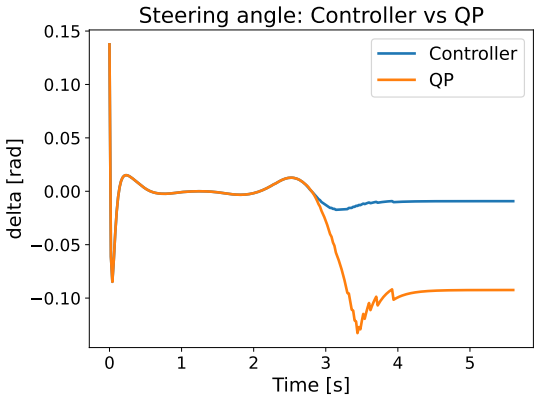


Figure 4.5: Steering angle comparison.

deviate from its nominal value as well.

After introducing the modifications required for the iHOCBF implementation, the QP was also able to act directly on ω, allowing the vehicle to actively steer around obstacles, Fig. 4.6.

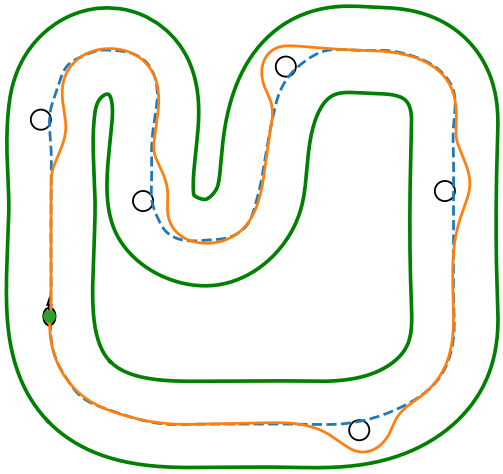
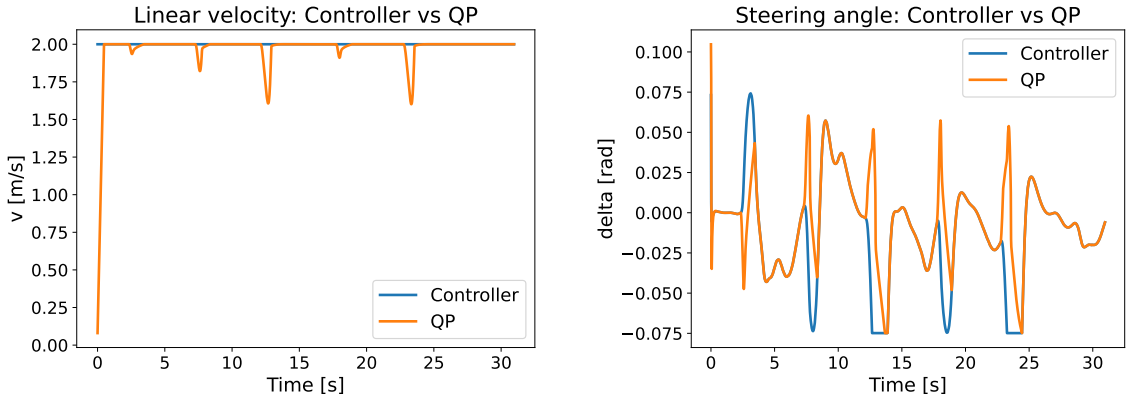


Figure 4.6: Vehicle trajectory using the iHOCBF formulation.



(a) Linear velocity

(b) Steering angle

Figure 4.7: Comparison between the nominal controller and the iHOCBF-QP solution.

In the linear velocity profile, Fig. 4.7a, small downward peaks can be observed throughout the trajectory. These correspond to the moments when the vehicle approaches an obstacle and consequently reduces its speed.

The steering angle plot, Fig. 4.7b, reveals that, at certain instants, the steering command generated by the nominal controller and the steering command produced by the QP exhibit nearly opposite behaviours. These situations correspond to obstacle avoidance manoeuvres, where the QP imposes a steering action in a given direction to prevent a collision. However, as the vehicle deviates from the reference path, the path-following controller attempts to compensate for the resulting tracking error by generating a steering action in the opposite direction in order to return the vehicle to the desired trajectory.

However, it was observed that the use of iHOCBFs introduces several disadvantages and practical challenges when compared with standard CBFs or even simpler HOCBF formulations.

One of the main drawbacks is the increased mathematical and computational complexity resulting from the successive differentiations of the barrier function. In this example, a relatively simple safety constraint was considered. Nevertheless, for more complex constraints with higher relative degrees, the resulting expressions become significantly more involved, potentially requiring multiple Lie derivatives and nonlinear terms. This not only complicates the implementation but also makes the system analysis considerably more difficult. Furthermore, the increased complexity translates directly into a higher computational burden for the QP solver, which can become problematic in real-time applications where optimization problems must be solved rapidly.

Another important challenge arises from the need to introduce auxiliary dynamics for inputs whose derivatives appear in the constraints, as was the case for the linear velocity. Although this approach allows all control inputs to appear explicitly in the iHOCBF constraint, it also increases the dimension of the system state and changes the physical interpretation of the control variables used by the QP. In the present case, the optimization problem no longer acts directly on the linear velocity but rather on its derivative, namely the linear acceleration. While this is still acceptable, if the original control input were already the acceleration, the optimization problem would instead act on its derivative, potentially leading to less intuitive behaviour and making controller tuning more difficult.

Moreover, an inconsistency may arise between the orders of the control inputs. For example, the QP may simultaneously act on linear acceleration and angular velocity, rather than on two variables of the same dynamic order. This mismatch can result in unbalanced system responses, since each input affects the system through different dynamic mechanisms. In practice, this may lead to less smooth behaviour, difficulties in tuning the QP weights, and the need to introduce additional compatibility layers between the optimization framework and the tracking controller.

Another relevant aspect is that increasing the relative degree may also make the system more sensitive to noise and modelling uncertainties. Since the constraints depend on successive derivatives, small state-estimation errors can propagate and become amplified through the differentiation process, ultimately reducing the robustness of the controller. Consequently, practical implementations of iHOCBFs may require additional filtering, more accurate state estimation, and greater care in the selection of the class $\mathscr{K}$ function parameters.

Finally, although iHOCBFs make it possible to handle systems in which certain control inputs do not appear directly in the derivative of the barrier function, this flexibility comes at the cost of a more complex and less intuitive control architecture, making the implementation, validation, and maintenance of the controller significantly more demanding.

4.2 Lookahead-Based CBF

As discussed previously, although the iHOCBF formulation allows the angular velocity to appear explicitly in the safety constraint, it introduces additional complexity due to the higher order derivatives and auxiliary dynamics required. To overcome this limitation, an alternative approach

based on a lookahead point is investigated in this dissertation and has also been presented in [29]. The main idea is to redefine the barrier function so that both control inputs appear directly in its first derivative, while preserving a simpler and more computationally efficient formulation.

4.2.1 Formulation

Therefore, the goal is to find an alternative solution capable of eliminating the mismatch between the derivative orders of the control inputs and, consequently, reducing the relative degree of the CBF in autonomous vehicle applications.

From (4.1), it can be observed that ω does not appear in the CBF constraint because the partial derivative of $h(\mathbf{x})$ with respect to θ is equal to zero. Intuitively, this means that a variation in θ alone does not produce an immediate change in the vehicle position.

To overcome this limitation, a lookahead point can be introduced, such that the safety constraint is no longer applied to the centre of the vehicle but instead to a point located slightly ahead of it.

As a result, a variation in θ does not affect the position of the vehicle centre, but it immediately changes the position of the lookahead point, allowing ω to directly influence the CBF.

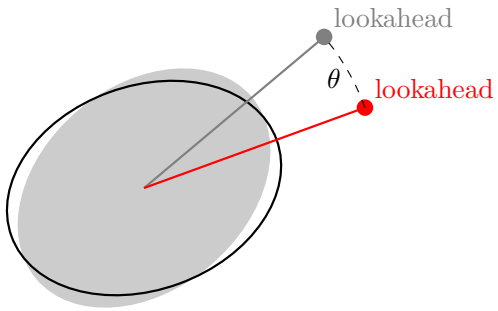


Figure 4.8: Lookahead point concept [29].

The lookahead point is defined as

$$l_x = x + l\cos(\theta), \qquad l_y = y + l\sin(\theta),$$

where l denotes the distance between the vehicle centre and the lookahead point. The barrier function then becomes

$$h(\mathbf{x}) = (l_x - x_{\text{obs}})^2 + (l_y - y_{\text{obs}})^2 - r_{\text{safe}}^2.$$

Equivalently,

$$h(\mathbf{x}) = d_x^2 + d_y^2 - r_{\text{safe}}^2,$$

where

$$d_x = l_x - x_{\text{obs}}, \qquad d_y = l_y - y_{\text{obs}}.$$

The corresponding partial derivatives are

$$\frac{\partial h}{\partial x} = 2d_x,$$

$$\frac{\partial h}{\partial y} = 2d_y,$$

and

$$\frac{\partial h}{\partial \theta} = -2ld_x \sin\theta + 2ld_y \cos\theta.$$

Therefore,

$$\dot{h}(\mathbf{x}) = \begin{bmatrix} 2d_x \\ 2d_y \\ -2ld_x \sin\theta + 2ld_y \cos\theta \end{bmatrix}^\top \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}.$$

Making the control inputs explicit yields

$$\dot{h}(\mathbf{x}) = \begin{bmatrix} 2(d_x \cos\theta + d_y \sin\theta) \\ 2l(-d_x \sin\theta + d_y \cos\theta) \end{bmatrix}^\top \begin{bmatrix} v \\ \omega \end{bmatrix}.$$

It can now be clearly observed that the angular velocity term ω is no longer cancelled.

4.2.2 Simulation Results and Discussion

After incorporating these modifications into the algorithm, new simulations were performed, producing the results presented below.

The vehicle was now able to complete an entire lap while successfully avoiding all obstacles, confirming the effectiveness of the lookahead approach, Fig. 4.9. However, a closer inspection reveals that, in some situations, the vehicle avoids an obstacle from the less efficient side, as can be observed for the third and last obstacles.

This behaviour results from the interaction between the various parameter values and the QP cost function. The optimization problem seeks a control input that satisfies the safety constraints imposed by the CBF while remaining as close as possible to the nominal control generated by the path-following controller.

Considering the third obstacle, for example, the CBF starts reacting when the vehicle orientation is more favourable to avoiding the obstacle on the left-hand side rather than on the right-hand side, even though the latter would be the preferred option from the perspective of path tracking. To address this issue, the parameters should be adjusted such that the vehicle can approach the

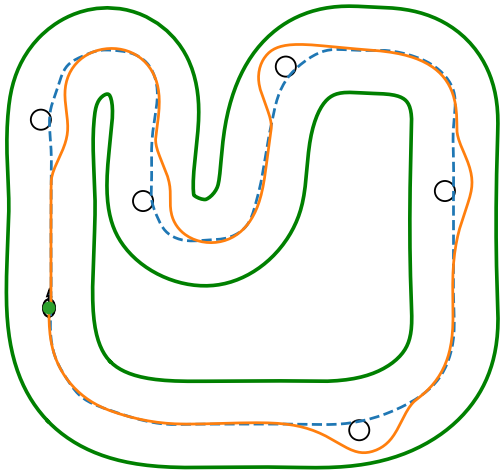


Figure 4.9: Vehicle trajectory using the lookahead-based CBF.

obstacle more closely before the CBF becomes active, allowing the vehicle orientation to evolve into a configuration that favours obstacle avoidance on the side most compatible with the desired trajectory.

One of the parameters with the greatest influence on obstacle avoidance performance is the class $\mathscr{K}$ function parameter α, which determines how aggressively the vehicle reacts and how closely it is allowed to approach an obstacle.

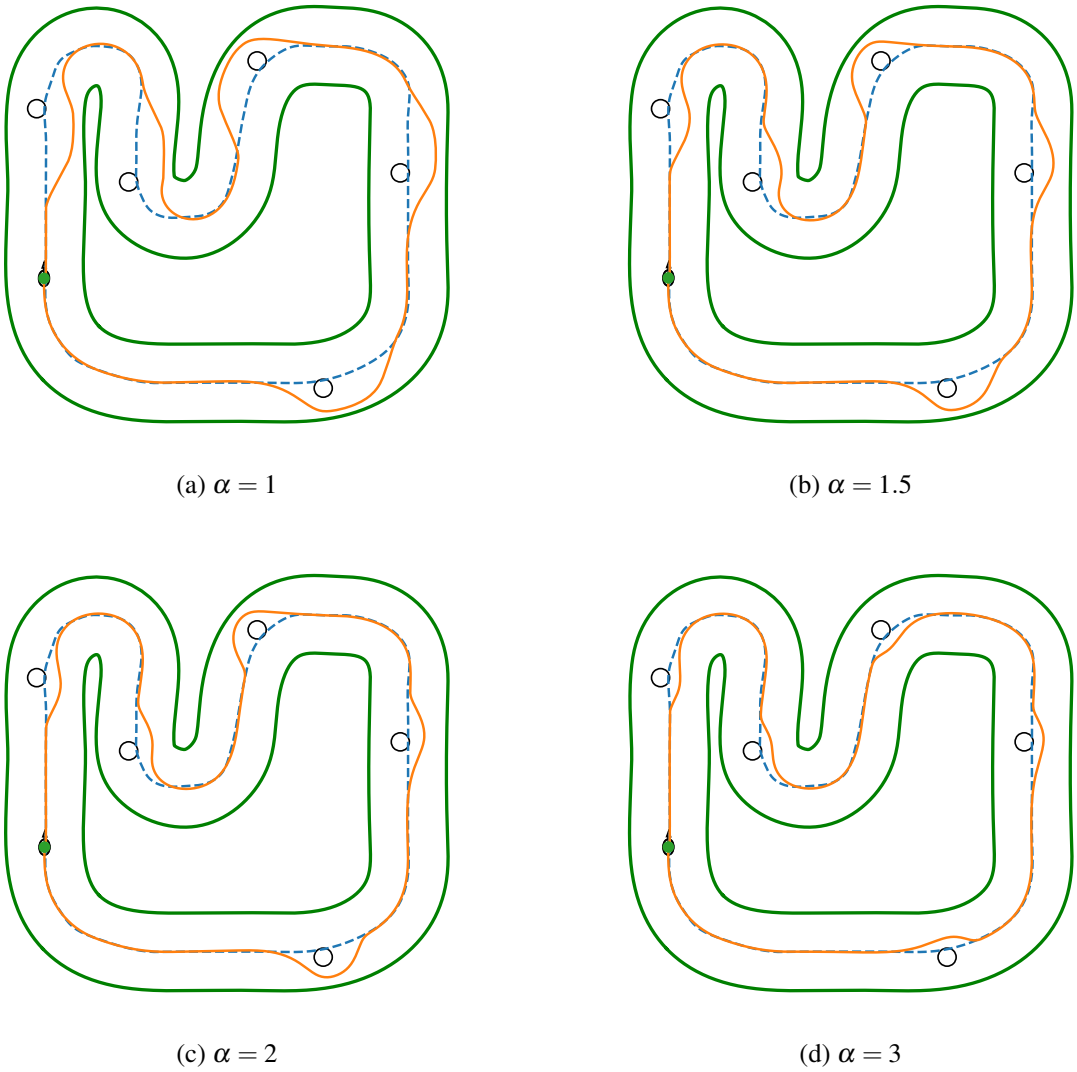
From Figure 4.10, it can be observed that increasing the value of α allows the vehicle to approach obstacles more closely before the CBF constraint becomes dominant. As a consequence, the resulting avoidance manoeuvres become smoother and less conservative, while also increasing the likelihood of selecting the side that is more favourable for path tracking, as illustrated in Figure 4.10(d).

Another parameter that influences the distance at which the vehicle starts reacting to an obstacle is the lookahead distance. In general, a smaller lookahead causes the vehicle to react later, allowing it to approach the obstacle more closely before initiating the avoidance manoeuvre. This behaviour can be observed in Figure 4.11, where different lookahead values produce noticeably different trajectories around the obstacles.

However, some important considerations must be taken into account when selecting the lookahead distance.

If the lookahead distance is chosen too small, as shown in Figure 4.12(a), the influence of the angular velocity on the barrier function becomes weaker, for the reasons discussed previously. Consequently, the CBF loses part of its ability to steer the vehicle around obstacles.

On the other hand, if the lookahead distance becomes excessively large, as illustrated in Figure 4.12(b), the vehicle may collide with an obstacle despite the lookahead point remaining outside the unsafe region. This occurs because the safety constraint is enforced on the lookahead point rather than on the vehicle itself, causing the controller to react to a virtual point located ahead of



(a) $\alpha = 1$ (b) $\alpha = 1.5$

(c) $\alpha = 2$ (d) $\alpha = 3$

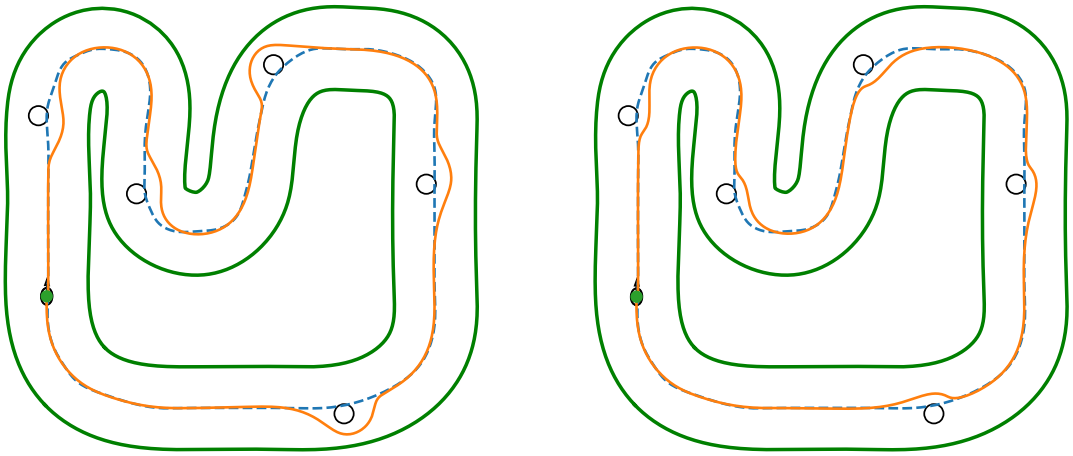
Figure 4.10: Influence of the class $\mathcal{K}$ parameter α on obstacle avoidance behaviour.

the vehicle body.

Additional parameters with a significant impact on the vehicle behaviour are the weights W_v and W_ω, which determine the relative cost associated with modifying the linear and angular velocities, respectively.

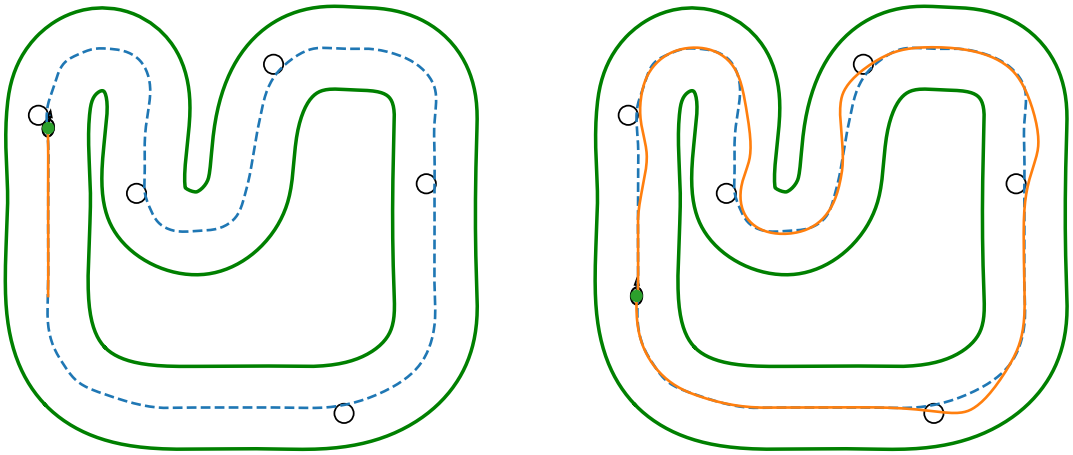
Choosing $W_v \gg W_\omega$ assigns a much higher cost to variations in linear velocity than to variations in angular velocity. Consequently, the optimization problem tends to preserve the nominal linear velocity and instead relies more heavily on steering actions to satisfy the safety constraints. This generally results in a faster vehicle, as less speed reduction is required. However, the vehicle also tends to initiate avoidance manoeuvres earlier and more aggressively, preventing it from approaching the obstacle as closely and potentially leading to the undesired behaviour previously observed for the third obstacle, Fig. 4.13b.

Conversely, selecting a smaller value of W_v, Fig. 4.13a, allows larger variations in the linear



(a) $l = 0.1$ (b) $l = 0.01$

Figure 4.11: Influence of the lookahead distance on obstacle avoidance.

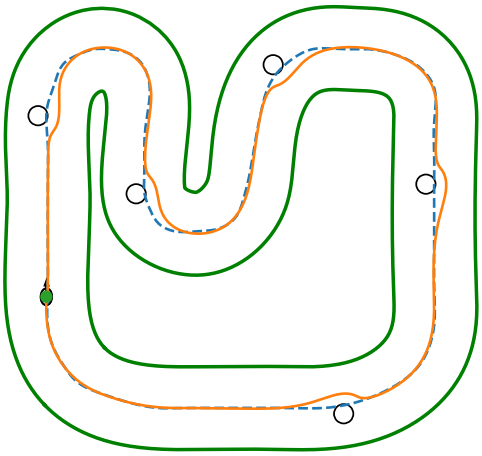


(a) $l = 0.001$ (b) $l = 2$

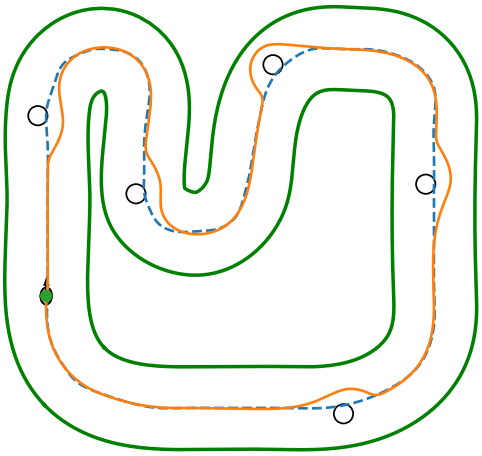
Figure 4.12: Limitations associated with excessively small and excessively large lookahead distances.

velocity while reducing the relative influence of angular velocity corrections. In this case, obstacle avoidance is achieved through a combination of steering and speed reduction, resulting in smoother steering actions but lower overall vehicle speed.

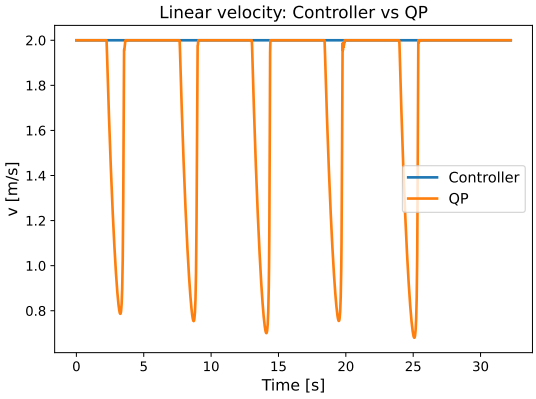
Figure 4.13 compares the cases $(W_v, W_\omega) = (200, 1)$ and $(W_v, W_\omega) = (2000, 1)$, shown in the left and right columns, respectively. It can be observed that, for the lower value of W_v, the vehicle experiences more pronounced reductions in linear velocity when approaching obstacles, while the angular velocity exhibits smoother transitions. In contrast, increasing W_v significantly reduces the speed variations and produces faster and more aggressive steering responses.



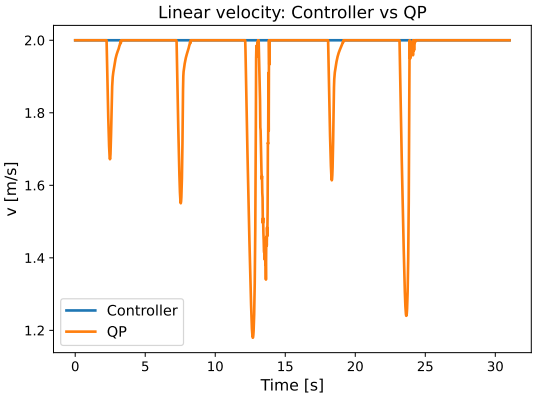
(a) $(W_v, W_\omega) = (200, 1)$



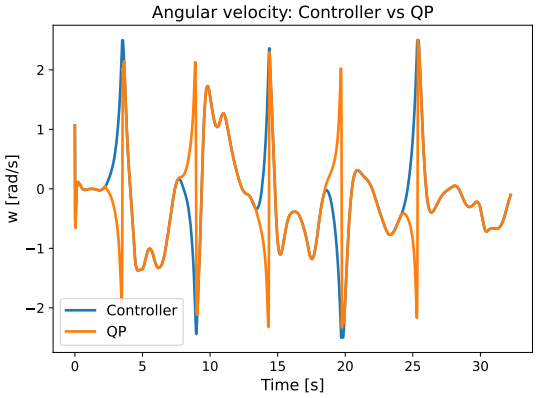
(b) $(W_v, W_\omega) = (2000, 1)$



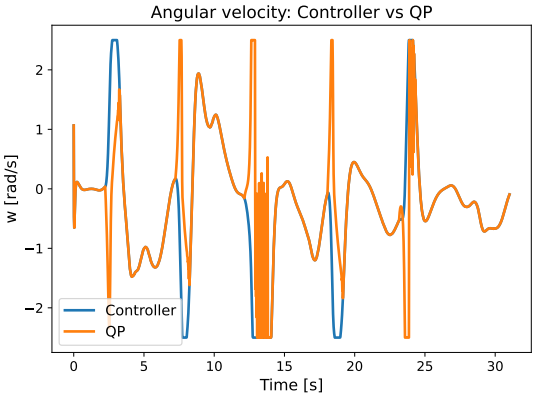
(c) Linear velocity



(d) Linear velocity



(e) Angular velocity



(f) Angular velocity

Figure 4.13: Influence of the weighting matrix parameters on the obstacle avoidance behaviour.

Finally, safety constraints were defined not only for the obstacles but also for the track boundaries, resulting in additional barrier functions. Since the boundaries do not possess a uniform geometric shape, the only difference with respect to the circular obstacle formulation lies in the

definition of r_{safe}.

From a computational perspective, the track boundaries are represented by a set of short line segments. Each segment can therefore be approximated as a circular obstacle with zero radius and centred at the point of the segment closest to the vehicle at each instant.

To evaluate the effectiveness of this approach, the vehicle was simulated on the track without obstacles and without any path-following controller. The QP was provided only with a non-zero nominal linear velocity and a zero nominal angular velocity. The resulting behaviour showed that the vehicle was able to steer away from the track boundaries and avoid collisions, as illustrated in Figure 4.14.

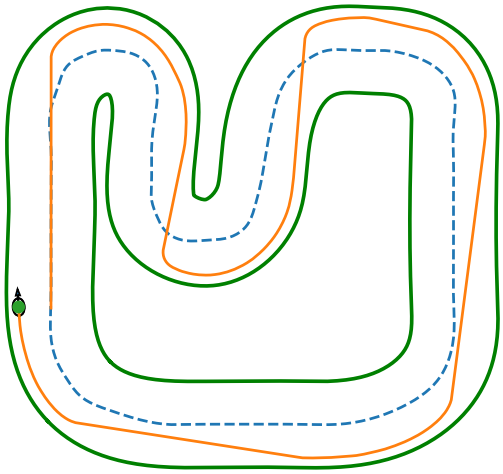


Figure 4.14: Vehicle behaviour considering only the barrier constraints associated with the track boundaries.

4.3 CLF-CBF-QP Controller

The Lyapunov-based control formulation is now incorporated into the QP. The objective is to drive the tracking error between the vehicle position and its corresponding projection onto the reference trajectory towards zero.

4.3.1 Formulation

The trajectory tracking errors are defined as

$$e_x = \cos(\theta_r)(x - x_r) + \sin(\theta_r)(y - y_r),$$

$$e_y = -\sin(\theta_r)(x - x_r) + \cos(\theta_r)(y - y_r),$$

$$e_\theta = \theta - \theta_r.$$

where (x_r, y_r) denotes the closest point on the reference trajectory to the vehicle and ψ_r is the trajectory orientation at that point.

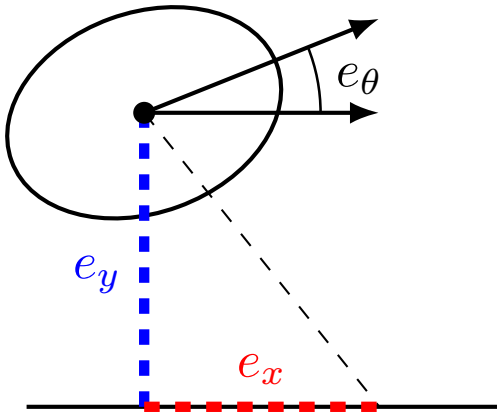


Figure 4.15: Trajectory tracking error definition [29].

However, similarly to what was previously observed for the CBF formulation, it can be anticipated that the lateral tracking error e_y does not directly depend on θ. Computing the derivatives of the error states yields

$$\dot{e}_x = v\cos(e_\theta) - v_{\text{ref}} + \omega_{\text{ref}} e_y,$$

$$\dot{e}_y = v\sin(e_\theta) - \omega_{\text{ref}} e_x,$$

$$\dot{e}_\theta = \omega - \omega_{\text{ref}}.$$

It can be observed that the lateral error dynamics do not depend on ω. Consequently, within the QP formulation, the angular velocity will not contribute directly to reducing the lateral tracking error. In other words, the vehicle would not actively steer towards the reference trajectory.

To overcome this limitation, the same lookahead technique introduced previously for the CBF is applied. The tracking errors are therefore redefined as

$$e_x = \cos(\theta_r)(x_l - x_r) + \sin(\theta_r)(y_l - y_r),$$

$$e_y = -\sin(\theta_r)(x_l - x_r) + \cos(\theta_r)(y_l - y_r),$$

$$e_\theta = \theta - \theta_r,$$

where

$$x_l = x + l\cos\theta,$$

$$y_l = y + l\sin\theta.$$

A Lyapunov function is then chosen as

$$V(e_x, e_y, e_\theta) = \frac{1}{2}\left(q_x e_x^2 + q_y e_y^2 + q_\theta e_\theta^2\right),$$

where the parameters $q_x, q_y, q_\theta > 0$ are weighting coefficients that determine the relative importance of the longitudinal, lateral, and angular tracking errors, respectively. In general, $q_y > q_x$ is selected in order to prioritize lateral convergence towards the reference trajectory.

Differentiating the Lyapunov function yields

$$\begin{aligned}\dot{V} &= \frac{\partial V}{\partial e_x}\dot{e}_x + \frac{\partial V}{\partial e_y}\dot{e}_y + \frac{\partial V}{\partial e_\theta}\dot{e}_\theta \\ &= q_x e_x \dot{e}_x + q_y e_y \dot{e}_y + q_\theta e_\theta \dot{e}_\theta.\end{aligned}$$

Substituting the error dynamics results in

$$\begin{aligned}\dot{V} = {} & q_x e_x \Big(v\cos(e_\theta) - v_{\text{ref}} + \omega_{\text{ref}} e_y - l\,\omega\sin(e_\theta)\Big) \\ & + q_y e_y \Big(v\sin(e_\theta) - \omega_{\text{ref}} e_x + l\,\omega\cos(e_\theta)\Big) \\ & + q_\theta e_\theta (\omega - \omega_{\text{ref}}).\end{aligned}$$

To guarantee the convergence of the tracking errors, a CLF constraint is introduced in the form

$$\dot{V} \leq -\varepsilon V + s,$$

where $\varepsilon > 0$ defines the convergence rate and $s \geq 0$ is a relaxation variable introduced to ensure the feasibility of the optimization problem.

Since the derivative of the Lyapunov function can be expressed in a control-affine form, it follows that

$$\dot{V} = \begin{bmatrix} q_x e_x \cos(e_\theta) + q_y e_y \sin(e_\theta) \\ -q_x e_x l \sin(e_\theta) + q_y e_y l \cos(e_\theta) + q_\theta e_\theta \end{bmatrix}^\top \begin{bmatrix} v \\ \omega \end{bmatrix} + c,$$

where

$$c = -q_x e_x v_{\text{ref}} + \omega_{\text{ref}} e_x e_y (q_x - q_y) - q_\theta e_\theta \omega_{\text{ref}}.$$

Substituting this expression into the CLF condition yields the following linear constraint, which can be directly incorporated into the QP:

$$G_{\text{clf}} \begin{bmatrix} v \\ \omega \\ s \end{bmatrix} \leq h_{\text{clf}},$$

with

$$G_{\text{clf}} = \begin{bmatrix} q_x e_x \cos(e_\theta) + q_y e_y \sin(e_\theta) & -q_x e_x l \sin(e_\theta) + q_y e_y l \cos(e_\theta) + q_\theta e_\theta & -1 \end{bmatrix},$$

and

$$h_{\text{clf}} = -\left(-q_x e_x v_{\text{ref}} + \omega_{\text{ref}} e_x e_y (q_x - q_y) - q_\theta e_\theta \omega_{\text{ref}}\right) - \varepsilon V.$$

Finally, the nominal control input from (3.6) is defined as

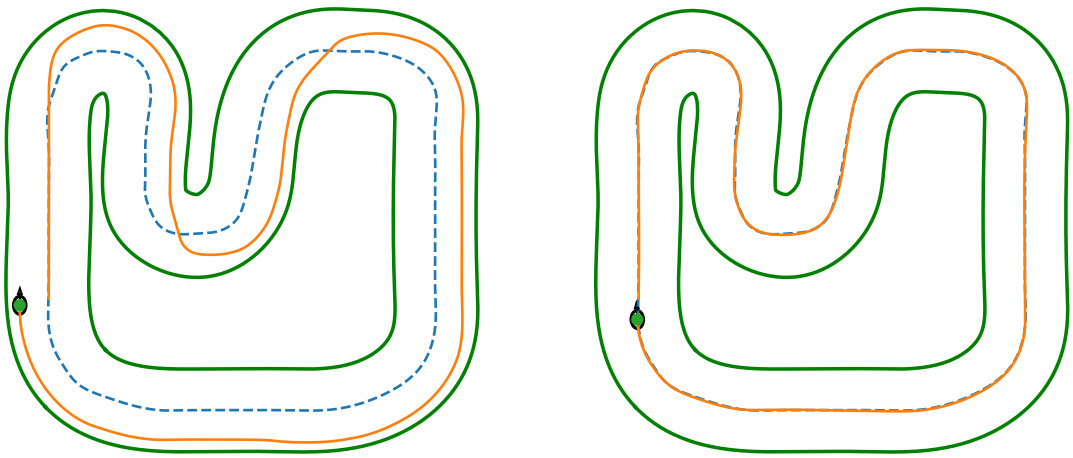
$$\mathbf{u}_{\text{nom}} = \begin{bmatrix} v_{\text{ref}} \\ 0 \end{bmatrix},$$

where v_{ref} denotes the desired forward velocity. Consequently, the objective function penalizes deviations from the reference velocity while simultaneously favoring small steering inputs. In the absence of active safety or stability constraints, the optimal solution coincides with the nominal control input, resulting in the vehicle moving forward at the desired speed without steering corrections.

4.3.2 Simulation Results and Discussion

The proposed formulation was then implemented in the Python simulation framework and several simulations were performed to analyse the vehicle behaviour.

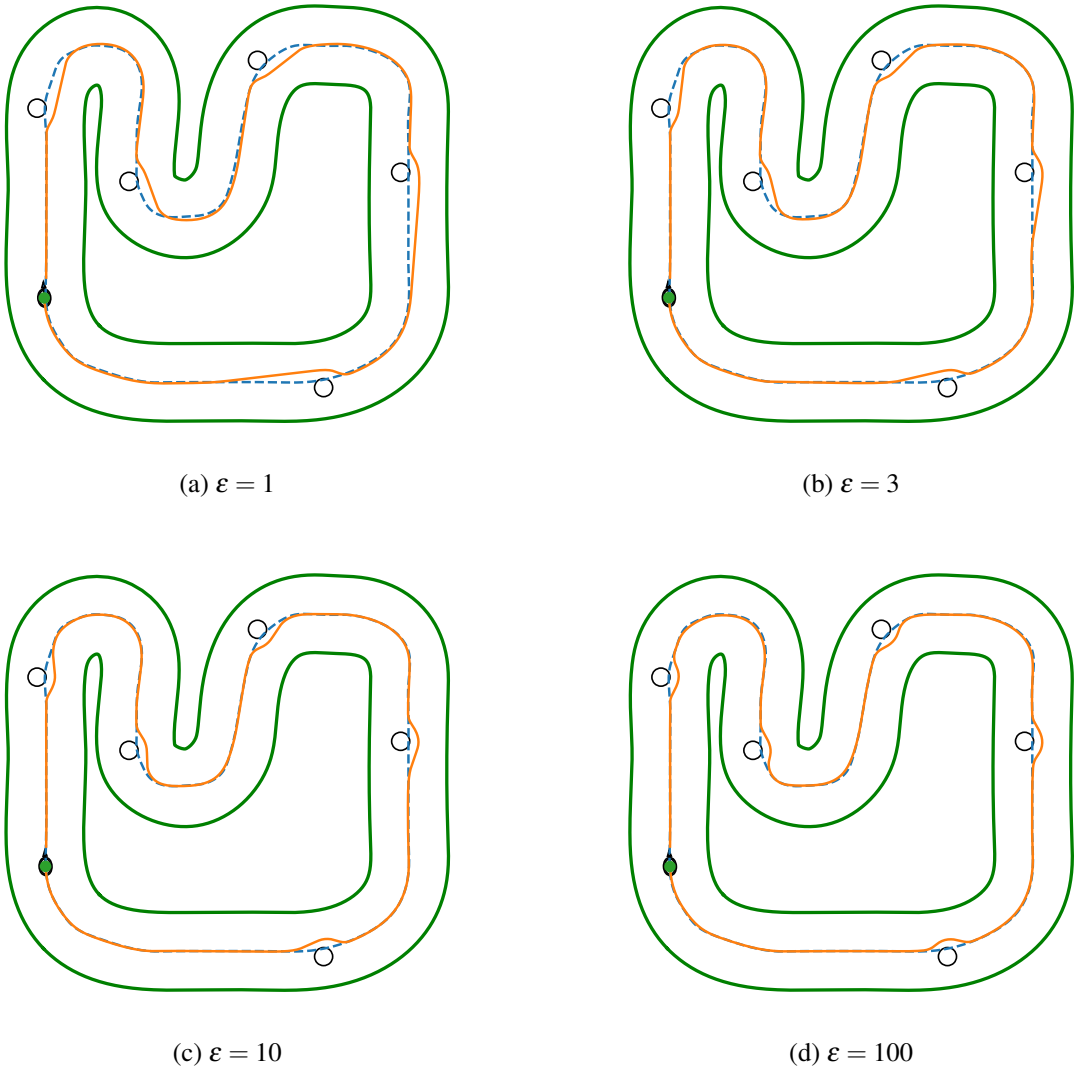
In a first simulation, the lookahead distance was set to zero in order to evaluate its influence on the correction of the lateral tracking error, Fig. 4.16a. A second simulation was then performed using a positive lookahead distance for comparison, Fig. 4.16b.



(a) $l = 0$ (b) $l = 0.5$

Figure 4.16: Influence of the lookahead distance on trajectory tracking.

The difference between the two simulations is evident. In the first case, the vehicle struggles to remain on the reference path. The only mechanisms affecting the steering behaviour are the barrier functions associated with the track boundaries and the orientation error e_θ, which merely keeps the vehicle approximately aligned with the trajectory. In contrast, when a lookahead distance is introduced, variations in ω directly influence the lateral tracking error, allowing the vehicle to effectively converge to and follow the reference trajectory.



(a) $\varepsilon = 1$

(b) $\varepsilon = 3$

(c) $\varepsilon = 10$

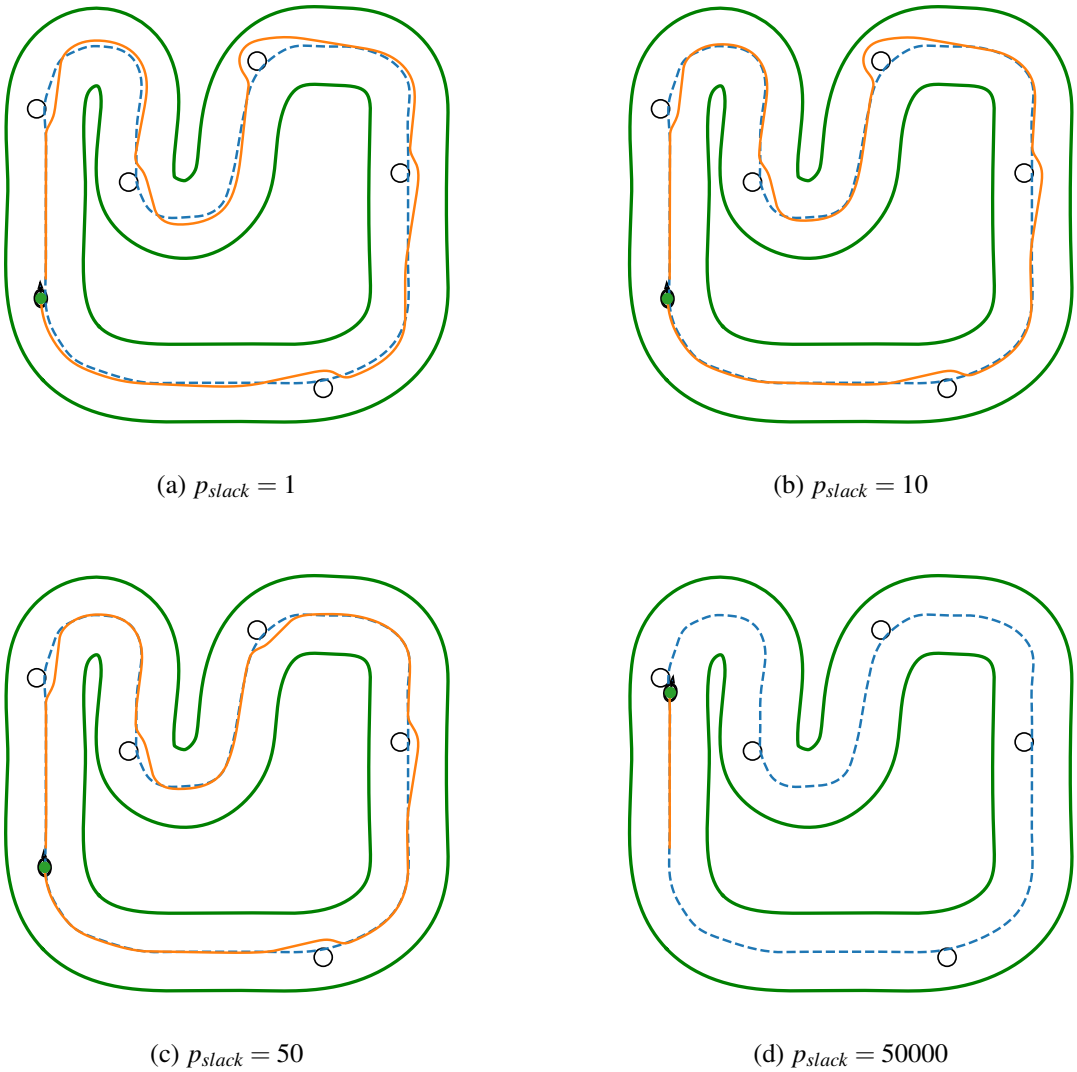
(d) $\varepsilon = 100$

Figure 4.17: Influence of the CLF convergence parameter ε.

Having previously analysed the effects of parameters such as α, W_v, and W_ω, the focus now shifts to the parameters associated with the CLF formulation, namely ε, p_{slack}, and the trajectory-tracking weights q_x, q_y, and q_θ.

Several simulations were carried out in the presence of obstacles, beginning with variations of the parameter ε. This parameter plays a role similar to that of α in the CBF formulation, determining how aggressively the vehicle attempts to converge towards the reference trajectory.

From Figure 4.17(a), corresponding to the smallest value of ε, it can be observed that the vehicle requires a longer distance to return to the trajectory after avoiding the fourth and fifth obstacles. As ε increases, the vehicle converges more rapidly to the reference path. However, faster convergence is not necessarily desirable, particularly in real-world applications, where it may result in more abrupt and aggressive manoeuvres.



(a) $p_{slack} = 1$ (b) $p_{slack} = 10$

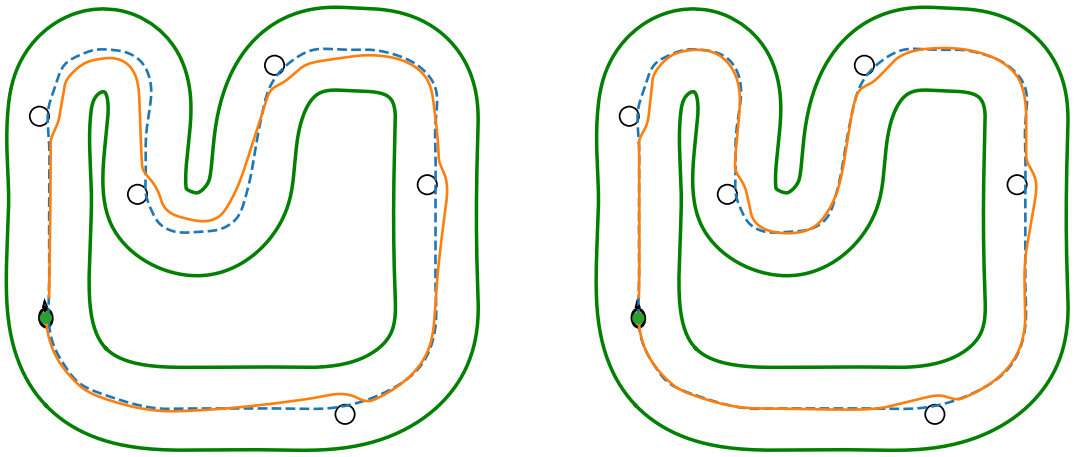
(c) $p_{slack} = 50$ (d) $p_{slack} = 50000$

Figure 4.18: Influence of the slack-variable weight p_{slack}.

Figure 4.18 presents several simulations obtained by varying the value of p_{slack}, the weight associated with the slack variable. This parameter determines the relative importance of the Lyapunov constraint with respect to the CBF constraints while preserving the feasibility of the QP.

Without the slack variable, the CLF would continuously drive the solution towards the reference trajectory, whereas the CBF would simultaneously push it away from nearby obstacles. These competing objectives could easily result in conflicting constraints.

In Figure 4.18(a), the value of p_{slack} is small, causing the vehicle to exhibit noticeable difficulties in maintaining the desired trajectory. Increasing the parameter, as shown in Figure 4.18(b), improves the tracking performance considerably, although the vehicle still avoids the third obstacle through the less desirable side. Increasing p_{slack} further, as in Figure 4.18(c), places greater emphasis on trajectory convergence, resulting in the vehicle passing the third obstacle on the right-hand side, which is more consistent with the desired path. Finally, in Figure 4.18(d), the value of p_{slack} becomes excessively large, causing the CLF constraint to dominate the optimization problem and conflict with the CBF constraints. As a consequence, the vehicle becomes unable to progress and eventually stops.



(a) $(q_x, q_y, q_\theta) = (1, 1, 1)$ (b) $(q_x, q_y, q_\theta) = (1, 10, 1)$

Figure 4.19: Influence of the trajectory-tracking weights.

Finally, Figure 4.19 illustrates the influence of the trajectory-tracking weights, particularly the lateral-error weight q_y.

In Figure 4.19(a), where all weights are set to one, the orientation error has a relatively strong influence compared with the lateral error. To increase the importance of lateral convergence, the value of q_y was increased while keeping the remaining weights unchanged. The resulting behaviour, shown in Figure 4.19(b), demonstrates a noticeably more accurate tracking of the reference trajectory.

4.4 Chapter Conclusions

This chapter presented the development of several CBF-based safety control strategies for autonomous vehicles and evaluated their performance in obstacle avoidance and trajectory-tracking scenarios.

Initially, a standard CBF formulation combined with an external path-following controller was investigated. Although this approach was able to guarantee collision avoidance, the resulting

behaviour relied primarily on reductions in the vehicle linear velocity. Since the angular velocity did not appear explicitly in the barrier constraint, the controller had limited ability to actively steer around obstacles.

To overcome this limitation, an iHOCBF formulation was explored. This approach enabled the angular velocity to directly influence the safety constraint, allowing obstacle avoidance manoeuvres to be performed through steering actions. However, the introduction of higher-order derivatives and auxiliary dynamics significantly increased the complexity of both the mathematical formulation and the resulting optimization problem.

An alternative solution based on a lookahead point was then proposed. By redefining the barrier function using a virtual point located ahead of the vehicle, the angular velocity appeared directly in the first derivative of the barrier function, eliminating the need for higher-order formulations. Simulation results demonstrated that this approach successfully enabled steering-based obstacle avoidance while maintaining a simpler and more intuitive control architecture.

Finally, the lookahead-based formulation was integrated with a CLF within a QP framework. This CLF-CBF-QP controller allowed safety and trajectory tracking objectives to be addressed simultaneously. The obtained results demonstrated that the vehicle was capable of tracking the reference path while maintaining safe distances from obstacles and track boundaries.

Overall, the results presented in this chapter show that CBFs provide an effective framework for safety-critical control of autonomous vehicles. Among the investigated approaches, the lookahead-based CLF-CBF-QP formulation achieved the best compromise between safety, trajectory tracking performance, implementation simplicity, and computational efficiency. Consequently, this formulation is adopted in the following chapters as the basis for parameter optimization, multi-vehicle interaction, and real-world experimental validation.

Chapter 5

Parameter Optimization Using Machine Learning

One of the main challenges associated with the implementation of CBFs is the selection of appropriate parameter values. In particular, parameters such as α, the QP weighting coefficients, and relaxation terms such as ε and s have a direct impact on the system behaviour, simultaneously influencing safety, trajectory smoothness, obstacle avoidance aggressiveness, and trajectory tracking performance.

However, selecting these parameters is far from straightforward, since a strong interdependence exists between them. Modifying a given parameter may significantly alter the effect of the remaining ones, resulting in behaviours that are difficult to predict analytically. Furthermore, the traditional trial and error tuning process is extremely time consuming, requiring a large number of simulations and iterative adjustments before satisfactory performance can be achieved. An example of this difficulty was previously observed in Figure 4.13, where an attempt to reduce the trajectory deviation by forcing the vehicle to avoid the third obstacle on the right hand side resulted in a reduction of the average vehicle speed.

5.1 Methodology

To overcome these limitations, Machine Learning (ML) techniques were employed to automate the parameter selection process. The main idea is to allow the system to automatically identify parameter combinations capable of achieving better performance according to predefined evaluation metrics.

In the implemented approach, the learning problem was initially simplified by focusing exclusively on the adjustment of the class $\mathscr{K}$ function, which was modified as

$$\alpha(h(\mathbf{x})) = p\,h(\mathbf{x})^q,$$

where p and q became the parameters to be optimized. Since q may assume non integer values, the function $\alpha(h) = ph^q$ is only defined for $h \geq 0$. This does not impose any practical limitation, since the CBF condition is enforced only while the system remains within the safe set.

Consequently, the CBF constraint becomes

$$\dot{h}(\mathbf{x}) + p\,h(\mathbf{x})^q \geq 0,$$

or equivalently,

$$L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} + p\,h(\mathbf{x})^q \geq 0.$$

The behaviour of the controller therefore becomes directly dependent on the parameter pair (p,q). Different parameter combinations can produce substantially different behaviours in terms of obstacle avoidance aggressiveness, minimum distance to obstacles, smoothness of the control actions, and even feasibility of the optimization problem itself.

For instance, different values of p modify the intensity with which the CBF reacts to approaching obstacles, affecting both the moment at which the QP begins to modify the control inputs and the aggressiveness of the resulting manoeuvre. On the other hand, the parameter q changes the manner in which this corrective action evolves as the vehicle approaches the safety boundary. Depending on the selected combination, the system may exhibit excessively conservative behaviour, overly aggressive responses, or even feasibility issues associated with the QP solver.

The parameter selection problem can then be formulated as the optimization problem

$$(p^\star, q^\star) = \arg\min_{p,q} J(p,q),$$

where $J(p,q)$ denotes a cost function associated with the overall system performance.

Rather than defining the optimization objective directly as a cost, it is often more intuitive to introduce a performance score based on a set of desired metrics, such as

- progress along the track;
- minimum distance to obstacles;
- lateral tracking error;
- smoothness of the control inputs.

The optimization objective can then be obtained through the transformation

$$J = -S,$$

where S denotes the performance score obtained during the simulations.

However, as discussed previously, many combinations of (p,q) may render the QP infeasible, making the optimization problem in (5.4) difficult to solve directly.

To address this issue, a classification method based on Support Vector Machines (SVMs) can be employed. SVMs are supervised learning algorithms designed to separate different classes of data by constructing a decision boundary within the parameter space [30], [31], [32]. In the present work, the objective of the SVM is to distinguish feasible parameter combinations from infeasible ones, thereby providing an approximation of the feasible region of the parameter space. This feasibility map can then be used to restrict the subsequent optimization process to regions where the controller is expected to operate successfully.

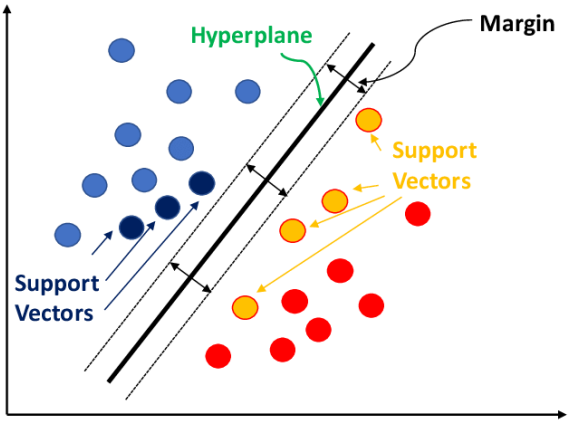


Figure 5.1: Illustration of SVM classification in the parameter space [33].

The SVM attempts to find a hyperplane that maximizes the separation between the different classes. In the present case, the objective is to separate parameter combinations (p,q) that produce feasible QP solutions from those that lead to infeasible ones. Each point in the parameter space is therefore assigned a label, typically $+1$ for feasible combinations and -1 for infeasible combinations.

Using this labelled dataset, the SVM learns a classification function $H(p,q)$ that defines a decision boundary between the feasible and infeasible regions. In particular,

$$H(p,q) \geq 0$$

represents the set of parameter combinations predicted to yield feasible solutions.

The learned function H can then be interpreted as a constraint analogous to a barrier function $h(\mathbf{x})$ by defining the auxiliary dynamical system

$$(\dot{p},\dot{q}) = v,$$

subject to

$$H(p,q) \geq 0, \tag{5.1}$$

where $v \in \mathbb{R}^2$ represents the control input of the auxiliary system used to determine the optimal parameters $p^\star$ and $q^\star$.

At this stage, all the necessary components are available to formulate an optimization problem capable of minimizing the cost function $J(p,q)$. However, the control input v must first appear explicitly in the objective function. This can be achieved by differentiating J with respect to time:

$$\frac{dJ(p,q)}{dt} = \frac{dJ(p,q)}{d(p,q)}\, v.$$

Similarly, rewriting (5.1) as a CBF constraint yields

$$\dot{H}(p,q) + \beta(H(p,q)) \geq 0,$$

which can be expressed as

$$\frac{dH(p,q)}{d(p,q)}\, v + \beta(H(p,q)) \geq 0.$$

The resulting parameter optimization problem can therefore be formulated as the following Linear Program (LP):

$$\begin{aligned} \min_{v} \quad & \frac{dJ(p,q)}{d(p,q)}\, v, \\ \text{s.t.} \quad & \frac{dH(p,q)}{d(p,q)}\, v + \beta(H(p,q)) \geq 0, \\ & v_{\min} \leq v \leq v_{\max}. \end{aligned} \tag{5.2}$$

This formulation enables the optimization process to search for improved parameter combinations while remaining within regions of the parameter space that are predicted to produce feasible QP solutions.

In this way, the parameter optimization process becomes guided by an estimate of the feasible region, significantly reducing the number of infeasible evaluations and improving the efficiency of the search for optimal parameters.

Accordingly, during the simulation process, parameter combinations (p,q) are randomly selected within predefined ranges and classified as either feasible or infeasible according to the behaviour observed during the simulation, following the procedure described previously:

$$label = \begin{cases} +1, & \text{if the simulation is considered feasible} \\ -1, & \text{if the simulation is considered infeasible} \end{cases}$$

A total of one hundred and one samples were collected over the parameter intervals $q \in [0.1, 2.5]$ and $p \in [0.1, 15]$, as shown in Fig. 5.2. From the distribution of the samples, it can be observed that the feasible and infeasible regions are not linearly separable. In other words, they cannot be separated by a single linear decision boundary. Therefore, a polynomial kernel was employed to train the SVM model, allowing nonlinear decision boundaries to be represented.

The following seventh degree polynomial kernel was selected:

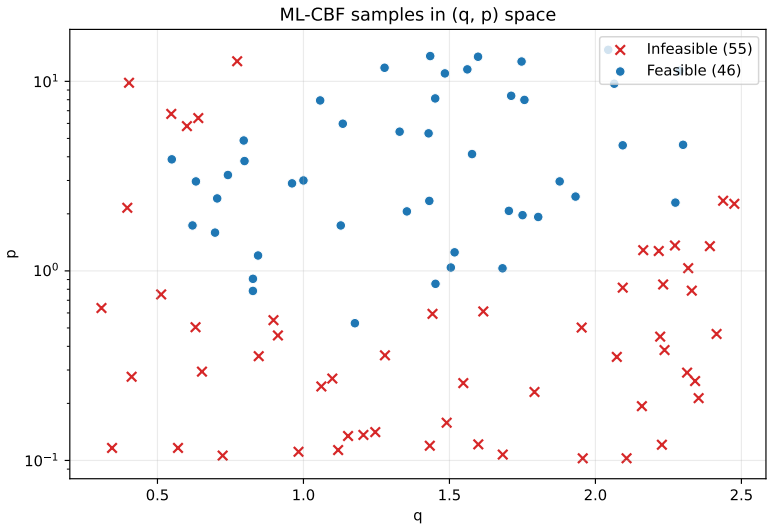


Figure 5.2: Feasible and infeasible parameter samples collected from simulations.

$$k(y,z) = \left(c_1 + c_2 y^T z\right)^7$$

where y and z represent two input vectors of the SVM, corresponding to different parameter combinations (p,q). The kernel constants were adopted from [10]. Accordingly, the values were set to $c_1 = 0.8$ and $c_2 = 0.5$, as they provide a suitable nonlinear mapping for the classification problem considered in this work.

After training the SVM model, the classification function H is obtained. A three dimensional representation of this function is shown in Fig. 5.3, where the surface defined by $H(p,q) = 0$ acts as the decision boundary separating the feasible and infeasible samples.

Once the feasible boundary has been learned, the next step is to obtain a function capable of estimating the performance score associated with a given parameter combination (p,q).

For this purpose, a Gaussian Radial Basis Function (GRBF) interpolation method was employed. The GRBF approximates the score surface by placing a Gaussian function centred at each sampled point and combining the contribution of all Gaussian functions to produce a continuous estimate of the score over the entire parameter space [34], [35], as illustrated in Fig. 5.4.

This continuous approximation allows the performance score to be evaluated at parameter combinations that were not explicitly simulated, enabling the optimization algorithm to search for high performance regions of the parameter space without requiring an exhaustive number of simulations.

The GRBF interpolation is given by

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}},$$

where

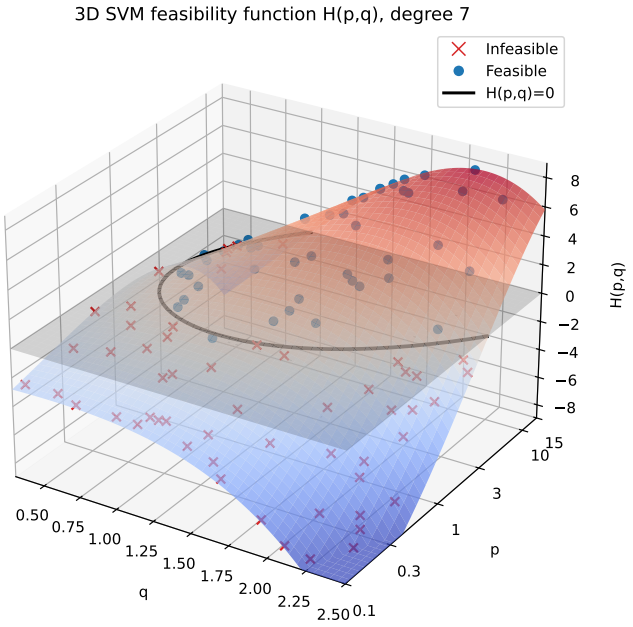


Figure 5.3: Learned SVM decision function $H(p,q)$ and the associated feasibility boundary.

- $\mathbf{x}$ denotes the point at which the function value is to be estimated;
- $\mathbf{x}_i$ represents the centre of the i-th Gaussian basis function, corresponding to a sampled point;
- w_i is the weight associated with the i-th Gaussian function;
- N denotes the total number of points used in the interpolation process;
- σ controls the width of the Gaussian functions and, consequently, the spatial influence of each sample point;
- $\|\mathbf{x}-\mathbf{x}_i\|^2$ represents the squared Euclidean distance between the point $\mathbf{x}$ and the centre $\mathbf{x}_i$.

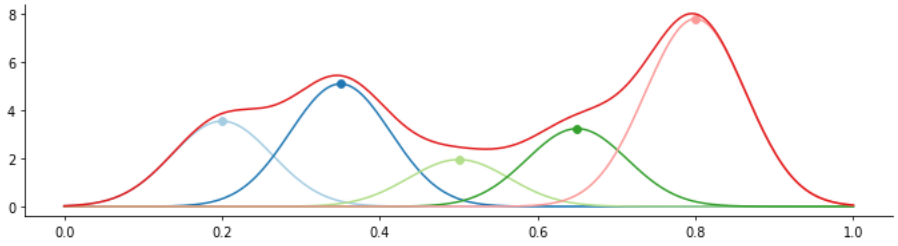


Figure 5.4: GRBF interpolation obtained from the sum of Gaussian basis functions [36].

It can be observed that the interpolation function contains a set of weights w_i. These weights are adjusted such that the interpolated surface passes exactly through the known sample points. Indeed, as illustrated in Fig. 5.4, simply summing the Gaussian functions is generally insufficient to satisfy this requirement.

To enforce interpolation, the function must satisfy the condition

$$f(\mathbf{x}_j) = y_j$$

for every known sample point $\mathbf{x}_j$.

Substituting the GRBF expression yields

$$\sum_{i=1}^{N} w_i e^{-\frac{\|\mathbf{x}_j - \mathbf{x}_i\|^2}{2\sigma^2}} = y_j,$$

which results in a linear system whose solution provides the weights w_i associated with each Gaussian basis function.

5.2 Results and Discussion

Having obtained all the necessary components to solve the optimization problem in (5.2), the ML algorithm was then used to automatically adjust the parameters in order to maximize the performance score, that was defined as

$$\begin{aligned} S = {} & 70\, r_{\text{prog}} + 100\, \mathbb{I}_{\text{completed}} - 3\, \Delta v_{\text{abs}} \\ & - 100 \max(0, -m_{\text{safety}})^2 - 200\, \overline{|e_{\text{cte}}|} - 80\, \mathbb{I}_{\text{collided}}, \end{aligned}$$

where r_{prog} denotes the progress ratio along the track, $\mathbb{I}_{\text{completed}}$ is equal to one if the vehicle completes the lap and zero otherwise, Δv_{abs} represents the accumulated absolute variation of the linear velocity, m_{safety} is the safety margin, $\overline{|e_{\text{cte}}|}$ is the mean absolute cross-track error and $\mathbb{I}_{\text{collided}}$ is equal to one if a collision occurs and zero otherwise. The safety margin is defined as

$$m_{\text{safety}} = \min\left(d_{\text{obs,min}}, d_{\text{track,min}}\right),$$

where $d_{\text{obs,min}}$ denotes the minimum clearance between the vehicle and any obstacle during the simulation, and $d_{\text{track,min}}$ denotes the minimum clearance between the vehicle and the track boundaries.

This score rewards lap completion and progress along the track, while penalizing unsafe behaviour, excessive velocity reductions, poor trajectory tracking and collisions.

The resulting score surface is shown in Fig. 5.5, where the star indicates the maximum score obtained from the simulations and the cross indicates the maximum score predicted by the optimization framework. The corresponding parameter values are $(5.97, 1.13)$ and $(6.76, 1.03)$, respectively.

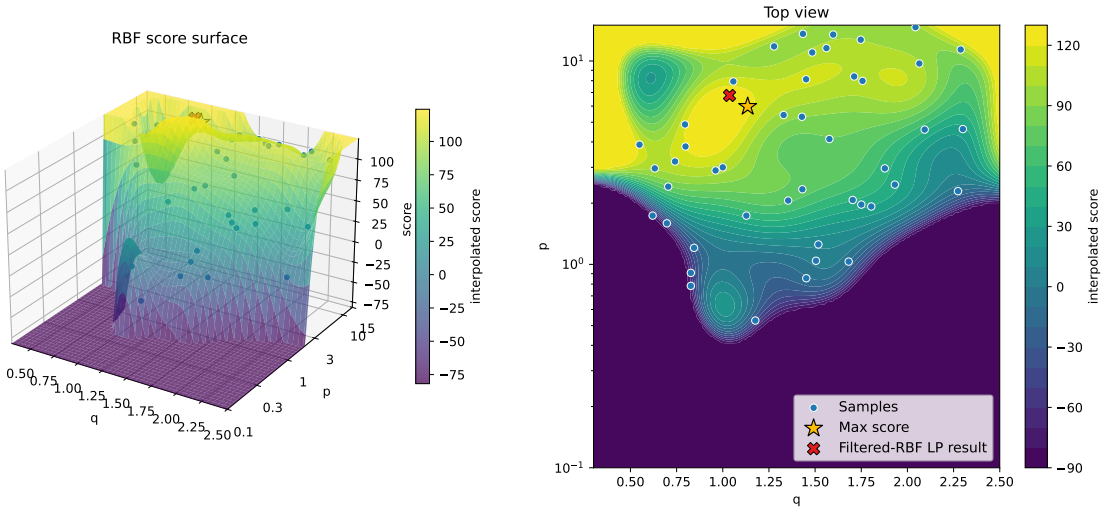


Figure 5.5: Estimated score surface obtained through GRBF interpolation. The star denotes the maximum simulated score, while the cross indicates the optimum predicted by the optimization framework.

Simulations were then performed using both the parameter values obtained through the LP optimization, $(6.76, 1.03)$, and the default values, $(3.0, 1.0)$. The former achieved a score of 134.44, whereas the latter obtained a score of 124.64.

The results are compared in Fig. 5.6, where the left and right columns correspond to the optimized and default parameter sets, respectively. Visually, the LP-optimized solution produces a smoother trajectory, with less aggressive deviations from the reference path.

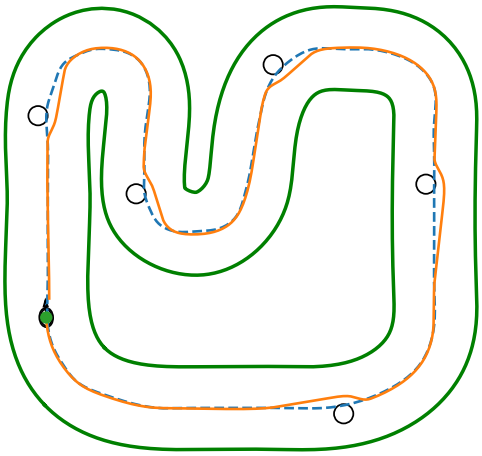
Furthermore, the lateral tracking error shown in Fig. 5.6(c) is lower than that observed in Fig. 5.6(d), indicating improved trajectory tracking performance. It can also be observed that the average vehicle speed is higher in the optimized case, suggesting that the vehicle is able to complete the lap in less time when using the parameters obtained through the optimization process.

However, it was observed that the maximum score obtained at the best simulation point $(5.97, 1.13)$ was 135.55, that is slightly higher than the value predicted by the model. Nevertheless, this behaviour is not entirely unexpected, since the score estimation is based on a discrete set of simulation samples.

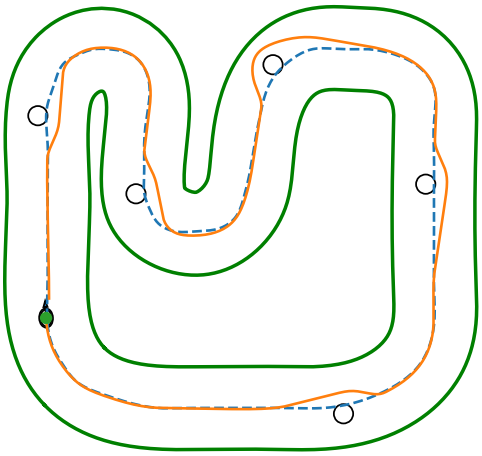
One possible way to improve the estimation accuracy would be to increase the total number of training samples, that is, to perform a larger number of simulations. However, this would also lead to a corresponding increase in computational cost.

Alternatively, a more efficient strategy consists of performing additional simulations only in the neighbourhood of the best point identified during the optimization process, using smaller intervals for the parameters p and q. This approach increases the density of samples around the region of interest, allowing the score estimation to be refined significantly without substantially increasing the overall computational effort.

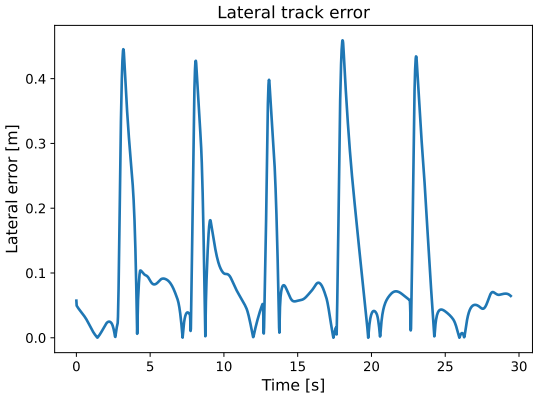
The resulting score surface is shown in Fig. 5.7. The highest score obtained from the simulations was 134.26 at the point $(5.22, 0.70)$, whereas the maximum score predicted by the model



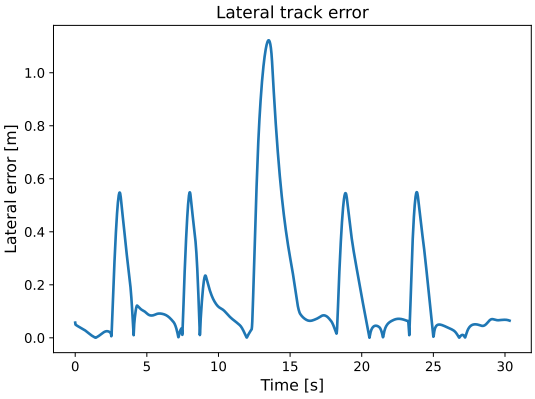
(a) Optimized parameters $(p,q) = (6.76, 1.03)$



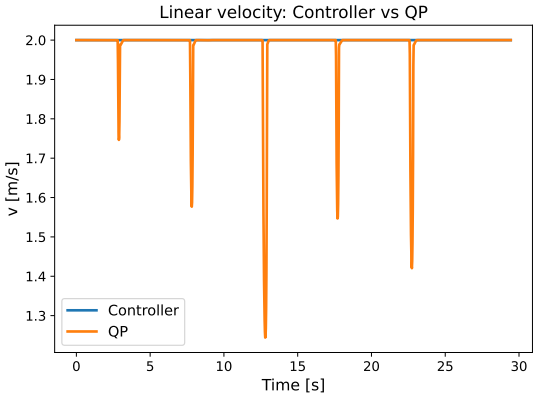
(b) Default parameters $(p,q) = (3.0, 1.0)$



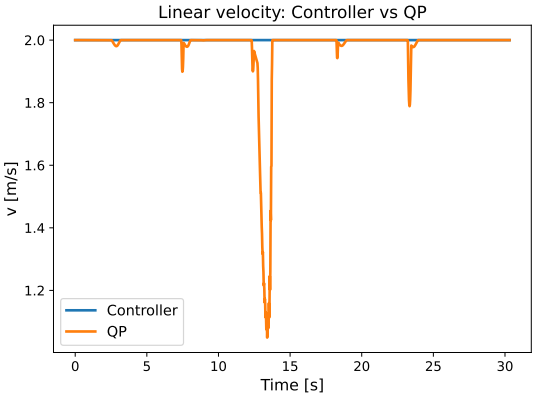
(c) Lateral tracking error



(d) Lateral tracking error



(e) Linear velocity



(f) Linear velocity

Figure 5.6: Comparison between the optimized parameter set and the default parameter set.

was 144.58 at $(5.55, 0.71)$.

A new simulation was therefore performed using the latter parameter values, yielding a score of 144.21. This result confirms that the local refinement procedure significantly improved the

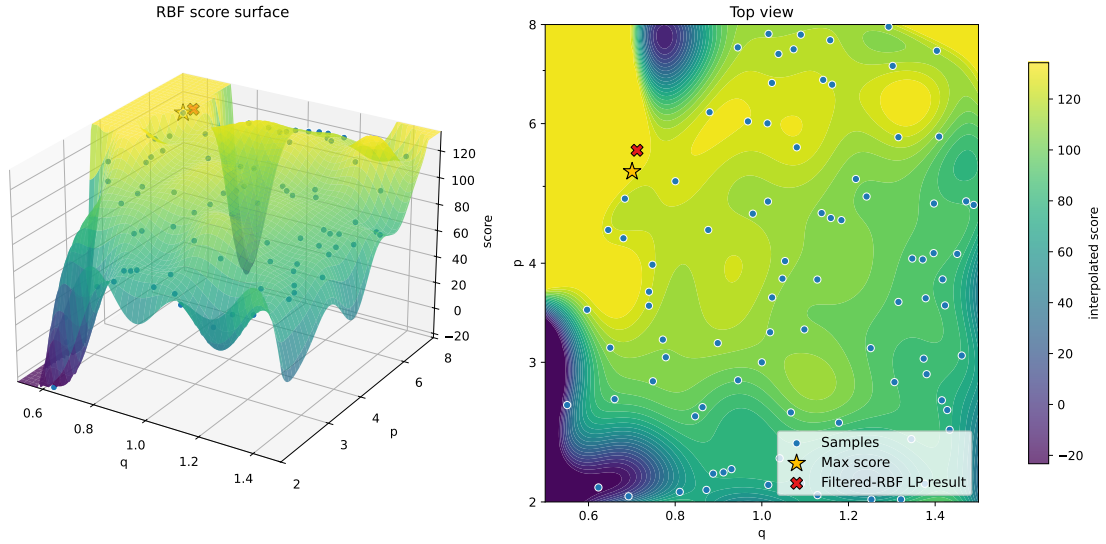


Figure 5.7: Refined score surface obtained through local sampling around the predicted optimum.

accuracy of the score estimation and allowed the optimization framework to identify a parameter combination that outperformed all previously evaluated samples.

The results demonstrate that the proposed optimization framework is capable of identifying parameter combinations that outperform those obtained through manual tuning. The optimized parameters produced smoother trajectories, lower lateral tracking errors, and higher average vehicle speeds, resulting in an overall improvement of the performance score.

Furthermore, the local refinement procedure significantly improved the accuracy of the GRBF approximation, allowing the optimization framework to locate a parameter combination that achieved a score of 144.21, surpassing all previously evaluated samples.

These results suggest that the proposed ML approach constitutes an effective tool for the offline tuning of CBF parameters. Rather than replacing the controller itself, the optimization framework complements the CBF-QP formulation by providing a systematic method for selecting parameters that would otherwise require extensive manual experimentation.

However, when the obstacle positions were modified and the simulation was repeated using the parameter values obtained through the machine learning approach, it was observed that the solution became infeasible for this new obstacle configuration. The vehicle became trapped while attempting to bypass the first obstacle, as shown in Fig. 5.8. This behaviour may indicate a certain degree of overfitting to the obstacle configuration used during the training process.

A new training process was therefore performed using the modified obstacle configuration. The maximum score obtained was 129.8 at the point $(5.26, 0.58)$. Simulations were then carried out using these parameter values. Although the resulting solution was feasible, it did not provide the best performance, as the vehicle still tended to avoid obstacles on the side that produced a larger lateral tracking error with respect to the desired path.

One of the most important factors influencing the side on which the vehicle chooses to bypass an obstacle is the QP cost function, which was defined as the quadratic difference between the lin-

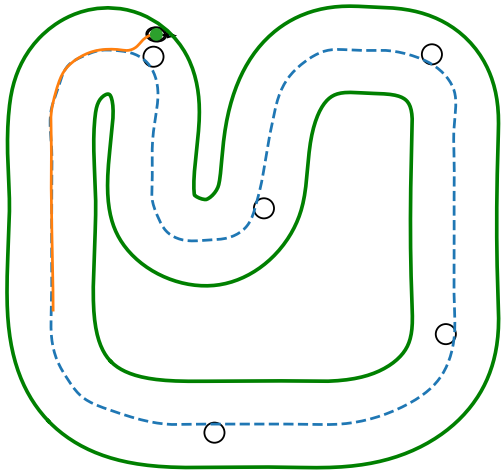


Figure 5.8: Vehicle becoming trapped when using parameters learned for a different obstacle configuration.

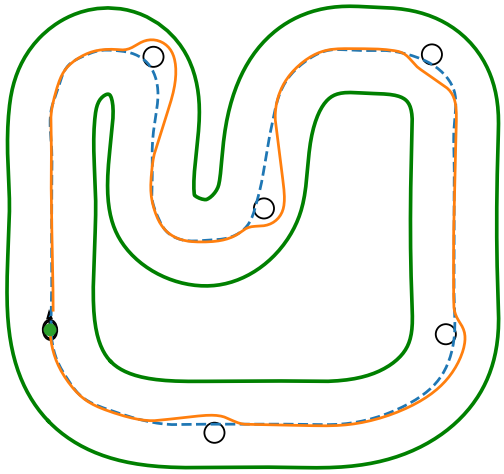


Figure 5.9: Trajectory obtained using the parameters learned for the new obstacle configuration.

ear velocity and a nominal linear velocity, and between the angular velocity and a nominal angular velocity (3.5). Since the nominal angular velocity was defined as zero, the vehicle naturally preferred manoeuvres that minimised the absolute value of the angular velocity, rather than selecting the side that would minimise the lateral tracking error.

A simple modification consists of replacing the nominal angular velocity of zero with the reference angular velocity of the trajectory, i.e., the angular velocity required for the vehicle to follow the desired path in the absence of obstacles. The resulting trajectory is shown in Fig. 5.10.

Furthermore, to reduce overfitting in the parameter learning process, the training procedure should ideally include variations in obstacle positions throughout the simulations. Additionally, improved performance could potentially be achieved by extending the learning process beyond

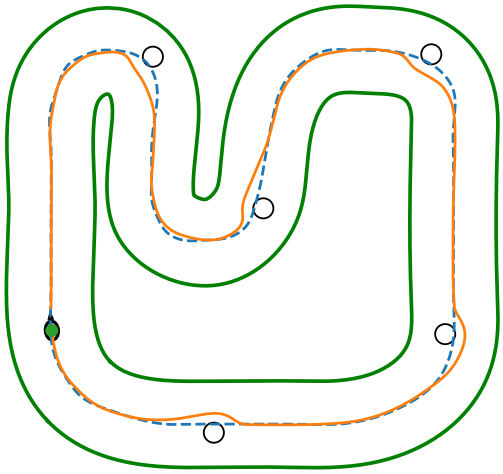


Figure 5.10: Trajectory obtained using the reference angular velocity as the nominal angular velocity in the QP cost function.

only two parameters and including other influential quantities, such as the QP weights and parameters associated with the Lyapunov based controller. However, these improvements would inevitably increase both the computational cost and the total training time, since a larger number of simulations would be required to adequately explore the expanded parameter space.

5.3 Chapter Conclusions

The results demonstrate that the proposed machine learning based framework is capable of identifying parameter combinations that improve the overall performance of the CBF based controller. By combining feasibility classification through an SVM with score estimation using GRBF interpolation, it was possible to efficiently explore the parameter space and predict promising operating regions without requiring exhaustive simulations.

The obtained results also provide insight into the influence of the class-$\mathscr{K}$ parameters p and q on the obstacle avoidance behaviour. These parameters directly affect the aggressiveness of the barrier constraints and therefore determine the trade-off between safety and trajectory tracking performance. Different parameter combinations lead to significantly different vehicle behaviours, ranging from conservative obstacle avoidance manoeuvres to more aggressive trajectories that remain closer to the reference path. The optimization process was therefore able to identify parameter values that provide a better balance between these competing objectives.

It should be noted that the machine learning framework does not learn how to control the vehicle directly. Instead, it learns the relationship between the barrier function parameters and the resulting controller performance. Consequently, the proposed approach should be interpreted as an automated parameter tuning methodology rather than a learning based control strategy.

However, the experiments also revealed an important limitation. The parameters learned using a specific obstacle configuration did not generalize well when the obstacle positions were modified. In some cases, the vehicle became trapped or failed to complete the track, despite achieving high scores during training. This behaviour suggests the presence of overfitting, since the learning process was based on simulations performed using a single obstacle arrangement. As a consequence, the learned parameters became specialized to that particular scenario and were unable to maintain the same level of performance under different environmental conditions.

Several strategies could be adopted to improve the robustness of the proposed approach. One possibility consists of increasing the number of training samples, allowing the score estimation model to better capture the relationship between the parameters and the resulting performance. Another potentially more effective strategy would be to generate training data using multiple obstacle configurations. By exposing the learning process to a wider range of scenarios, the resulting parameters would be expected to generalize better to previously unseen environments and reduce the risk of overfitting.

Finally, the results indicate that a local refinement stage can further improve the accuracy of the score estimation. After identifying a promising region in the parameter space, additional simulations can be performed using smaller parameter intervals around the predicted optimum. This increases the density of samples in the region of interest and allows a more precise approximation of the true optimum without significantly increasing the overall computational cost.

Chapter 6

Multi-Vehicle Safety Control

6.1 Static Obstacle Formulation

Without modifying the CBF formulations presented previously, it is possible to extend the proposed approach to scenarios involving multiple vehicles moving simultaneously on the track.

The main idea consists of treating the remaining vehicles as moving obstacles. However, during the resolution of the QP for a given vehicle, the positions of all other vehicles are assumed to be fixed. Consequently, from the perspective of that vehicle, the remaining vehicles can be modelled as static obstacles at that particular instant.

This approach allows the same barrier functions previously used for obstacle avoidance to be employed directly, requiring no modifications to the controller formulation.

6.1.1 Formulation

Consider two vehicles with positions (x_i, y_i) and (x_j, y_j). Following the lookahead-based formulation introduced previously, the safety constraint is applied to a point located ahead of vehicle i, defined as

$$l_{x,i} = x_i + l\cos\theta_i,$$
$$l_{y,i} = y_i + l\sin\theta_i.$$

The Euclidean distance between the lookahead point and vehicle j is given by

$$d_{ij} = \sqrt{(l_{x,i} - x_j)^2 + (l_{y,i} - y_j)^2}.$$

Defining a minimum safety distance d_{safe}, the collision avoidance condition becomes

$$d_{ij} \geq d_{\text{safe}}.$$

Using the same formulation adopted previously for obstacle avoidance, the corresponding barrier function is defined as

$$h_{ij}(\mathbf{x}) = (l_{x,i} - x_j)^2 + (l_{y,i} - y_j)^2 - d_{\text{safe}}^2.$$

It can be observed that this expression is identical to the one previously used for obstacle avoidance, with the only difference being that the obstacle centre is now replaced by the position of another vehicle.

Therefore, the corresponding CBF constraint is given by

$$L_f h_{ij}(\mathbf{x}) + L_g h_{ij}(\mathbf{x})u + \alpha(h_{ij}(\mathbf{x})) \geq 0.$$

When multiple vehicles are present, a barrier function is generated for each neighbouring vehicle and incorporated into the QP as an additional safety constraint.

Although the remaining vehicles are considered static during each optimization step, their positions are updated at every control cycle. Consequently, the generated barrier functions also change over time, allowing the controller to react continuously to the motion of the surrounding vehicles.

6.1.2 Simulation Results and Discussion

Several simulations were carried out in order to evaluate the behaviour of the controller in multi-vehicle scenarios. Figure 6.3 presents representative examples of interactions between two vehicles sharing the same track.

The obtained results show that the proposed methodology can successfully prevent collisions between vehicles while maintaining the path-following objective. Since each vehicle independently solves its own QP while considering the others as obstacles, no explicit communication or coordination mechanism is required.

However, there are some limitations in this implementation. Since the motion of neighbouring vehicles is not taken into account, the controller reacts only to their current positions. As a consequence, the generated behaviour may become conservative in situations where vehicles approach each other rapidly.

Nevertheless, the results demonstrate that the proposed CBF framework naturally extends to multi-vehicle scenarios without requiring any modifications to the original controller formulation. By modelling neighbouring vehicles as obstacles whose positions are updated at every control cycle, safe operation can be achieved while preserving the simplicity of the original approach.

6.2 Relative Velocity CBF Formulation

The formulation presented previously models neighbouring vehicles as static obstacles during each optimization step. Although their positions are updated at every control cycle, the barrier function derivative is computed assuming that the obstacle velocity is zero.

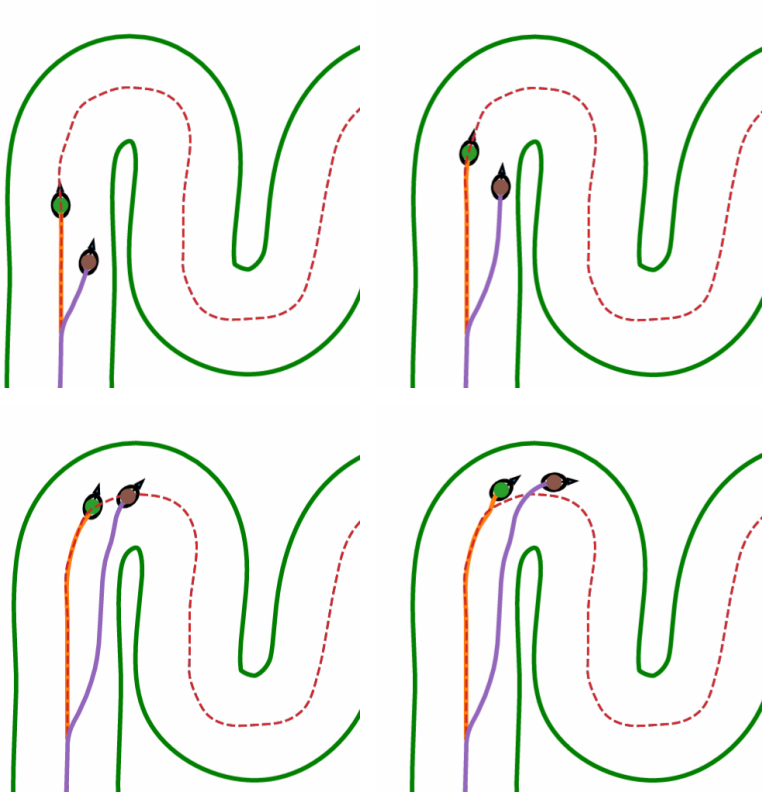


Figure 6.1: Evolution of the vehicle trajectories during the simulation using the static obstacle formulation.

6.2.1 Formulation

Consider the lookahead point of vehicle i,

$$l_{x,i} = x_i + l\cos\theta_i,$$
$$l_{y,i} = y_i + l\sin\theta_i.$$

and the position of a neighbouring vehicle,

$$o = \begin{bmatrix} x_j \\ y_j \end{bmatrix}.$$

Defining

$$p = \begin{bmatrix} l_{x,i} \\ l_{y,i} \end{bmatrix},$$

the barrier function can be written as

$$h(p,o) = \|p-o\|^2 - d_{\text{safe}}^2,$$

where d_{safe} denotes the minimum safety distance between the two vehicles.

In the formulation used previously, the neighbouring vehicle is assumed to remain stationary during the optimization step, i.e.,

$$\dot{o} = 0.$$

Consequently, the barrier derivative is given by

$$\dot{h} = 2(p-o)^T \dot{p},$$

leading to the standard CBF condition

$$\dot{h} + \alpha(h) \geq 0.$$

However, when both vehicles are moving, the obstacle position becomes time-varying,

$$o = o(t),$$

with velocity

$$\dot{o} = \begin{bmatrix} v_j \cos\theta_j \\ v_j \sin\theta_j \end{bmatrix},$$

where v_j and θ_j denote the velocity and heading angle of vehicle j, respectively.

The barrier derivative must then be computed using the relative velocity between the two vehicles:

$$\dot{h} = 2(p-o)^T (\dot{p} - \dot{o}).$$

The velocity of the lookahead point is given by

$$\dot{p} = \begin{bmatrix} v_i \cos\theta_i - l\omega_i \sin\theta_i \\ v_i \sin\theta_i + l\omega_i \cos\theta_i \end{bmatrix}.$$

Substituting the vehicle kinematics yields

$$\dot{h} = a_v v_i + a_\omega \omega_i - 2(p-o)^T \dot{o},$$

where

$$a_v = 2\left[(l_{x,i} - x_j)\cos\theta_i + (l_{y,i} - y_j)\sin\theta_i\right]$$

and

$$a_\omega = 2l\left[-(l_{x,i} - x_j)\sin\theta_i + (l_{y,i} - y_j)\cos\theta_i\right].$$

The resulting CBF condition becomes

$$a_v v_i + a_\omega \omega_i - 2(p-o)^T \dot{o} + \alpha(h) \geq 0.$$

Compared with the previous formulation, the only additional term is

$$-2(p-o)^T \dot{o},$$

which explicitly accounts for the motion of the neighbouring vehicle. When the vehicles move towards each other, this term makes the safety constraint more restrictive, causing the controller to react earlier. Conversely, when the vehicles move away from each other, the constraint becomes less restrictive.

6.2.2 Simulation Results and Discussion

As can be seen in Fig. 6.2, the previously discussed behaviour is confirmed. The follower vehicle reacts later to the leader vehicle than in Fig. 6.1, since the predicted relative motion indicates that the distance between them will increase during the next time instant.

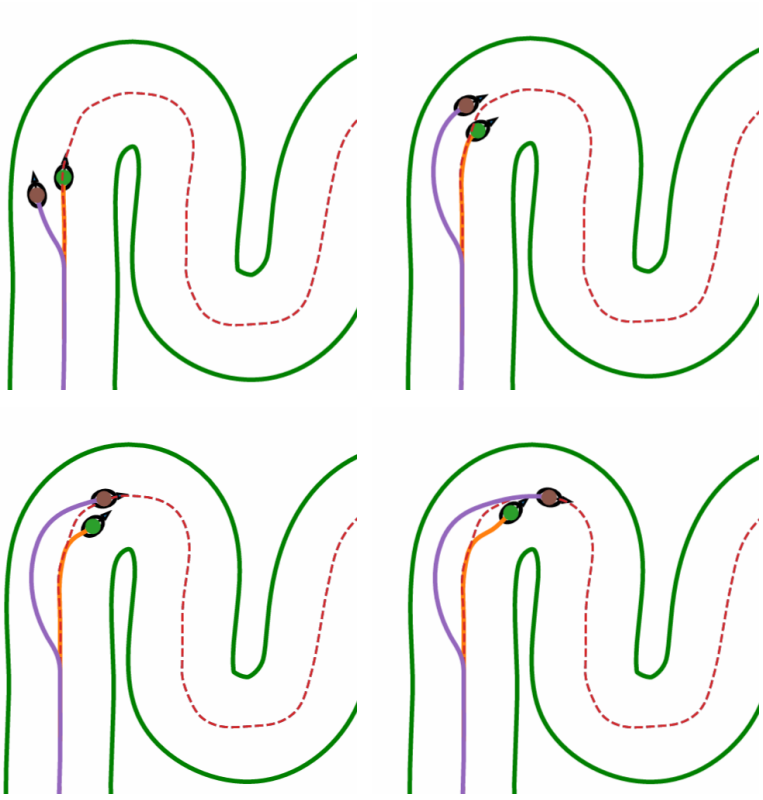


Figure 6.2: Evolution of the vehicle trajectories during the simulation using the relative velocity formulation.

Therefore, unlike the Static Obstacle formulation, which only considers the instantaneous positions of the neighbouring vehicles, this approach also incorporates their velocities through the relative-motion term, providing a more accurate representation of multi-vehicle interactions.

Furthermore, static obstacles can also be added to the track, allowing the simultaneous evaluation of obstacle avoidance with respect to both static and dynamic obstacles, as illustrated in Fig. 6.3.

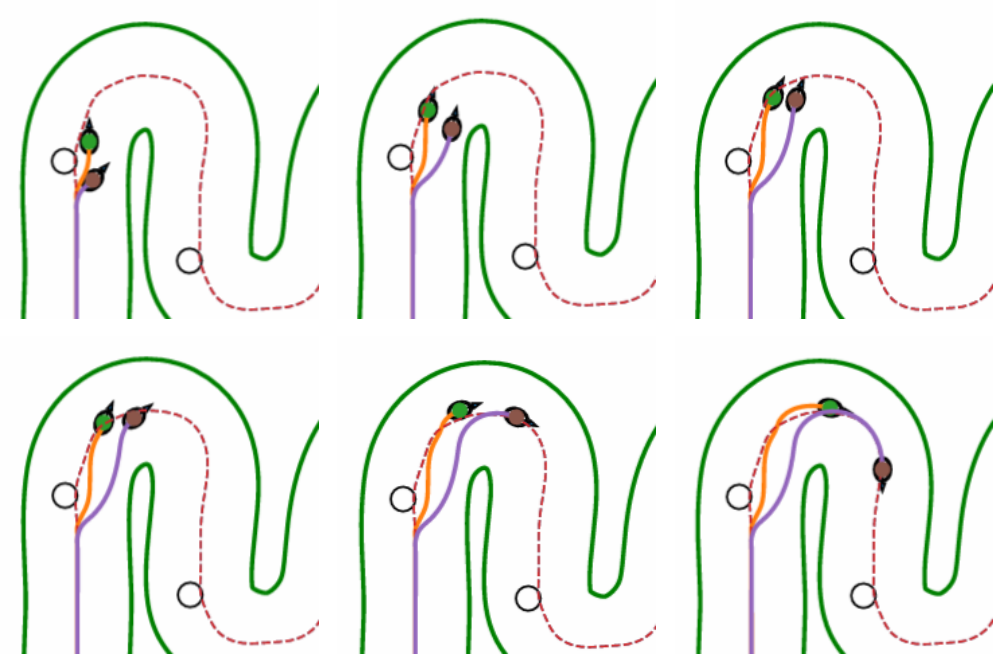


Figure 6.3: Examples of multi-vehicle interactions using the proposed CBF-based controller.

As the number of vehicles increases, the controller must simultaneously satisfy multiple safety constraints, one for each neighbouring vehicle. Since the proposed formulation incorporates the relative velocity of each vehicle, the controller can better distinguish between potentially dangerous interactions and situations in which neighbouring vehicles are naturally moving away from each other.

This reduces unnecessary evasive manoeuvres and allows the vehicles to maintain smoother trajectories while still guaranteeing safety. In particular, when two vehicles travel in similar directions with comparable velocities, the relative-motion term reduces the conservativeness of the barrier constraints, preventing excessive deviations from the reference path.

The results shown in Fig. 6.3 demonstrate that the proposed formulation can handle both static obstacles and dynamic vehicle interactions simultaneously. Throughout the simulation, the vehicles successfully avoid collisions while preserving stable path-following behaviour, indicating that the controller scales naturally to more complex traffic scenarios involving multiple moving agents.

6.3 Chapter Conclusions

This chapter extended the proposed safety critical control framework to multi vehicle scenarios, where neighbouring vehicles were treated as dynamic obstacles. Two different formulations were

investigated.

The first approach considered neighbouring vehicles as static obstacles during each control update. Although this formulation was able to guarantee collision avoidance and maintain safe vehicle interactions, it did not explicitly account for the future motion of surrounding vehicles. Consequently, the controller reacted only to the current positions of neighbouring vehicles, often resulting in conservative avoidance manoeuvres.

To address this limitation, a second formulation based on relative velocity information was introduced. By incorporating the relative motion between vehicles directly into the Control Barrier Function constraint, the controller became capable of distinguishing between approaching and receding vehicles. As a result, safety constraints became more restrictive when the distance between vehicles was decreasing and less restrictive when vehicles were moving apart.

Simulation results demonstrated that both formulations successfully prevented collisions and maintained the forward invariance of the safe set. However, the relative velocity formulation produced smoother and less conservative interactions, allowing the vehicle to better exploit the available free space while preserving safety guarantees. Furthermore, the results highlighted the importance of considering obstacle motion in dynamic environments, particularly when multiple autonomous vehicles share the same workspace.

Overall, the proposed CBF framework proved capable of extending the safety guarantees obtained in single vehicle obstacle avoidance scenarios to multi vehicle environments. The incorporation of relative velocity information represents an effective strategy for improving interaction quality while maintaining the formal safety properties of the controller. These results provide a foundation for the experimental validation presented in the following chapter.

Chapter 7

Real Vehicle Implementation

7.1 Experimental Methodology

7.1.1 State Estimation

As described previously, the vehicle state was estimated using an overhead camera combined with computer vision algorithms developed in OpenCV. The perception system was responsible for determining the vehicle's position and orientation in real time and transmitting this information to the controller.

Vehicle detection was performed through colour segmentation in the HSV colour space, Fig. 7.1. After image acquisition, a colour mask was applied to isolate the marker attached to the vehicle. The centroid of the detected region was then used to estimate the vehicle position in image coordinates.

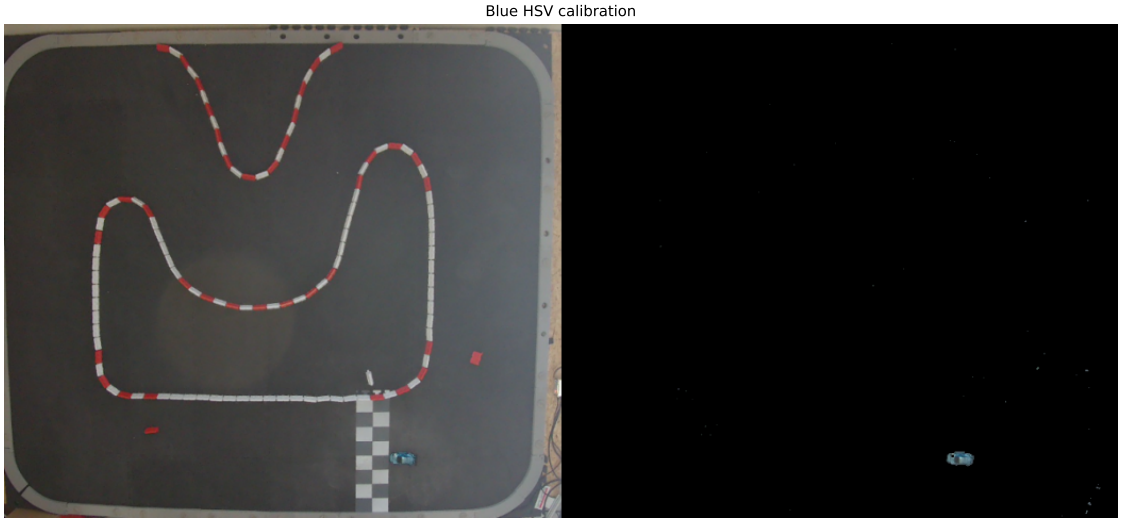


Figure 7.1: HSV calibration interface used for colour segmentation tuning.

To convert pixel coordinates into metric coordinates, a calibration factor was obtained experimentally by measuring the displacement of the vehicle in both the image and the real world. This

calibration allowed the controller to operate directly in metric units.

The vehicle orientation was estimated using consecutive position measurements. Given two successive position estimates, the orientation angle was computed from the direction of the displacement vector, Fig. 7.2. Although this approach is simple to implement, it becomes less precise at higher vehicle velocities.

$$\theta = \text{atan2}(y_t - y_{t-1}, x_t - x_{t-1})$$

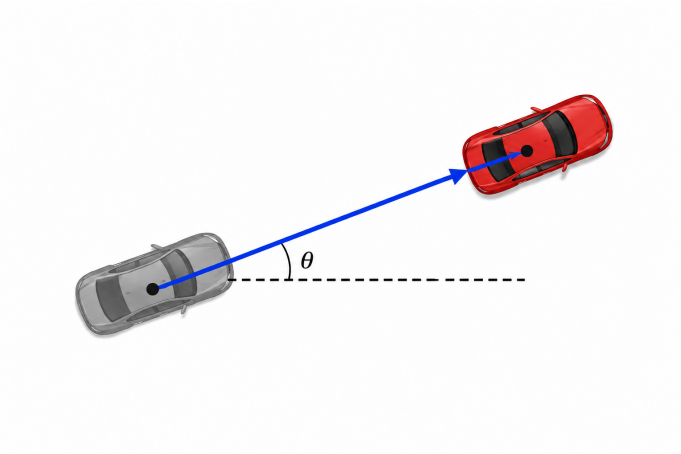


Figure 7.2: Vehicle orientation estimation based on consecutive position measurements.

7.1.2 Communication Framework

The developed system relies on two communication channels connecting the perception, control, and actuation modules.

First, the estimated vehicle state generated by the perception system is transmitted to the controller through a UDP connection. This communication method was selected due to its low latency and simplicity, making it suitable for real time control applications.

After receiving the vehicle state, the controller computes the desired linear and angular velocity commands using the proposed CLF-CBF-QP framework. These commands are then transmitted to the vehicle through a BLE connection.

Onboard the vehicle, the ESP32-C3 microcontroller receives the commands and converts them into steering and propulsion signals that are applied to the servo motor and DC motor driver, respectively.

The resulting information flow can be summarized as follows:

$$\text{Camera} \rightarrow \text{Perception} \rightarrow \text{Controller} \rightarrow \text{Vehicle}.$$

7.1.3 Test Scenarios

Several experiments were conducted to evaluate the performance of the proposed controller under real world conditions.

The first experiment focused on trajectory tracking performance. In this scenario, the vehicle was required to follow the reference path in the absence of obstacles, allowing the effectiveness of the tracking controller to be evaluated independently of the obstacle avoidance component.

The second experiment evaluated obstacle avoidance performance. Static obstacles were placed along the track, requiring the controller to simultaneously satisfy the trajectory tracking and safety objectives imposed by the CLF-CBF-QP formulation.

Finally, computational performance measurements were performed to evaluate the real time feasibility of the proposed architecture. The execution times associated with image acquisition, image processing, communication, optimization, and command transmission were measured and analysed.

These experiments allowed both the effectiveness of the proposed control framework and the practical limitations of the experimental platform to be assessed.

7.2 Experimental Results and Discussion

7.2.1 Trajectory Tracking

The first set of experiments was then conducted to evaluate the trajectory tracking capabilities of the proposed controller. Figure 7.3 presents the trajectory followed by the vehicle during the experiment.

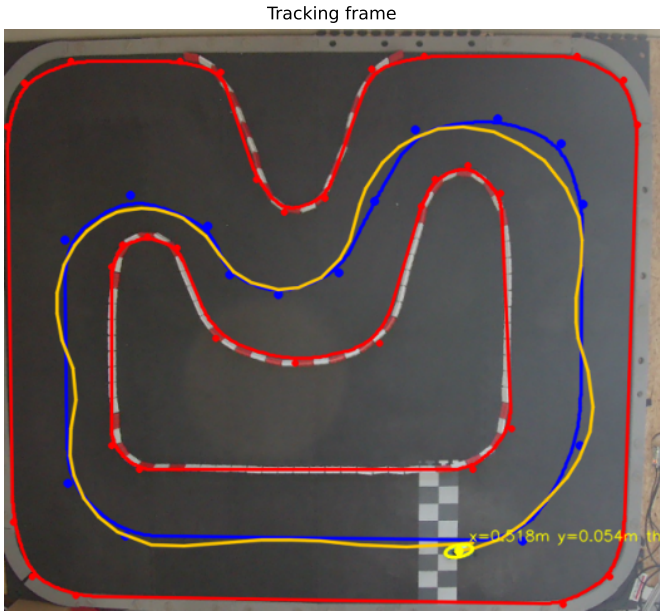


Figure 7.3: Trajectory tracking experiment using the CLF-CBF-QP controller.

The obtained results demonstrate that the vehicle was able to successfully follow the reference path while maintaining smooth and controlled motion throughout the track. Although the overall trajectory closely matches the desired path, small deviations can be observed in some sections of the track.

These deviations are primarily attributed to perception noise, communication delays, actuator limitations, and modelling inaccuracies that are not present in simulation. In particular, the orientation estimation method based on consecutive position measurements introduces additional noise and delay, which directly affects the steering commands generated by the controller.

Nevertheless, the controller remained capable of correcting the resulting tracking errors and maintaining the vehicle within the desired path boundaries, demonstrating the effectiveness of the proposed CLF-CBF-QP framework under real world conditions.

7.2.2 Obstacle Avoidance

The obstacle avoidance experiments were performed by placing static obstacles along the track and requiring the vehicle to simultaneously satisfy both trajectory tracking and safety objectives.

Figure 7.4 presents some examples of obstacle avoidance manoeuvres executed by the vehicle. The results show that the controller was capable of successfully avoiding the obstacles while continuing to progress along the track.

The observed behaviour is consistent with the simulation results presented in Chapter 4. As the vehicle approaches an obstacle, the barrier constraints become active and modify the nominal control commands. Consequently, the vehicle temporarily deviates from the reference trajectory in order to maintain a safe distance from the obstacle before gradually returning to the desired path.

Although the overall behaviour closely matches the simulation results, for some obstacle positions the vehicle is unable to deviate successfully and occasionally collides with the track boundaries, Fig. 7.5 . This occurs mainly due to the limited maximum steering angle of the vehicle, combined with the perception, control and communication delays.

However, this scenario also highlights the controller's ability to enforce the safety constraints associated with the track boundaries. As can be seen in Fig. 7.5, the vehicle stopped as a result of the simultaneous activation of the boundary constraints, demonstrating that safety is maintained even when no feasible motion satisfying all constraints exists.

Overall, the experiments confirm that the proposed safety critical control framework can successfully transfer from simulation to a real vehicle platform while preserving its collision avoidance capabilities.

7.2.3 Computational Performance

To evaluate the real time feasibility of the proposed framework, the execution times of the main software components were measured during the experiments. The obtained results are summarized in Table 7.1.

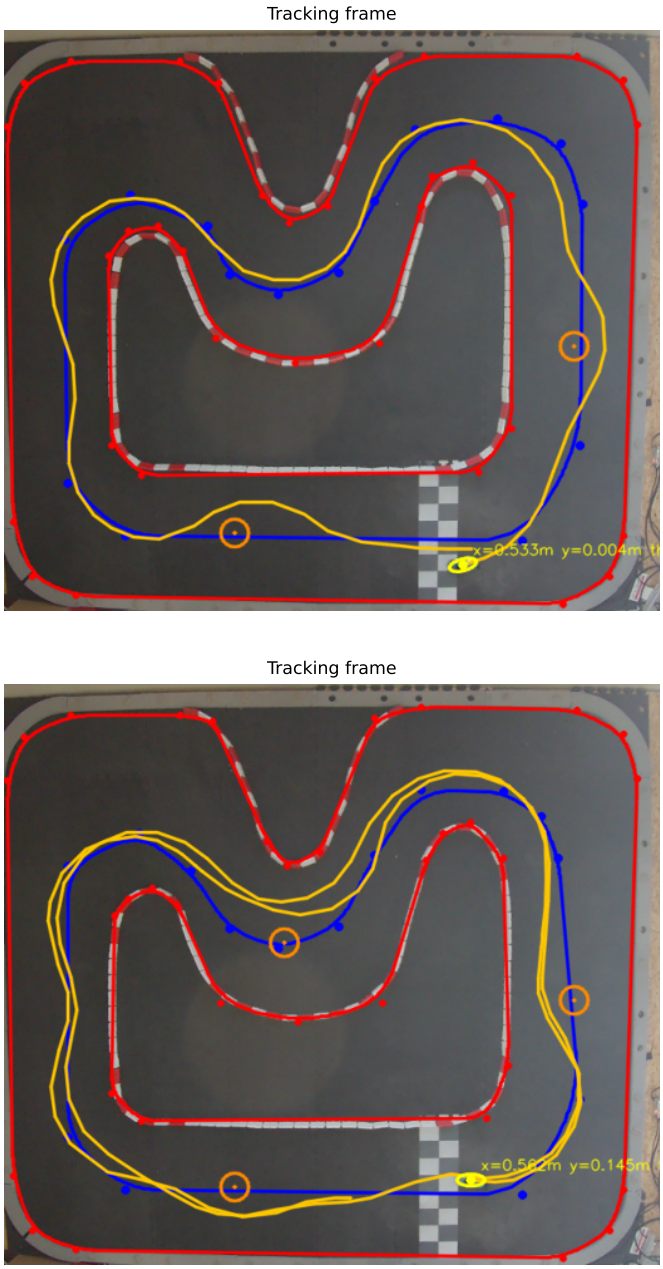


Figure 7.4: Real world obstacle avoidance experiments using the proposed CBF-based controller.

Table 7.1: Average execution times of the experimental platform.

Operation	Average Time (ms)
Image acquisition	6.76
Image processing	27.39
State acquisition (UDP)	28.29
QP solution	7.49
BLE communication	4.34
Total control process loop	40

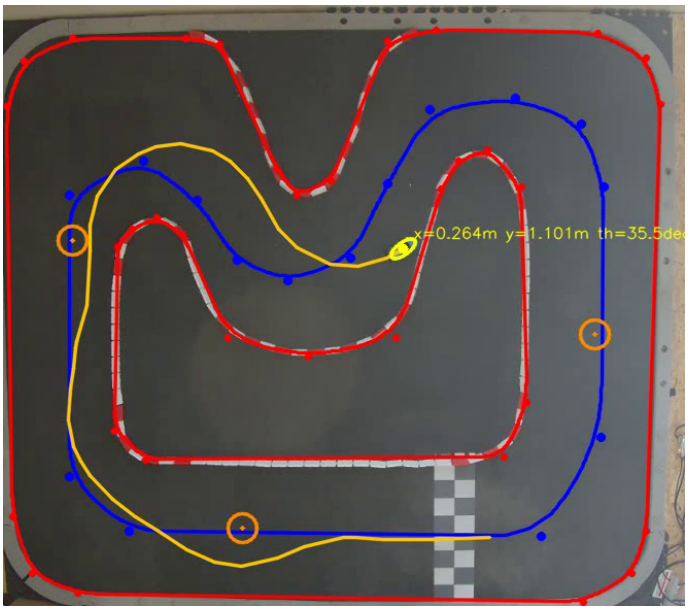


Figure 7.5: Vehicle stopped due to simultaneous activation of track boundary safety constraints.

The largest computational cost is associated with the perception module, particularly image processing operations required for colour segmentation and state estimation. In contrast, the execution time of the CLF-CBF-QP controller remains relatively small, demonstrating that the optimization problem can be solved efficiently in real time.

The average execution time of the complete control loop was approximately 40 ms, corresponding to an operating frequency of approximately 25 Hz. Furthermore, considering the delays associated with image acquisition and communication, the average age of the state information available to the controller was approximately 62.44 ms.

These results indicate that the proposed architecture is capable of supporting real time autonomous vehicle control. Moreover, the relatively low computational cost of the optimization stage suggests that more advanced safety critical control formulations could be implemented without compromising real time operation.

7.3 Sources of Experimental Error

Although the experimental results demonstrate the feasibility of the proposed approach, several sources of error may affect the overall system performance.

First, the perception system introduces delays associated with image acquisition, processing, and state estimation. Since the vehicle position is obtained from a camera operating at a finite frame rate, the controller always acts on slightly outdated information. This delay becomes more significant as the vehicle speed increases.

The vision state estimation is also affected by segmentation inaccuracies. The HSV thresholds used to detect the vehicle may be influenced by illumination variations, reflections, and image noise, leading to small errors in the estimated position. Since the vehicle orientation is computed

from consecutive position measurements, these errors may be amplified when the displacement between frames is big.

Occasional frame losses or delayed image processing may further increase the age of the state estimate available to the controller. Although such events were infrequent during the experiments, they may introduce temporary degradation in the control performance.

Communication between the computer and the vehicle is performed through BLE, which introduces additional latency between the computation of the control action and its execution by the vehicle. While this latency was sufficiently small for the operating conditions considered, it still contributes to the overall system delay.

The steering actuator also introduces non idealities. The commanded steering angle is converted into discrete servo positions, resulting in a quantization effect that limits the steering resolution. Furthermore, the actuator dynamics are not explicitly modelled and may differ from the ideal instantaneous response assumed in simulation.

At the vehicle level, wheel slip and variations in tire-ground interaction may cause deviations from the ideal kinematic behaviour. These effects become particularly relevant during sharp turns and rapid manoeuvres.

Another factor that may influence the vehicle behaviour is the battery charge level. As the battery voltage decreases during operation, the maximum torque and speed that can be delivered by the propulsion system may also change. Consequently, the same control commands may produce slightly different vehicle responses depending on the battery state, introducing additional variability that is not captured by the simulation model.

Finally, the controller is based on a kinematic bicycle model, which represents a simplified description of the real vehicle dynamics. Factors such as actuator dynamics, wheel slip, mechanical tolerances, and communication delays are not explicitly included in the model. Consequently, a discrepancy between simulation and experimental results is expected.

Despite these limitations, the proposed controller was able to successfully perform trajectory tracking and obstacle avoidance in the real vehicle experiments, demonstrating a satisfactory level of robustness under realistic operating conditions.

7.4 Chapter Conclusions

This chapter presented the implementation and experimental validation of the proposed safety critical control framework on a small scale autonomous vehicle platform.

A complete experimental system was developed, integrating computer vision based state estimation, wireless communication, onboard actuation, and the proposed CLF-CBF-QP controller. This platform enabled the evaluation of the control framework under realistic operating conditions, including sensing delays, communication latencies, actuator limitations, and modelling inaccuracies.

The experimental results demonstrated that the vehicle was capable of successfully tracking the reference trajectory while avoiding both static obstacles and track boundaries. The observed

behaviour was consistent with the simulation results presented in previous chapters, confirming the ability of the proposed controller to transfer from simulation to a real world implementation.

The computational performance analysis showed that the complete control loop could be executed in real time, achieving an average operating frequency of approximately 25 Hz. Furthermore, the relatively low computational cost of the optimization stage confirms the practicality of applying CBF-based safety critical control methods to autonomous vehicle systems.

Several sources of experimental error were identified, including perception noise, communication delays, actuator nonlinearities, wheel slip, and modelling simplifications. Although these factors introduced deviations from the simulated behaviour, the controller was still able to maintain safe operation and achieve the desired objectives.

Overall, the experimental validation confirmed the effectiveness, robustness, and practical applicability of the proposed control framework. The results demonstrate that the developed CBF-based methodology can successfully provide safe trajectory tracking and obstacle avoidance on a real autonomous vehicle platform.

Chapter 8

Conclusions and Future Work

This dissertation investigated safety critical control strategies for autonomous racing vehicles based on CBFs. The main objective was to develop and validate control methodologies capable of guaranteeing collision avoidance while maintaining satisfactory trajectory tracking performance in both simulated and real world environments.

Initially, several safety control formulations were studied, including conventional CBFs, HOCBFs, and iHOCBFs. Although the iHOCBF formulation allowed both vehicle control inputs to be incorporated into the safety constraint, it introduced additional complexity due to the higher order derivatives and auxiliary dynamics required. To overcome these limitations, a lookahead based barrier formulation was proposed, enabling a simpler implementation while maintaining effective obstacle avoidance capabilities.

Subsequently, the proposed framework was extended through the integration of CLFs within a QP formulation. This approach allowed safety and trajectory tracking objectives to be addressed simultaneously, resulting in a controller capable of modifying the nominal control actions only when required to satisfy safety constraints.

To reduce the dependence on manual parameter tuning, a ML based optimization methodology was developed. Using simulation data, feasible and infeasible parameter regions were identified through SVM classification, while GRBF interpolation was employed to estimate controller performance throughout the parameter space. The obtained results demonstrated that the proposed methodology can successfully improve controller performance while significantly reducing the tuning effort.

The developed framework was further extended to multi vehicle scenarios. Two formulations were investigated, namely a static obstacle approach and a relative velocity formulation. The results demonstrated that incorporating relative motion information reduces conservativeness and improves vehicle interactions while preserving the safety guarantees provided by the CBF framework.

Finally, the proposed controller was experimentally validated using a small scale autonomous vehicle platform equipped with computer vision state estimation and wireless communication. The experimental results confirmed the ability of the controller to perform trajectory tracking and

obstacle avoidance under real world conditions. Although performance degradation was observed when compared with simulation results, mainly due to perception noise, communication delays, actuator limitations, and modelling inaccuracies, the controller consistently maintained safe operation and demonstrated robustness to these practical challenges.

Overall, the results obtained throughout this dissertation demonstrate that CBFs constitute an effective and flexible framework for safety critical autonomous vehicle control. The proposed methodologies successfully combined safety guarantees, trajectory tracking, automatic parameter optimization, and multi vehicle interaction capabilities within a unified control architecture.

Future work may focus on several directions. First, improvements to the perception system could be investigated in order to increase state estimation accuracy and reduce the impact of measurement noise. In particular, the incorporation of onboard sensors such as an Inertial Measurement Unit (IMU) combined with sensor fusion techniques could provide more accurate and robust estimates of vehicle orientation and velocity, reducing the dependence on vision based measurements alone. Second, more advanced vehicle models could be considered in order to reduce the simulation to reality gap and improve controller performance at higher speeds. Third, the ML optimization framework could be extended to simultaneously optimize additional controller parameters, including QP weights and CLF gains, while considering multiple track and obstacle configurations to improve generalization. Fourth, the multi vehicle formulation could be enhanced through the incorporation of motion prediction techniques, allowing the controller to anticipate future vehicle behaviour rather than reacting only to current relative states. Finally, the proposed framework could be integrated with higher level planning algorithms and validated on larger and faster autonomous vehicle platforms operating in more complex environments.

These directions constitute promising opportunities for further improving the performance, robustness, and applicability of safety critical control methods for autonomous driving systems.

References

[1] U. Nations, *Sustainable Development*, en. Accessed: Jun. 25, 2026. [Online]. Available: https://www.un.org/en/teach/SDGs.

[2] M. Guériau and I. Dusparic, "Quantifying the impact of connected and autonomous vehicles on traffic efficiency and safety in mixed traffic," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, Sep. 2020, pp. 1–8. DOI: 10.1109/ITSC45102.2020.9294174. Accessed: Apr. 18, 2026. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9294174.

[3] R. E. Stern et al., "Quantifying air quality benefits resulting from few autonomous vehicles stabilizing traffic," *Transportation Research Part D: Transport and Environment*, vol. 67, pp. 351–365, Feb. 2019, ISSN: 1361-9209. DOI: 10.1016/j.trd.2018.12.008. Accessed: Apr. 18, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361920918304383.

[4] L. Ye and T. Yamamoto, "Evaluating the impact of connected and autonomous vehicles on traffic safety," *Physica A: Statistical Mechanics and its Applications*, vol. 526, p. 121 009, Jul. 2019, ISSN: 0378-4371. DOI: 10.1016/j.physa.2019.04.245. Accessed: Apr. 18, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437119306181.

[5] Y. Huang and Z. Yang, *Autonomous Driving Handbook*, en. Singapore: Springer Nature, 2026, ISBN: 9789819537242 9789819537259. DOI: 10.1007/978-981-95-3725-9. Accessed: Jun. 19, 2026. [Online]. Available: https://link.springer.com/10.1007/978-981-95-3725-9.

[6] J. Betz et al., "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022, ISSN: 2687-7813. DOI: 10.1109/OJITS.2022.3181510. Accessed: Dec. 23, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9790832.

[7] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," en, in *53rd IEEE Conference on Decision and Control*, Los Angeles, CA, USA: IEEE, Dec. 2014, pp. 6271–6278, ISBN: 978-1-4673-6090-6 978-1-4799-7746-8 978-1-4799-7745-1. DOI: 10.1109/CDC.2014.7040372. Accessed: Jan. 7, 2026. [Online]. Available: http://ieeexplore.ieee.org/document/7040372/.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control Barrier Function Based Quadratic Programs for Safety Critical Systems," en, *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017, ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2016.2638961. Accessed: Nov. 1, 2025. [Online]. Available: http://ieeexplore.ieee.org/document/7782377/.

[9] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," in *2019 18th European Control Conference (ECC)*, Jun. 2019, pp. 3420–3431. DOI: 10.23919/ECC.2019.8796030. Accessed: Nov. 3, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8796030.

[10] W. Xiao, C. G. Cassandras, and C. Belta, *Safe Autonomy with Control Barrier Functions: Theory and Applications* (Synthesis Lectures on Computer Science), en. Cham: Springer International Publishing, 2023, ISBN: 978-3-031-27575-3 978-3-031-27576-0. DOI: 10.1007/978-3-031-27576-0. Accessed: Nov. 1, 2025. [Online]. Available: https://link.springer.com/10.1007/978-3-031-27576-0.

[11] A. Morris, "On the Migration of Risks and Liabilities for Increased Automation," Jan. 2020. DOI: 10.2514/6.2020-0454.

[12] S. D. Pendleton et al., "Perception, Planning, Control, and Coordination for Autonomous Vehicles," en, *Machines*, vol. 5, no. 1, Feb. 2017, Company: Multidisciplinary Digital Publishing Institute Distributor: Multidisciplinary Digital Publishing Institute Institution: Multidisciplinary Digital Publishing Institute Label: Multidisciplinary Digital Publishing Institute, ISSN: 2075-1702. DOI: 10.3390/machines5010006. Accessed: Dec. 24, 2025. [Online]. Available: https://www.mdpi.com/2075-1702/5/1/6.

[13] A. Jiang, "Research on the Development of Autonomous Race Cars And Impact on Self-driving Cars," *Journal of Physics: Conference Series*, vol. 1824, p. 012 009, Mar. 2021. DOI: 10.1088/1742-6596/1824/1/012009.

[14] N. AbuJabal et al., "A Comprehensive Study of Recent Path-Planning Techniques in Dynamic Environments for Autonomous Robots," en, *Sensors*, vol. 24, no. 24, Dec. 2024, Company: Multidisciplinary Digital Publishing Institute Distributor: Multidisciplinary Digital Publishing Institute Institution: Multidisciplinary Digital Publishing Institute Label: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s24248089. Accessed: Dec. 26, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/24/24/8089.

[15] A. Bemporad, "Model Predictive Control Design: New Trends and Tools," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec. 2006, pp. 6678–6683. DOI: 10.1109/CDC.2006.377490. Accessed: Dec. 27, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/4178103.

[16] G. V. Raffo, G. K. Gomes, J. E. Normey-Rico, C. R. Kelber, and L. B. Becker, "A Predictive Controller for Autonomous Vehicle Path Tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 92–102, Mar. 2009, ISSN: 1558-0016. DOI: 10.1109/TITS.2008.2011697. Accessed: Dec. 27, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/4768725.

[17] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, Feb. 2017, ISSN: 1939-9359. DOI: 10.1109/TVT.2016.2555853. Accessed: Dec. 27, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/7458179.

[18] M. Arnold, R. Negenborn, G. Andersson, and B. De Schutter, "Multi-Area Predictive Control for Combined Electricity and Natural Gas Systems," *2009 European Control Conference, ECC 2009*, Mar. 2015.

[19] *Understanding Model Predictive Control*, en. Accessed: Dec. 31, 2025. [Online]. Available: https://www.mathworks.com/videos/series/understanding-model-predictive-control.html.

[20] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation Proceedings*, vol. 2, Mar. 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247. Accessed: Dec. 30, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/1087247/.

[21] W. Siming, Z. Tiantian, and L. Weijie, "Mobile Robot Path Planning Based on Improved Artificial Potential Field Method," in *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*, Aug. 2018, pp. 29–33. DOI: 10.1109/IRCE.2018.8492951. Accessed: Dec. 30, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8492951.

[22] S. M. H. Rostami, A. K. Sangaiah, J. Wang, and X. Liu, "Obstacle avoidance of mobile robots using modified artificial potential field algorithm," en, *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 70, Mar. 2019, ISSN: 1687-1499. DOI: 10.1186/s13638-019-1396-2. Accessed: Dec. 30, 2025. [Online]. Available: https://doi.org/10.1186/s13638-019-1396-2.

[23] S. Campbell, W. Naeem, and G. Irwin, "A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres," en, *Annual Reviews in Control*, vol. 36, no. 2, pp. 267–283, Dec. 2012, ISSN: 13675788. DOI: 10.1016/j.arcontrol.2012.09.008. Accessed: Dec. 30, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1367578812000430.

[24] P. Yan, Z. Yan, H. Zheng, and J. Guo, "Real Time Robot Path Planning Method Based on Improved Artificial Potential Field Method," in *2018 37th Chinese Control Conference (CCC)*, Jul. 2018, pp. 4814–4820. DOI: 10.23919/ChiCC.2018.8482571. Accessed: Dec. 30, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8482571.

[25] M. Menner and E. Lavretsky, *Translation of Nagumo's Foundational Work on Barrier Functions: On the Location of Integral Curves of Ordinary Differential Equations*, arXiv:2406.18614 [eess], Jun. 2024. DOI: 10.48550/arXiv.2406.18614. Accessed: Apr. 27, 2026. [Online]. Available: http://arxiv.org/abs/2406.18614.

[26] *Parametric Control Barrier Functions based Adaptive Safe Control for Heterogeneous Auton. Vehicles*, May 2022. Accessed: Jun. 25, 2026. [Online]. Available: https://www.youtube.com/watch?v=XWQ_h9wOje0.

[27] M. A. Bhatti, *Practical Optimization Methods*, en. New York, NY: Springer, 2000, ISBN: 978-1-4612-6791-1 978-1-4612-0501-2. DOI: 10.1007/978-1-4612-0501-2. Accessed: Jun. 13, 2026. [Online]. Available: http://link.springer.com/10.1007/978-1-4612-0501-2.

[28] W. Xiao and C. Belta, "High-Order Control Barrier Functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3655–3662, Jul. 2022, ISSN: 1558-2523. DOI: 10.1109/TAC.2021.3105491. Accessed: Jun. 13, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9516971.

[29] P. Lopes and L. T. Paiva, "Lookahead-augmented control barrier functions for safe trajectory tracking in autonomous vehicles," Submitted to CONTROLO'26 – 17th APCA International Conference on Automatic Control and Soft Computing, 2026.

[30] V. Kecman, "Support Vector Machines – An Introduction," en, in *Support Vector Machines: Theory and Applications*, L. Wang, Ed., Berlin, Heidelberg: Springer, 2005, pp. 1–47, ISBN: 978-3-540-32384-6. DOI: 10.1007/10984697_1. Accessed: Jun. 13, 2026. [Online]. Available: https://doi.org/10.1007/10984697_1.

[31] J. A. K. Suykens, "Support Vector Machines: A Nonlinear Modelling and Control Perspective," *European Journal of Control*, vol. 7, no. 2, pp. 311–327, Jan. 2001, ISSN: 0947-3580. DOI: 10.3166/ejc.7.311-327. Accessed: Jun. 13, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0947358001711521.

[32] C. Cortes and V. Vapnik, "Support-vector networks," en, *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 1573-0565. DOI: 10.1007/BF00994018. Accessed: Jun. 19, 2026. [Online]. Available: https://doi.org/10.1007/BF00994018.

[33] *Figure 2: Structure of support vector machines. E. Multilayer...* en. Accessed: Jun. 25, 2026. [Online]. Available: https://www.researchgate.net/figure/Structure-of-support-vector-machines-E-Multilayer-Perceptron-MLP-One-definition-of-a_fig2_367023699.

[34] M. Figueiredo, "On Gaussian radial basis function approximations: Interpretation, extensions, and learning strategies," en, in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 2, Barcelona, Spain: IEEE Comput. Soc, 2000, pp. 618–621, ISBN: 978-0-7695-0750-7. DOI: 10.1109/ICPR.2000.906151. Accessed: Jun. 19, 2026. [Online]. Available: http://ieeexplore.ieee.org/document/906151/.

[35] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY: Springer, 2009, ISBN: 978-0-387-84857-0 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7. Accessed: Jun. 19, 2026. [Online]. Available: http://link.springer.com/10.1007/978-0-387-84858-7.

[36] *Data Interpolation with Radial Basis Functions (RBFs)*. Accessed: Jun. 25, 2026. [Online]. Available: https://shihchinw.github.io/2018/10/data-interpolation-with-radial-basis-functions-rbfs.html.