ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 37

Settore Concorsuale: 09/G1 - AUTOMATICA

Settore Scientifico Disciplinare: ING-INF/04 - AUTOMATICA

SAFE NAVIGATION STRATEGIES FOR QUADROTORS

Presentata da: Biagio Trimarchi

Coordinatore Dottorato
Prof. Michele Monaci

Supervisore
Prof. Lorenzo Marconi

Co-Supervisore
Dott. Fabrizio Schiano

Esame finale anno 2025

Hofstadter's Law:
It always takes longer than you expect,
even when you take into account Hofstadter's Law

DOUGLAS HOFSTADTER

You look at where you're going and where you are
and it never makes sense,
but then you look back at where you've been
and a pattern seems to emerge.

ROBERT M. PIRSIG

Abstract

Autonomous quadcopters are rapidly emerging as a mature technology poised to play a significant role in shaping society in the near future, thanks to their increasing availability and diverse range of applications. From agricultural operations to the transportation of goods and people, these vehicles are set to become an integral part of our daily lives. Given this growing presence, it is crucial to equip these autonomous systems with state-of-the-art algorithms for collision avoidance, which will help prevent damage to people and property, while also ensuring the continued autonomy and operational integrity of the vehicles. The central objective of this thesis is to address this critical challenge.

The initial chapters of the thesis provide a comprehensive introduction to the subject matter, including an overview of the relevant literature. We begin by exploring the dynamical model of quadrotors, highlighting its key properties and the challenges these present when designing feasible trajectories for navigating cluttered environments.

Following this, we delve into the fundamental concepts of Control Barrier Functions (CBFs) and their application to collision avoidance scenarios. We examine how safety filters can be derived from distance measurements and used to design robust control laws for safe navigation in unknown environments.

Finally, driven by the limitations of the sensors commonly employed on autonomous quadrotors—such as monocular and stereo cameras—the concluding section of this work shifts focus towards addressing these constraints. Specifically, we propose an approach based on the CBF framework that accounts for the limited field of view inherent in visual sensors. Additionally, we present a control law, rooted in Control Lyapunov Functions approach, designed to track reference trajectories using feedback based on visual bearings.

Keywords: Autonomous Aerial Vehicles, Control Theory, Collision Avoidance, Gaussian Processes, Lyapunov Methods, Nonlinear Systems, Path Planning, Safety-critical Control, State-space Methods, Visual Servoing

Aknowledgments

This thesis represents the culmination of the research I conducted as a Ph.D. student at the Research Center on Complex and Autonomous Systems (CASY) at the University of Bologna. These have been three meaningful years, and I could not have completed this work without the support of many remarkable individuals.

First, I extend my deepest gratitude to my supervisors, Professor Lorenzo Marconi and Doctor Fabrizio Schiano, for their invaluable guidance and continuous encouragement. Thank you for the insightful advice and for pushing me to give my best effort in all aspects of this research.

I would also like to thank my colleagues in the laboratory: Alessandro Cecconi, Alice Rosetti, Andrea Camisa, Andrea Testa, Andrea Tramaloni, Guido Carnevale, Ivano Notarnicola, Lorenzo Gentilini, Lorenzo Pichierri, Lorenzo Sforni, Marco Borghesi, Mario Spirito, Matteo Lanzarini, Matteo Sartoni, Riccardo Brumali, and Simone Baroncini. I will always remember the time we spent together, from lunches to chess matches during breaks. Special thanks to Cheng for those late nights working side-by-side, for our badminton matches after work, and for the memorable meals we eat together.

I am also grateful to my teammates from the Leonardo Drone Contest: Luigi Chiocchi, Daniele Ceccarelli, Giovanna Crisci, Davide Valenti, Matteo Petrone, Dante Piotto, Simone Molinari, Sebastiano Mengozzi, Sebastiano Bertamé, Francesco Villari, Alberto Elio, and Doctor Nicola Mimmo. Working with you taught me so much and made our achievements possible; the journey would have been far more challenging without your support.

Thanks also to Professor Roberto Tron, who provided me the opportunity to spend a semester at the Boston University Robotics Lab. His supervision and insights during my time there were invaluable.

Special thanks to Laura Lavezzo and Giacomo Santato. The former for kindly reviewing some chapters of this thesis and for offering helpful suggestions on how to improve clarity and flow. The latter for the continuous encouragement during these three years. I hope the best for both of you, and wish both you a successfull Ph.D. career!

Finally, my heartfelt gratitude goes to my partner, Anna. She has stood by me through every high and low, and her constant support has been a source of strength. Thank you for listening, for lifting me up when my spirits were low, and for being my anchor. I love you.

Contents

Abstract **i**

Aknowledgments **iii**

List of Figures **vii**

Notation **1**

1 Introduction **3**
1.1 Literature Review . . . 3
1.2 Structure of the Thesis . . . 5

2 Dynamics and Control **7**
2.1 Dynamical Model . . . 7
2.1.1 Propeller Thrusts and Angular Velocities . . . 8
2.1.2 Euler Angles . . . 9
2.2 Differential Flatness . . . 10
2.3 Control . . . 11

3 Path Planning with Bezier Curves **13**
3.1 Path Planning . . . 13
3.2 Optimal Path and Curve Representation . . . 14
3.2.1 Path Planning with Safe Corridors . . . 16

4 Control Barrier Functions for Obstacle Avoidance **19**
4.1 Definitions and Properties . . . 19
4.1.1 Quadratic Program Safety Filter . . . 21
4.1.2 CLFs . . . 21
4.1.3 Multiple Constraints . . . 22
4.1.4 Application Example: Obstacle Avoidance . . . 23
4.2 High Order Control Barrier Functions . . . 24
4.2.1 Application Example: Obstacle Avoidance . . . 27

5 Safe Navigation in Unknown Environments **29**
5.1 Navigation with Gaussian Processes . . . 29
5.1.1 Full Environment Knowledge . . . 30
5.1.2 Navigation in Unknown Environments . . . 30

5.2 Navigation with log-GPIS . . . 33
5.2.1 Numerical Simulations . . . 36
5.3 High Gain Observers for Safety Filters Learning . . . 36

6 Navigation With Vision Constraints **43**
6.1 Control Barrier Functions for Visual Constraints . . . 43
6.1.1 Velocity Control . . . 45
6.1.2 Acceleration Control . . . 47
6.1.3 Multiple Features . . . 51
6.2 Numerical Simulation . . . 51
6.3 Control Lyapunov Functions for Visual Servoing . . . 53
6.3.1 Numerical Simulation . . . 54

7 Conclusions **57**

A Bézier Curves **59**
A.1 Relation to Monomial Basis . . . 59
A.2 Remarkable Properties . . . 60

B Gaussian Processes **63**
B.1 Gaussian Processes . . . 63
B.2 Gaussian Process Regression . . . 64
B.2.1 Regression . . . 64
B.2.2 Gaussian Process Regression . . . 65

Bibliography **70**

List of Figures

2.1 Sketch of the rigid body model used to describe the quadrotor dynamics. 7

3.1 The figures shows the trajectory generated by 3.4 in a static environments divided in polygonal safe corridors. Figure (a) shows the entire resulting trajectory as a solid red line, while figure (b) shows each of the three Beziérs segments composing the curve, each of them entirely contained in a empty convex polytope. 17

4.1 The figure shows an agent navigating around an obstacle using a safety filter based on CBF. We can see how the vector field of the closed loop system wrap around the obstacle. 24

4.2 The figure shows an agent controlled with a combination of Control Barrier and Control Lyapunov function navigating around an obstacle. We can see how the vector field of the closed loop system wrap around the obstacle. 25

4.3 The figure shows an agent modeled as a double integrator navigating using a HOCBF based safety filter to reach a goal position in the presence of obstacles in the workspaces. 27

5.1 The figures show the *Signed Euclidean Distance Field* of the workspace. Figure (a) shows the tridimensional plot of the function $d_{\mathcal{O}}(p)$, while figure (b) shows the safe set associated with the function $h(x) = d_{\mathcal{O}}(p) - 0.1$. 31

5.2 The figures shows the posterion mean function $\mu(p)$ obtained on the dataset $\mathcal{D}_{50}$. Figure (a) shows the tridimensional plot of $\mu(p)$, while figure (b) shows the safe set associated with the function $h(x) = \mu(p)$. We can see a superposition between the obstacle set $\mathcal{O}$ and the safe set $\mathcal{C}_h$. 31

5.3 The figures shows the posterion mean function $\mu(p)$ obtained on the dataset $\mathcal{D}_{25}$. Figure (a) shows the tridimensional plot of $\mu(p)$, while figure (b) shows the safe set associated with the function $h(x) = \mu(p)$.We can see a superposition between the obstacle set $\mathcal{O}$ and the safe set $\mathcal{C}_h$. 32

5.4 The figure shows the safe set $\mathcal{C}_h$ associated with the function $h(x) = \mu(p) - 0.3$, where μ is the posterion mean obtained on the dataset $\mathcal{D}_{50}$ 32

5.5 The figures show an agent navigating to a goal in an unknown environment while collecting samples of $d_{\mathcal{O}}(p)$ to learn the Control Barrier Function. In figure (a) the agent is modeled as a single integrator, while in figure (b) the agent is modeled as double integrator. In both cases the agent get stuck after a few seconds trying to reach the goal on the other side of the workspace. 33

5.6 The figures show an agent navigating trough a set of carefully selected waypoints p_i^* while collecting samples of $d_{\mathcal{O}}(p)$. In figure (a) the agent is modeled as a single integrator, while in figure (b) the agent is modeled as a double integrator. We can see clearly see that, due to the low number of collected samples, we have $\mathcal{O} \cap \mathcal{C}_h$, and this may lead to a collision. 34
5.7 The figures show a failure case of the proposed approach. By overestimating the dimension of the safe set, the agent crashes on the obstacle and tries to navigate on it before collecting a new sample and realizing the distance got negative. 34
5.8 The figure shows the value of the barrier function $h(x) = \mu(p)$ over time and gives more insight to explain why the agent crushed with the obstacle. We can see at at time $t = 7.3s$, after collecting a new distance sample, the estimate got negative, implying the agent had already crushed. 35
5.9 The figures shows the reconstruction of the distance function trough the use of the log-Gaussian Process Implicit Surface regression, with $h = 3.5$ over a set of obstacle samples that are 0.5 m apart from each other. Figure (a) shows the tridimensional plot of (5.8), while figure (b) shows the corresponding safe set. 37
5.10 The figures shows an agent navigating trough an unkwnown envinroment. In figure (a) the agent is modeled as a single integrator, while Figure(b) it is modeled as a double integrator. 37

6.1 A quadrotor traversing a race circuit needs to keep the gates (depicted in red) as long as possible in its field of view (the blue cone) to orient itself and race through the circuit. 44
6.2 The figure shows the relation between the allowable error ration $\tilde{d}$ and the resulting range for c_2 in Theorem 6.1.3. As we can see, a trade-off exists between the bounds on $\tilde{d}$ and the resulting range for c_2. 50
6.3 Snapshot of the simulation scenario of the numerical experiments. We can see that agent (represented by a blue cross) needs to turn to keep all the features (red points) in its field of view (blue cone) to follow the reference trajectory (black line). (a) Front view. (b) Side view (c) Top-down view. (d) Tilted view. 51
6.4 *Acceleration Control:* Plot (a) shows the position tracking error of a rigid body actuated both in linear and angular acceleration along the prescribed path. Plot (b) shows the minimum value among the barrier functions associated with the features. 52
6.5 *Quadrotor:* Plot (a) shows the position tracking error of a quadrotor actuated both in trust and torques along the prescribed path. Plot (b) shows the minimum value among the barrier functions associated with the features. 52
6.6 Figure (a) shows the scenario of the simulation. $p(T)$ denotes the position of the agent at $T = 10$ (s), and the yellow cone represents its field of view in that instant. Figure (b) shows the position tracking error: p_x denotes the x component of the position vector, and p_y denotes the y component. 55
6.7 Figure (a) show the evolution in time of the various Lyapunov functions V_i. We can see that each of them converge exponentially fast to 0. Figure (b) shows the minimum value of the Control Barrier Functions; as predicted by the theory, it stays positive during the execution of the task. 55

Notation

In the following $\mathbb{N}$ and $\mathbb{R}$ will denote, respectively, the set of naturals and reals numbers. $\mathbb{N}_0$ will denote the set of naturals numbers without the elememnt 0, i.e. $\mathbb{N} \setminus 0$. $\mathbb{R}_+$ and $\mathbb{R}_{++}$ will denote, respectively, set of non negative real numbers and the set of positive real numbers.

Given a matrix $A \in \mathbb{R}^{m \times n}$, with $m, n \in \mathbb{N}_0$, we use $A_{(i,j)}$, $i \in \{1, \dots, m\} j \in \{1, \dots, n\}$, to denote the element on the i^{th} row and j^{th} column of A. The identity matrix of dimension $d \in \mathbb{N}_0$ will be denoted as I_d. The set of symmetric positive definite matrices of dimension $d \in \mathbb{N}_0$ will be denoted as S_+^d.

The set of rotation matrices of dimension $d \in \mathbb{N}_0$ will be denoted with $SO(3)$, while the set of skew-symmetric matrices of dimension $d$of dimension d will be denoted as $\mathfrak{so}(3)$.

Given a unit vector $v \in \mathbb{R}^d$, with $d \in \mathbb{N}_0$, we will denote with $P_v \in \mathbb{R}^{d \times d}$ the projection matrix of the vector v, which is defined as $P_v = I_d - vv^T$.

We denote with $[\cdot]_\times : \mathbb{R}^3 \to \mathfrak{so}(3)$ the *hat map* which is defined as

$$[v]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

where $v_1, v_2, v_2 \in \mathbb{R}$ are the component of vector $v \in \mathbb{R}^3$. The inverse of the hat map, called the *vee map*, will be denoted as $\cdot^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$.

Given a smooth manifold $\mathcal{M}$, we denote the *tangent space* of $\mathcal{M}$ at $x \in \mathcal{M}$ as $T_x\mathcal{M}$. In this work we always equip each tangent space $T_p\mathcal{M}$ with standard *inner product* $\langle \cdot, \cdot \rangle_x : T_p\mathcal{M} \times \mathcal{M} \to \mathbb{R}$ defined as

$$\langle v, w \rangle_x = v^T w.$$

As an abuse of notation, we will drop the subscript x when it is clear from the context.

Given a function $f : \mathcal{M} \to \mathbb{R}$, we will denote with $\nabla f : \mathcal{M} \to T_x\mathcal{M}$ the *gradient* of f, and with $Hess f[\cdot] : T_x\mathcal{M} \to T_x\mathcal{M}$ its *Hessian*.[1]

Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$, with $m, n, p, q \in \mathbb{N}_0$, the *Kronecker product* of A and B, denoted as $A \otimes B$, is defined as

$$A \otimes B = \begin{bmatrix} A_{(1,1)}B & A_{(1,2)}B & \dots & A_{(1,n)}B \\ A_{(2,1)}B & A_{(2,2)}B & \dots & A_{(2,n)}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{(m,1)}B & A_{(m,2)}B & \dots & A_{(m,n)}B \end{bmatrix}$$

Given a finite of points $S \in \mathbb{R}^d$, $d \in \mathbb{N}_0$, we denote the *convex hull* of S with $conv\,(S)$.

[1] See [1] for a formal definition.

Chapter 1

Introduction

Autonomous quadrotors represent a groundbreaking technology poised to redefine various aspects of modern life. From precision agriculture to rapid goods delivery and even passenger transport, their capabilities are transforming industries and inspiring innovative solutions across domains. However, as their presence in our lives increases, so too does the need for reliable, robust algorithms to ensure safe, collision-free navigation.

This thesis provides an in-depth exploration of the algorithms and techniques essential for achieving safe, autonomous flight in complex environments. Building on the latest advancements in control theory, including path planning through Bézier curve frameworks, Gaussian Process-based safety filters, and vision-based control through Control Barrier and Control Lyapunov Functions, this work aims to address the critical challenge of reliable collision avoidance. Together, these efforts represent a contribution toward a future where quadrotors can operate with an unprecedented degree of autonomy and safety, paving the way for a new era of unmanned aerial applications.

1.1 Literature Review

The field of collision avoidance is as old as the field of mobile robotics itself, emerging as an essential component in any autonomous system moving through a dynamic environment. From foundational artificial potential fields to sophisticated control barrier functions, the development of collision avoidance techniques has played a central role in ensuring safe robotic navigation.

One of the earliest demonstrations of autonomous movement was seen in Walter's 1950 tortoise robot, an early exploration in robotics that illustrated the potential for machines to navigate independently [2]. Decades later, Khatib formally introduced the concept of artificial potential fields (APFs) [3], proposing virtual repulsive forces to guide robots away from obstacles. APFs laid the foundation for many future developments, though they were limited by local minima, which can trap robots in undesired configurations. Navigation functions emerged to address these issues by introducing global stability properties for obstacle-rich environments, providing robust navigation in spaces with complex obstacle shapes [4, 5, 6, 7]. Further refinements by Loizou [8] mitigated issues with local minima through diffeomorphisms, increasing the applicability of APFs to realistic and more challenging environments. For quadrotors specifically, vector field control techniques [9] have become valuable in supporting directionally controlled motion in complex obstacle settings, enhancing safe navigation through dynamic fields.

Path planning has also been instrumental to collision avoidance, with foundational contributions from sampling-based methods like Rapidly-exploring Random Trees (RRT) [10] and Probabilistic

Road Maps (PRM) [11], which remain core techniques in robotics. These methods have proved particularly suitable for high-dimensional spaces, an advantage essential for navigating the complex 3D environments encountered by quadrotors. Further improvements by Karaman and Frazzoli [12] refined RRT and PRM algorithms to enhance efficiency, thereby cementing them as standard approaches in path planning for mobile and aerial robots.

The specific dynamics of quadrotors necessitate a range of control strategies that are tailored to the intricacies of six degrees of freedom (6-DOF) flight. Lee et al. [13] laid foundational work in geometric control for quadrotors, effectively utilizing geometric principles to achieve precise orientation and trajectory management. Complementary to this approach, the concept of differential flatness, introduced by Mellinger et al. [14, 15], has simplified trajectory generation by transforming complex dynamics into simpler representations. These two approaches have together shaped a solid framework for the control of quadrotors in both controlled and unpredictable environments.

Building upon these earlier control frameworks, Control Barrier Functions (CBFs) have become a powerful approach for maintaining safe operation within defined constraints. Ames and collaborators pioneered CBFs in adaptive cruise control applications [16, 17] and subsequently broadened their application to various robotics domains [18]. By constraining control inputs to guarantee forward invariance of safe states, CBFs provide an effective method for collision avoidance in safety-critical scenarios. Exponential CBFs [19] and high-order CBFs [20] have been introduced to enable applications in systems with intricate dynamics. Singletary's comparative analysis [21] of CBFs and APFs highlighted the enhanced reliability of CBFs for collision avoidance, especially in dynamic environments. Recent contributions by Krstic [22] proposed inverse optimal filters to extend the versatility of CBFs, allowing greater flexibility in the design of safety filters across variable conditions. For quadrotors in particular, geometric CBFs [23, 24] have been tailored to handle their unique dynamics, further enhancing the efficacy of CBFs for these applications.

A complementary area of research in collision avoidance is polynomial path planning, especially through the use of Bézier curves and B-splines. These methods are particularly advantageous for generating smooth, feasible trajectories that respect the dynamics of quadrotors. Mueller et al. [25] demonstrated how quadratic programming (QP) could be applied to compute real-time trajectories, a crucial advantage for complex flight maneuvers. Lakshmanan [26] extended this QP-based approach by employing Bézier polynomials, which enhance smoothness in trajectory continuity and transition. Recent advancements have focused on Bézier curve-based planners specifically designed for cluttered environments [27, 28, 29], providing refined control in intricate obstacle fields. Buffer-based Voronoi cell methods [30, 31] and receding horizon techniques [32] further contribute to collision avoidance by ensuring consistent safe distances from obstacles.

In complex 3D environments, B-spline-based trajectory optimization techniques such as those presented by Zhou et al. [33] have shown considerable promise for path generation, as they allow paths to conform to both quadrotor dynamics and the distribution of obstacles. More recently, deep learning approaches [34] have been introduced to manage the intricate task of time allocation and to adapt quadrotor trajectories to real-time conditions, enabling them to react flexibly to new environmental conditions.

Together, these advancements in artificial potential fields, quadrotor control, path planning, and control barrier functions underscore the evolution of techniques necessary for robust collision avoidance in autonomous aerial vehicles. This thesis builds on these methodologies to contribute a unified framework for safe and efficient quadrotor navigation, synthesizing concepts from polynomial path planning, control barrier functions, and data-driven safety filters. The resulting navigation strategies bring quadrotors closer to reliable operation in complex, real-world environments, pushing forward the capabilities of autonomous aerial systems.

1.2 Structure of the Thesis

This thesis is structured as follows. We start by presenting a dynamical model of a quadorotor in Chapter 2 along with its properties, such as the differential flatness, and the more popular control schemes used to hover and tracking trajectories. In Chapter 3 we describe in general the path planning problem and then explain the nuances involved in designing trajectory that can be flyed by a quadrotor. In this chapter we also present a framework to generate safe trajectory in cluttered environments. In Chapter 4 we present recall the theory of Control Barrier Function, and we present some numerical example to explain their use in the context of obstacle and collision avoidance. In Chapter 5 we investigate how to learn safety filters from a set of collected data using Gaussian Processes. We start by presenting a direct approach used in the literature and by highlighting its weak points. Then we present a technique specialized on distance reconstruction that we will use to construct a Control Barrier Function for safe navigation in unknown environments. And we end the chapter with a technique aimed at learning the safety filter constraint of High Order Control Barrier Functions. Chapter 6 focus is quite different from the rest of thesis. While all the other chapters deal directly with the main topic of the thesis, i.e. collision avoidance, this chapter focused is dealing with limited field of view of vision sensor and on the design of trajectory tracking algorithms that are based on bearing measurements. In the end, in Chapter 7 we summarize the results of our work and discuss about future research directions.

Chapter 2

Dynamics and Control

This chapter introduces the standard mathematical model used to model the dynamics of a quadrotor to design control and motion planning algorithms. We also recall the differential flatness properties of the proposed model [14, 15], which permits to easily extract a reference state trajectory from the desired motion of the center of gravity of the vehicle, and one of the most popular control law for this kind of vehicles [13] to track the desired trajectory despite the presence of modelling errors. The interested reader may find a more comprehensive discussion about the topics introduced in this chapter may in [35], which also include references to other didactic and scientific material.

2.1 Dynamical Model

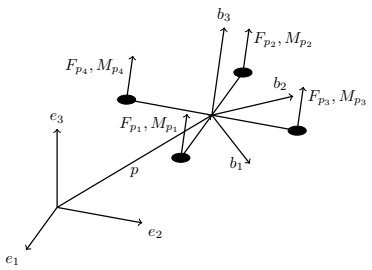


Figure 2.1: Sketch of the rigid body model used to describe the quadrotor dynamics.

As a first approximation, a quadrotor can be modeled as an actuated rigid body, where each propeller applies a force on the mounting point. So, let us consider an inertial frame $\mathcal{O}$ with the orthonormal base e_1, e_2, e_3 and a moving reference frame $\mathcal{B}$ with the orthonormal base b_1, b_2, b_3 centered in the center of mass of the quadrotor and attached to it, such as in figure 2.1. In the following, we will denote with the superscript $\mathcal{O}$ the quantities expressed in the inertial reference frame, and the superscript $\mathcal{B}$ the quantities expressed in the body frame.

We denote with $p^{\mathcal{O}} \in \mathbb{R}^3$ the position of the quadrotor in the inertial frame, and with $R \in SO(3)$ the orientation of the quadrotor reference frame with respect to the inertial one, which is given by components of the orthonormal bae of $\mathcal{B}$ expressed in $\mathcal{O}$, i.e. $R = \begin{bmatrix} b_1^{\mathcal{O}} & b_2^{\mathcal{O}} & b_3^{\mathcal{O}} \end{bmatrix}$. We denote with $v^{\mathcal{O}} \in T_p\mathbb{R}^3$ the linear velocity of the quadrotor in the inertial frame and with $\omega^{\mathcal{B}} \in T_R SO(3)$ the angular velocity of the quadrotor in the body frame. Likewise, we denote with $a^{\mathcal{O}} \in T_p\mathbb{R}^3$ the linear acceleration of the quadrotor in the inertial frame and with $\alpha^{\mathcal{B}} \in T_p\mathbb{R}^3$ the angular acceleration of the quadrotor in the body frame. Moreover, we denote the quadrotor mass with $m \in \mathbb{R}_{++}$, and with $J \in S^3_+$ its inertia matrix in the body reference frame. Let also $g^{\mathcal{O}} \in T_p\mathbb{R}^3$ denotes the acceleration of gravity. By denoting with $T \in \mathbb{R}_{++}$ the total force applied to the body, and by

$\tau^{\mathcal{B}} \in \mathbb{R}^3$ the total torque, we can write the equations of motion as follows

$$\dot{p}^{\mathcal{O}} = v^{\mathcal{O}} \tag{2.1a}$$
$$m\dot{v}^{\mathcal{O}} = mg^{\mathcal{O}} + Rb_3^{\mathcal{B}}T \tag{2.1b}$$
$$\dot{R} = R[\omega^{\mathcal{B}}]_\times \tag{2.1c}$$
$$J\dot{\omega^{\mathcal{B}}} = -[\omega^{\mathcal{B}}]_\times J\omega^{\mathcal{B}} + \tau^{\mathcal{B}} \tag{2.1d}$$

We collect the position, rotation, velocity, and angular velocity of the quadrotor in a single variable $x = \left(p^{\mathcal{O}}, R, v^{\mathcal{O}}, \omega^{\mathcal{B}}\right) \in \mathcal{X} \subset \mathbb{R}^3 \times SO(3) \times T_p\mathbb{R}^3 \times T_R SO(3)$ that fully describe the *state* of the dynamical model at each time instant. We collect the two acceleration in the variable $u = (T, \tau^{\mathcal{B}}) \in \mathcal{U} \subset \mathbb{R}_{++} \times \mathbb{R}^3$, which represent the *control input* of the dynamical system, i.e. the parameter that we assume we can set to any arbitrary value to steer the evolution of state. The set $\mathcal{U}$ encodes in its definition the actuation limits of the propellers, while the set $\mathcal{X}$ encodes workspace limitation, e.g. walls and unreachable areas, and physical limitation on the maximum achievable velocities.

2.1.1 Propeller Thrusts and Angular Velocities

Model (2.1) uses the total force and the total torque as control input. However, a quadrotor is actuated by the trusts produced by its rotors, which in turn is related to the angular velocities of the propellers. In this section we briefly recall how to model the relation between the aformentioned quantities.

First, we notice that the total thrust T and torques $\tau^{\mathcal{B}}$ applied on the quadrotor can be directly related to the thrust of each single propeller by simple algebraic equations. By denoting with $F_{p_i} \in \mathbb{R}$ the thrust of the i^{th} propeller, the total trust is given by

$$T = \sum_{i=0}^{4} F_{p_i}$$

Moreover, if we denote the position of the i^{th} propeller of the quadrotor with respect to vehicle center of mass with $r_{p_i}^{\mathcal{B}} \in \mathbb{R}^3$, and with $M_{p_i} \in \mathbb{R}$ the magnitude of the torque produces by each propeller around the b_3 axis, we can compute the total torque applied on the quadrotor as

$$\tau^{\mathcal{B}} = \sum_{i=1}^{4} [r_{p_i}^{\mathcal{B}}]_\times b_3^{\mathcal{B}} F_{p_i} + M_{p_i} b_3^{\mathcal{B}}$$

So, we can collect all of the previous equations in a compact matrix multiplication

$$\begin{bmatrix} T \\ \tau^{\mathcal{B}} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ [r_{p_1}^{\mathcal{B}}]_\times b_3^{\mathcal{B}} & [r_{p_2}^{\mathcal{B}}]_\times b_3^{\mathcal{B}} & [r_{p_3}^{\mathcal{B}}]_\times b_3^{\mathcal{B}} & [r_{p_4}^{\mathcal{B}}]_\times b_3^{\mathcal{B}} & b_3^{\mathcal{B}} & b_3^{\mathcal{B}} & b_3^{\mathcal{B}} & b_3^{\mathcal{B}} \end{bmatrix} \begin{bmatrix} F_{p_1} \\ F_{p_2} \\ F_{p_3} \\ F_{p_4} \\ M_{p_1} \\ M_{p_2} \\ M_{p_3} \\ M_{p_4} \end{bmatrix} \tag{2.2}$$

Now that we are equipped with equation (2.2), we only need to find a relation between the propellers rotation speed ω_{p_i} and the propellers thrusts F_{p_i} and torques M_{p_i}. According to [15], the relation can be approximated by the following algebraic equation:

$$\begin{aligned} F_{p_i} &= K_F \omega_{p_i}^2 \\ M_{p_i} &= (-1)^{(i-1)} K_M \omega_{p_i}^2 \end{aligned}$$

where $K_F, K_M \in \mathbb{R}_{++}$ are parameters that depends on the chosen propellers, while the -1 powers reflect the fact that propellers 1 and 3 rotate counter-clockwise while propeller 2 and 4 rotate clockwise.

2.1.2 Euler Angles

Before concluding this section, we take the opportunity to introduce some important quantities that we will use in the next section.

As we already said, we represent the orientation of drone with respect to the inertial frame by the rotation matrix R. Nonetheless, this is not the only possible choice to represent orientations and rotations, and despite all of its properties, it is not the most intuitive representation. So, while the orientation of a rigid body is more accurately described by a rotation matrix, the description used by human designers to encode the desired rotations is that of the so called *Euler Angles*, which are a triple of consecutve rotations around non parallel axis which are able to describe *almost* all rotations.

In our discussion we will use the ZXY Euler Angles triple, which means that the relation between the orientation of the drone R and the ϕ, θ, ψ is given by:

$$\begin{aligned} R =& R_y(\theta) R_x(\phi) R_z(\psi) = \\ =& \begin{bmatrix} \cos\psi\cos\theta - \sin\phi\sin\psi\sin\theta & -\cos\phi\sin\psi & \cos\psi\sin\theta + \cos\theta\sin\psi\sin\phi \\ \sin\psi\cos\theta - \sin\phi\cos\psi\sin\theta & \cos\phi\cos\psi & \sin\psi\sin\theta - \cos\theta\cos\psi\sin\phi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{bmatrix} \end{aligned}$$

where $R_x(\phi), R_y(\theta), R_z(\psi)$ are given, respectively, by

$$\begin{aligned} R_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\ R_y(\theta) &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \\ R_z(\psi) &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

With this representation choice, one can prove that the components of the angular velocity of the quadrotor $\omega^{\mathcal{B}} = [\omega_1, \omega_2, \omega_3]^T$ are related to the rate of change of ϕ, θ, ψ by

$$\begin{bmatrix} \cos\theta & 0 & -\cos\phi\sin\theta \\ 0 & 1 & \sin\phi \\ \sin\theta & 0 & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{2.3}$$

2.2 Differential Flatness

The control scheme we will review in section 2.3 needs a feasible state trajectory as reference, that is a curve $x^*(t) : \mathbb{R} \to \mathcal{X}$ that corresponds to desired behaviour we want to impose on the system. Obtaining such state space trajectory can be arbirarily challeging for a dynamical system, but it has been shown in [14, 15] that quadrotor dynamic is *differentially flat* with respect to the position of its center of mass $p^{\mathcal{O}}$ and its yaw angle ψ, and this property permits us to obtain a state space trajectory from the time evolution of this two quantities.

First of all, we can notice that $p^{\mathcal{O}}$ is already part of the state, and that $v = \dot{p}^{\mathcal{O}}$ for equation (2.1a). Rearraging the terms in equation (2.1b), we obtain

$$Rb_3^{\mathcal{B}}T = m\ddot{p}^{\mathcal{O}} + mg^{\mathcal{O}}$$

from which we derive

$$T = \|m\ddot{p}^{\mathcal{O}} + mg^{\mathcal{O}}\|_2$$
$$b_3^{\mathcal{O}} = \frac{m\ddot{p}^{\mathcal{O}} + mg^{\mathcal{O}}}{\|m\ddot{p}^{\mathcal{O}} + mg^{\mathcal{O}}\|_2}$$

which gives us one the control input and the third column of the rotation matrix R.

To fully determine the rotation matrix R we need to determine also $b_1^{\mathcal{O}}$ and $b_2^{\mathcal{O}}$, and this can be easily done from the yaw angle ψ now that we know $b_3^{\mathcal{O}}$. First, we define the auxiliary unit vector

$$b_{1c}^{\mathcal{O}} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ 0 \end{bmatrix}$$

which gives the orientation of the b_1 axis of the body frame when only the first of the three Euler Angles rotation is applied (i.e. the one around the b_3 axis). Due to the structure of the chosen Euler Angles triple, the orientation of the b_2 axis do not change with the subsequent rotations, so we can determine b_2 and b_1 as follows:

$$b_2^{\mathcal{O}} = \frac{b_3^{\mathcal{O}} \times b_{1c}^{\mathcal{O}}}{\|b_3^{\mathcal{O}} \times b_{1c}^{\mathcal{O}}\|_2}, \quad b_1^{\mathcal{O}} = b_2^{\mathcal{O}} \times b_3^{\mathcal{O}}$$

and now we have all the components to determine the orientation of the quadrotor:

$$R = \begin{bmatrix} b_1^{\mathcal{O}} & b_2^{\mathcal{O}} & b_3^{\mathcal{O}} \end{bmatrix}$$

To find the angular velocity we need to derive equation (2.1b)

$$m\,\dddot{p}^{\mathcal{O}} = \dot{R}b_3^{\mathcal{B}}T + Rb_3^{\mathcal{B}}\dot{T} = R[\omega^{\mathcal{B}}]_\times b_3^{\mathcal{B}}T + Rb_3^{\mathcal{B}}\dot{T}.$$

We notice that $\langle m\,\dddot{p}^{\mathcal{O}}, Rb_3^{\mathcal{B}} \rangle = Rb_3^{\mathcal{B}}\dot{T}$, which permits us to compute $Rb_3^{\mathcal{B}}\dot{T}$ from the knowledge of $\dddot{p}^{\mathcal{O}}$. Let us define the auxiliary quantity

$$h_\omega^{\mathcal{O}} = R[\omega]_\times b_3^{\mathcal{B}} = \frac{m}{T}\left(\dddot{p}^{\mathcal{O}} - \langle m\,\dddot{p}^{\mathcal{O}}, Rb_3^{\mathcal{B}} \rangle Rb_3^{\mathcal{B}}\right)$$

which has component only along b_1 and b_2. Expanding the computation, we obtain:

$$h_\omega^{\mathcal{O}} = \omega_2 b_1^{\mathcal{O}} - \omega_1 b_2^{\mathcal{O}}$$

which implies:

$$\omega_1 = -\langle h_\omega, b_2 \rangle, \ \omega_2 = \langle h_\omega, b_1 \rangle$$

To find the third component, we can notice that, due to the order of Euler angles chosen for represeting the rotation, the body angular velocity is related to the angular rates via:

$$\omega = \begin{bmatrix} b_{1c}^{\mathcal{O}} & b_2^{\mathcal{O}} & e_3^{\mathcal{O}} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.4}$$

from which we can solve for ω_3.

We now have all the state components associated with the reference trajectory. By deriving (2.1b) one more time we can find also the remaining control inputs by manipulating the resulting equation:

$$m \dddot{p}^{\mathcal{O}} = b_3^{\mathcal{B}} \ddot{T} + R[\omega^{\mathcal{B}}]_\times [\omega^{\mathcal{B}}]_\times b_3^{\mathcal{B}} T + R[\alpha^{\mathcal{B}}]_\times b_3^{\mathcal{B}} T + 2R[\omega^{\mathcal{B}}]_\times b_3^{\mathcal{B}} \dot{T}.$$

From the last equation we can easily compute the components of $\alpha^{\mathcal{B}}$ along b_1 and b_2 as we have done for the angular acceleration. Let us define the auxiliary quantity $h_\alpha^{\mathcal{O}} = R[\alpha^{\mathcal{B}}]_\times b_3^{\mathcal{B}}$, and notice that

$$h_\alpha^{\mathcal{O}} = \frac{m \dddot{p}^{\mathcal{O}} - (\ddot{T} b_3 + R[\omega^{\mathcal{B}}]_\times [\omega^{\mathcal{B}}]_\times b_3^{\mathcal{B}} T + 2R[\omega^{\mathcal{B}}]_\times b_3^{\mathcal{B}} \dot{T})}{T}$$

By projecting the above quantity onto the b_1 and b_2 axis, we get

$$\dot{\omega}_1 = -\langle h_\alpha, b_2 \rangle, \dot{\omega}_2 = \langle h_\alpha, b_1 \rangle$$

To find $\dot{\omega}_3$, one need to derive (2.4) and solve for $\dot{\omega}_3$.

Equipped with the knowledge of $\dot{\omega}_1$, $\dot{\omega}_2$ and $\dot{\omega}_3$, which are the components of $\alpha^{\mathcal{B}}$, we can find $\tau^{\mathcal{B}}$ from equation (2.1d).

2.3 Control

Equipped with the reference state trajectory $x^*(t) = (p^{\mathcal{O}*}(t), v^{\mathcal{O}*}(t), R^*(t), \omega^{\mathcal{B}*}(t))$ and the desired control input $u^*(t) = (T^*(t), \tau^{B*}(t))$ obtained from Section () we are now ready to design the control law for our quadrotor. In an ideal case, if the state of our quadrotor $x(t)$ was exactly $x^*(t)$ at the initial time instant $t_0 \in \mathbb{R}$, i.e. $x(t_0) = x^*(t_0)$, we could just apply the control input $u^*(t)$ for $t \geq t_0$ and the controlled quadrotor would follow the desired state. However, this is not the case and one would see the controlled quadrotor deviating significantly from the prescribed trajectory using the reference control law $u^*(t)$. The reason lies in the discrepancy between our simplified abstraction of the quadrotors dynamics, i.e. model , and the real world drone, which is far more complex and subject to a diverse range of phenomena not modeled in (2.1), such as wind effects, vibrations, elasticity, delay in the actuation, and so on. Moreover, the quality of our model depends on our ability to accurately measure the parameters that appears in the laws of motion, such as, in our case, the quadrotor mass m and inertia J, which affects not only the dynamics but also the computation of the reference control law $u^*(t)$. This means that to track as accurately as possible the reference state trajectory, we need to design a control law able to compensate for all of these mismatch between the simplified model and the real world system.

The control algorithm briefly recalled in this section is the one presented [13], which is a feedback state controller, which is a controller that computes the control inputs by elaborating the error betwen the desired state and the actual state. For the following discussion, we define the following quantities $e_p = p^{\mathcal{O}} - p^{\mathcal{O}*}$, $e_R = (R^{*T}R - R^T R)^\vee$, $e_v = v^{\mathcal{O}} - v^{\mathcal{O}*}$, and $e_\omega = \omega^{\mathcal{B}} - \omega^{\mathcal{B}*}$, which represents, respectively, the position error, the orientation error, the velocity error, and the angular velocity error. The control algorithm falls in the category *PD control with feedforward terms*, meaning that we rely on $u^*(t)$ to steer the system as good as possible on along the desired trajectory and we add to it a correction term that is proportional to the current state error:

$$
\begin{aligned}
T &= \left\langle -k_p e_p - k_v e_v - m g^{\mathcal{O}} + m \ddot{p}^{\mathcal{O}*}, b_3^{\mathcal{O}} \right\rangle \\
\tau^{\mathcal{B}} &= -k_R e_R - k_\omega e_\omega + [\omega^{\mathcal{B}}]_\times J \omega^{\mathcal{B}} - J([\omega^{\mathcal{B}}]_\times R^T R^* \omega^{\mathcal{B}*} - R^T R^* \alpha^{\mathcal{B}*})
\end{aligned}
$$

A more detailed explanation of this control law can be found if the original paper [13], along with the Lyapunov analysis to prove its convergence guarantuees.

Chapter 3

Path Planning with Bezier Curves

3.1 Path Planning

The aim of this section is to formalize several version of the path planning problem. First of all, we need to define some variables that will be of aid in this task.

The first thing we need to define is the set $\mathcal{P}$, which will denote the workspace in which an autonomous agent lives and moves (e.g. a robot, a car), which we identify with either $\mathbb{R}^2$ when the movement of the agent is restricted to a plane or $\mathbb{R}^3$ when the agent is able to move in a three dimensional space.

The position in the workspace occupied by the agent is denoted by $p \in \mathcal{P}$. As per our first definition definition of the path planning problem we need two position in the workspace, $p_0, p_F \in \mathcal{P}$, which will be referred to as the *initial position* and *goal position* and represent, as the names suggest, the position from which the agent start and the position it needs to reach. We also need two time instant, $t_0, t_F \in \mathbb{R}_+$ with $t_F > t_0$, which are called respectively the *initial time* and the *final time* and represent the time window to reach the goal position. We are now ready to give out first formulation of the path planning problem.

Problem 3.1.1 (Path Planning)**.** *Given p_0, $p_F \in \mathcal{P}$, t_0, and t_F, find a continuous curve $p : [t_0, t_F] \to \mathcal{P}$ such that:*

- $p(t_0) = p_0$ (Initial position)
- $p(t_F) = p_F$ (Final position)

let $\mathcal{S} \in \mathcal{P}$ denotes the region of space occupied by static obstacle, let $\mathcal{O} \in \mathbb{N}$ be a set of index, each corresponding to different moving obstacle.

Problem 3.1.2 (Path Planning with Static and Moving Obstacles)**.** *Given p_0, $p_F \in \mathcal{P}$, $\mathcal{S}$, $\mathcal{O}$, t_0, and t_F find a continuous curve $p : [t_0, t_F] \to \mathcal{P}$ such that:*

- $p(t_0) = p_0$ (Initial position)
- $p(t_F) = p_F$ (Final position)
- $\forall t \in [t_0, t_F]$, $p(t) \notin \mathcal{S}$ (Avoid collision with static obstacles)
- $\forall t \in [t_0, t_F]$, $\forall i \in \mathcal{O}$, $\|p(t) - o_i(t)\|_2 > r_s$ (Avoid collision with moving obstacle)

where $o_i(t) \in \mathcal{P}$ represents the position of the i^{th} obstacle at time t.

Despite the generality of its statements, trajectories that are solution to Problem 3.1.2 may still lack a fundamental property needed for real world applications: *physical feasibility*. In fact, any agent operating in the real world is subject to law of physics, and they may prevent the agent from executing an arbitrary trajectory. If we recall the discussion about differential flatness in Section 2.2, there we show the existence of a mapping between 4 times continously differentiable trajectories and the state trajectory of the center of mass of a quadrotor. So, we want to encode this constraint in our path planning problem. To achieve our aim, let us define $v, a, j \in \mathbb{R}^3$ ($v, a, j \in \mathbb{R}^2$ for the planar case) to be, respectively, the velocity, the acceleration and the jerk (i.e. the time derivative of the acceleration of the agent); then our new path planning problem becomes the following.

Problem 3.1.3 (Path Planning with Dynamics)**.** *Given p_0, $p_F \in \mathcal{P}$, $\mathcal{S}$, $\mathcal{O}$, t_0, and t_F, find a C^3 curve $p : [t_0, t_F] \to \mathcal{P}$ such that:*

- $p(t_0) = p_0$, $v(t_0) = 0$, $a(t_0) = 0$, $j(t_0) = 0$ (Initial condition)
- $p(t_F) = p_F$, $v(t_F) = 0$, $a(t_F) = 0$, $j(t_F) = 0$ (Final position)
- $\forall t \in [t_0, t_F]$, $p(t) \notin \mathcal{S}$ (Avoid collision with static obstacles)
- $\forall t \in [t_0, t_F]$, $\forall i \in \mathcal{O}$, $\|p(t) - o_i(t)\|_2 > r_s$ (Avoid collision with moving obstacle)

where $o_i(t) \in \mathcal{P}$ represents the position of the i^{th} obstacle at time t.

3.2 Optimal Path and Curve Representation

A solution to Problem 3.1.3 can be found by solving the following optimization problem

$$\begin{aligned} \min_{p(t)\in C^3} \quad & \int_{t_0}^{t_F} J(p(t))dt \\ \textit{subject to} \quad & p(t_0) = p_0, \quad p(t_F) = p_F \\ & v(t_0) = 0, \quad v(t_F) = 0 \\ & a(t_0) = 0, \quad a(t_F) = 0 \\ & j(t_0) = 0, \quad j(t_F) = 0 \\ & p(t) \notin \mathcal{S}, \quad \forall t \in [t_0, t_F] \\ & \|p(t) - o_i(t)\|_2 > r_s, \quad \forall t \in [t_0, t_F], \forall i \in \mathcal{O} \end{aligned} \tag{3.1}$$

where $J : C^3([x_0, x_F]) \to \mathbb{R}$ is a cost functional that can be used to specify other desired characteristics of the solution.

To solve problem 3.1, which is an infinite dimensional optimization problem, we are going to restrict the domain of the optimization problem from the entire C^3 to a smaller subspace that posses a finite base to make it tractable. The chosen subspace is that of the polynomial of degree 7, which will be denoted as $P^7([t_0, t_F])$, which, as we will soon see, will permit us to easily write and enforce the constraints of the optimization problem. To express the elements of $P^7([t_0, t_F])$ we choose to use the Bernstein Polynomials basis (see Appendix A) which permits us to write any polynomial curve $p(t) : [t_0, t_F] \to \mathcal{P}$ as

$$p(t) = \sum_{k=0}^{7} B_k^7 \left(\frac{t - t_0}{t_F - t_0} \right) P_k,$$

where $B_k^7 : [0,1] \to [0,1]$ are the *Bernstein Polynomial* of order 7, and the $P_k \in \mathcal{P}$ are the *control points* of the curve.

To allow a larger degree of flexibility in the design of the curve, we divide the interval $[t_0, t_F]$ into $N \in \mathbb{N}_0$ intervals $[t_i, t_{i+1}] \in [t_0, t_F]$ such that $\bigcup_{i=1}^{N}[t_{i-1}, t_i] = [t_0, t_F]$ (notice that $t_N = t_F$), and on each of them we define the the curve

$$p_i(t) = \sum_{k=0}^{7} B_k^7 \left(\frac{t - t_{i-1}}{t_i - t_{i-1}} \right) P_k^i,$$

where P_k^i are the control points of the i^{th} curve, so that we can express $p(t)$ as

$$p(t) = p_i(t), \quad t \in [t_{i-1}, t_i]. \tag{3.2}$$

For compactness, we stack all of the control points of the curves $p_i(t)$ into the vectors $\mathbf{P}_i = [P_0^{iT}, P_1^{iT}, \dots, P_7^{iT}]^T \in \mathbb{R}^{8dim(\mathcal{P})}$, and, exploiting the properties of Beziér Curves recalled in Appendix A, this permit us to write the following sub-optimal version of problem (3.1)

$$\begin{aligned}
\min_{\mathbf{P_1},\dots,\mathbf{P_N} \in \mathcal{P}^N} \quad & \int_{t_0}^{t_F} J(p(t))dt \\
\textit{subject to} \quad & P_0^1 = p_0, \quad P_7^N = p_F \\
& A_{1,1}\bar{D}_7^1 \mathbf{P}_1 = 0, \quad A_{2,1}\bar{D}_7^1 \mathbf{P}_N = 0 \\
& A_{1,2}\bar{D}_7^2 \mathbf{P}_1 = 0, \quad A_{2,2}\bar{D}_7^2 \mathbf{P}_N = 0 \\
& A_{1,3}\bar{D}_7^3 \mathbf{P}_1 = 0, \quad A_{2,3}\bar{D}_7^3 \mathbf{P}_N = 0 \\
& A_{2,0}\mathbf{P}_i = A_{1,0}\mathbf{P}_{i+1}, \quad \forall i \in \{1, \dots, N-1\} \\
& \frac{1}{t_i - t_{i-1}} A_{2,1}\bar{D}_7^1 \mathbf{P}_i = \frac{1}{t_{i+1} - t_i} A_{1,1}\bar{D}_7^1 \mathbf{P}_{i+1}, \quad \forall i \in \{1, \dots, N-1\} \\
& \frac{1}{(t_i - t_{i-1})^2} A_{2,2}\bar{D}_7^2 \mathbf{P}_i = \frac{1}{(t_{i+1} - t_i)^2} A_{1,2}\bar{D}_7^2 \mathbf{P}_{i+1}, \quad \forall i \in \{1, \dots, N-1\} \\
& \frac{1}{(t_i - t_{i-1})^3} A_{2,3}\bar{D}_7^3 \mathbf{P}_i = \frac{1}{(t_{i+1} - t_i)^3} A_{1,3}\bar{D}_7^3 \mathbf{P}_{i+1}, \quad \forall i \in \{1, \dots, N-1\} \\
& p(t) \notin \mathcal{S}, \quad \forall t \in [t_0, t_F] \\
& \|p(t) - o_i(t)\|_2 > r_s, \quad \forall t \in [t_0, t_F], \forall i \in \mathcal{O}
\end{aligned} \tag{3.3}$$

where $\bar{D}_7^j \in \mathbb{R}^{(8-j)dim(\mathcal{P}) \times 8dim(\mathcal{P})}$ is the *derivation matrix* defined in Theorem A.2.2.1, while $A_{1,j}, A_{2,j} \in \mathbb{R}^{dim(\mathcal{P}) \times (8-j)dim(\mathcal{P})}$, $j \in 0, 1, \dots, 4$, are auxiliary matrices defined as follows

$$\begin{aligned}
A_{1,j} &= \underbrace{[1, 0, \dots, 0]}_{8-j} \otimes I_{dim(\mathcal{P})} \\
A_{2,j} &= \underbrace{[0, 0, \dots, 1]}_{8-j} \otimes I_{dim(\mathcal{P})}
\end{aligned}$$

and are used to select, respectively, the first and last control point from of a Beziér curve of degree $7 - j$ when arranged in a vector.

Now that we have the backbone of the framework set up, we proceed to explain how to deal with static and dynamic obstacles.

3.2.1 Path Planning with Safe Corridors

Let us start by explaining how to plan a trajectory using (3.3) by exploting the properties of Beziér curves. Since we know by Theorem A.2.3 that a Beziér curve are always contained inside the convex hull of its control points, we can ensure the solution of (3.3) to be free of collision by constraining the control point of each segment to be entirely contained into a convex polytope, which results in additional linear constraints on the control points. In fact, a convex polytope $\mathcal{E} \in \mathcal{P}$ with $n_s > 3$ sides can be entirely described by the couple $(E, e) \in \mathbb{R}^{n_s \times dim(\mathcal{P})} \times \mathbb{R}^{n_s}$ as

$$\mathcal{E} = \{p \in \mathcal{P} \mid Ep \leq e\}.$$

So, given a sequence of polytopes[1] $\{\mathcal{E}_1, \ldots, \mathcal{E}_N\}$ such that $p_0 \in \mathcal{E}_1$, $p_F \in \mathcal{E}_N$, and $\mathcal{E}_i \cap \mathcal{E}_{i+1} \neq \emptyset, \forall i \in 1, \ldots, N-1$, we can write (3.3)[2] as

$$\begin{aligned}
\min_{\mathbf{P_1},\ldots,\mathbf{P_N} \in \mathcal{P}^N} \quad & \int_{t_0}^{t_F} J(p(t))dt \qquad (3.4)\\
\textit{subject to} \quad & P_0^1 = p_0, \quad P_7^N = p_F \\
& A_{1,1}\bar{D}_7^1\mathbf{P}_1 = 0, \quad A_{2,1}\bar{D}_7^1\mathbf{P}_N = 0 \\
& A_{1,2}\bar{D}_7^2\mathbf{P}_1 = 0, \quad A_{2,2}\bar{D}_7^2\mathbf{P}_N = 0 \\
& A_{1,3}\bar{D}_7^3\mathbf{P}_1 = 0, \quad A_{2,3}\bar{D}_7^3\mathbf{P}_N = 0 \\
& A_{2,0}\mathbf{P}_i = A_{1,0}\mathbf{P}_{i+1}, \quad \forall i \in \{1, \ldots, N-1\} \\
& \frac{1}{t_i - t_{i-1}} A_{2,1}\bar{D}_7^1\mathbf{P}_i = \frac{1}{t_{i+1} - t_i} A_{1,1}\bar{D}_7^1\mathbf{P}_{i+1}, \quad \forall i \in \{1, \ldots, N-1\} \\
& \frac{1}{(t_i - t_{i-1})^2} A_{2,2}\bar{D}_7^2\mathbf{P}_i = \frac{1}{(t_{i+1} - t_i)^2} A_{1,2}\bar{D}_7^2\mathbf{P}_{i+1}, \quad \forall i \in \{1, \ldots, N-1\} \\
& \frac{1}{(t_i - t_{i-1})^3} A_{2,3}\bar{D}_7^3\mathbf{P}_i = \frac{1}{(t_{i+1} - t_i)^3} A_{1,3}\bar{D}_7^3\mathbf{P}_{i+1}, \quad \forall i \in \{1, \ldots, N-1\} \\
& E_i\mathbf{P}_i \leq e_i, \quad \forall i \in \{1, \ldots, N-1\}
\end{aligned}$$

Quadratic Cost Functions

Since all the constraints of (3.4) are linear, we may want to choose a quadratic cost to take benefit from the fast and efficient algorithms designed to solve Quadratic Programs. Since, as shown in Chapter 2, the control inputs that the quadrotor need to generate to track a C^3 trajectory are proportional to the snap (fourth-order time derivative of the position), we propose, as in [14], the following cost functional:

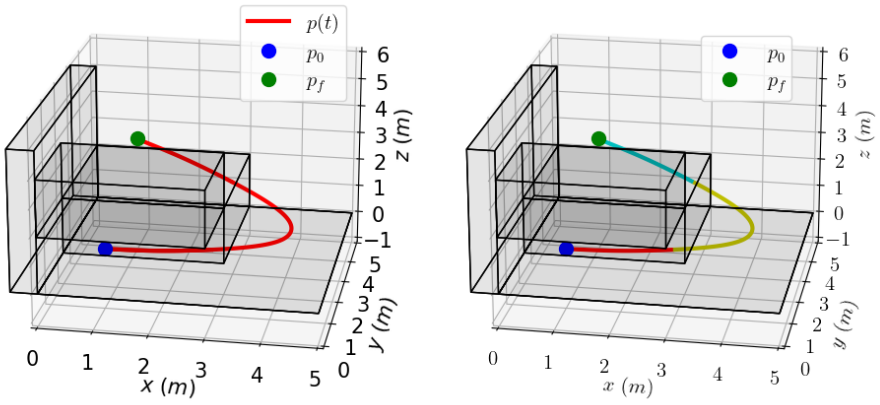
$$J(p(t)) = \frac{d^4 p(t)}{dt^4} \qquad (3.5)$$

which, in our case, makes the cumulative cost a function of only the control points [36, 26]

$$\int_{t_0}^{t_F} J(p(t))dt = \sum_{i=1}^{N} \mathbf{P}_i^T \bar{D}^{4T} \bar{Q}_4 \bar{D}^4 \mathbf{P}_i$$

[1] One can find such a sequence, for example, by building a graph where each node represents a convex polytope and each edge represents an intersection between two polytopes.

[2] Neglecting the moving obstacles constraints, for now.



(a) (b)

Figure 3.1: The figures shows the trajectory generated by 3.4 in a static environments divided in polygonal safe corridors. Figure (a) shows the entire resulting trajectory as a solid red line, while figure (b) shows each of the three Beziérs segments composing the curve, each of them entirely contained in a empty convex polytope.

where the matrix $\bar{Q}_4 \in \mathbb{R}^{4dim(\mathcal{P})\times 4dim(\mathcal{P})}$, $r \in \mathbb{N}$, can be shown to be equal to

$$\bar{Q}_4 = Q_4 \otimes I_{dim(\mathcal{P})}$$
$$Q_4 \in \mathbb{R}^{4\times 4}, [Q_4]_{i,j} = \frac{1}{7}\frac{\binom{3}{i-1}\binom{3}{j-1}}{\binom{6}{i+j-2}}$$

Numerical Results

Figure 3.1 shows the solution of problem (3.4) with the functional cost (3.5). The free space $\mathcal{P} \setminus \mathcal{S}$ has been divided into three polytope regions and three Beziér segments, one for each region, have been used to generate the trajectory.

Chapter 4

Control Barrier Functions for Obstacle Avoidance

The concept of Control Barrier Functions (CBFs) [18] emerged recently in the field of safety-critical systems [17] to design controllers that are mathematically certified to meet safety requirements. The main idea is to divide the state space into safe and unsafe regions. The controller is then constrained by an inequality that ensures the set of safe states remains forward invariant over time.

Enforcing this inequality is generally a challenging task. However, when the system under control can be modeled as an affine control system—such as the quadrotor dynamics and most robotic systems—the inequality becomes linear with respect to the control input. This allows the use of various synthesis techniques for controller design [22, 37, 18]. Among these techniques, Quadratic Program (QP)-based safety filters [18] are the most commonly used, and they will be presented in this chapter.

The aim of this chapter is to give a introduction to the theory of CBFs and to show its applicability to navigations tasks and obstacles avoidance.

4.1 Definitions and Properties

To introduce the concept of CBFs and discuss their properties, we first need to introduce the concepts of *control affine systems*, *Lie Derivatives*, and *class $\mathcal{K}$ functions*.

Definition 4.1.1 (Affine Control System)**.** *Let $x \in \mathcal{X} \subset \mathbb{R}^n$ be the state of a controlled dynamical system whose dynamics is described the differential equation*

$$\dot{x} = f(x) + g(x)u \tag{4.1}$$

where $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}^{n\times m}$ are locally Lipschitz continuous functions, $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input.

Definition 4.1.2 (Lie Derivative)**.** *Let $\mathcal{X} \subset \mathbb{R}$. Given a continously differentiable function $h : \mathcal{X} \to \mathbb{R}$, the* Lie Derivative *$L_f h : \mathcal{X} \to \mathbb{R}$ of h along a vector field $f : \mathcal{X} \to \mathbb{R}^n$ is defined to be*

$$L_f h(x) = \langle \nabla h, f(x) \rangle.$$

For function $f : \mathcal{X} \to \mathbb{R}^{n\times m}$ the Lie Derivative $L_f h : \mathcal{X} \to \mathbb{R}^m$ is instead defined as

$$L_f h(x) = \nabla h f(x).$$

Remark 4.1.1. *As for the gradient, the Lie Derivative $L_f h : \mathcal{X} \to \mathbb{R}^m$ over a multivalued vector field is represented as a row vector for computational purposes.*

If, with an abuse of notation, we denote with $x : \mathbb{R} \to \mathcal{X}$ the solution to the differential equation (4.1) for a given controller $u : \mathcal{X} \to \mathcal{U}$, then the time derivative of the function h along the trajectory $x(t)$ equals the Lie Derivative of h evaluated at $x(t)$, i.e.

$$\begin{aligned}\dot{h}(x(t)) = \langle \nabla h, \dot{x}(t) \rangle &= \langle \nabla h, f(x(t)) \rangle + \nabla h g(x(t)) u(x(t)) \\ &= L_f h(x(t)) + L_g h(x(t)) u(x(t))\end{aligned}$$

Definition 4.1.3. *A function $\alpha : \mathbb{R} \to \mathbb{R}$ is said to be a* class $\mathcal{K}$ function[1] *if it is strictly increasing and $\alpha(0) = 0$.*

To define *safety* in this framework, we rely on division of the state space in a *safe* and *unsafe set* based on the superlevel set of a continously differentiable function, which will become our candidate CBF.

Definition 4.1.4 (Safe Set)**.** *Given a function $h : \mathcal{X} \to \mathbb{R}$, we define the* safe set $\mathcal{C}_h$ *associated with h as the superlevel set of h, that is*

$$\begin{aligned}\mathcal{C}_h &= \{x \in \mathcal{C}_h \mid h(x) \geq 0\} \\ \partial\mathcal{C}_h &= \{x \in \mathcal{C}_h \mid h(x) = 0\} \\ Int\ \mathcal{C}_h &= \{x \in \mathcal{C}_h \mid h(x) > 0\}\end{aligned}$$

We are now ready to define the concept of CBF and discuss about its properties and applications.

Definition 4.1.5 (Control Barrier Function)**.** *A continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$ is called a* Control Barrier Function *for the system (4.1) if there exists a class $\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ and an open set $\mathcal{D} \supseteq \mathcal{C}_h$ such that, for each $x \in \mathcal{D}$*

$$\sup_{u \in \mathcal{U}} L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0 \tag{4.2}$$

Remark 4.1.2. *Assuming the existence of a open set $\mathcal{D}$ containing $\mathcal{C}_h$ will be useful to prove some robustness property of the CBFs.*

Definition 4.1.6 (Safe Controllers)**.** *Let h be a CBF for the system (4.1) for a class $\mathcal{K}$ function α. Let us the define the following set*

$$K_{h,\alpha}(x) = \left\{u \in \mathcal{U} \mid L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0\right\}.$$

Then, a controller $u : \mathcal{X} \to \mathcal{U}$ is said to be a safe controller *if*

$$u(x) \in K_{h,\alpha}(x), \forall x \in \mathcal{D}$$

Remark 4.1.3. *Notice that Equation (4.2) guarantees that the set $K_{h,\alpha}(x)$ is not empty for each $x \in \mathcal{X}$.*

The previous definition is justified by the following theorem, which is the main building block of the CBF paradigm.

[1] Sometimes, the literature refers to them as *extended class $\mathcal{K}$ functions, and restrict the definition of class $\mathcal{K}$ functions to functions $\alpha : \mathbb{R}_+ \to \mathbb{R}$*

Theorem 4.1.1. *Given a control barrier function h for the affine system (4.1) any safe controller $u(x)$ renders the set $\mathcal{C}_h$ forward invariant.*

Proof. Since $u(x) \in K_{h,\alpha}(x)$ for each $x \in \mathcal{D}$, we have $L_f h(x) + L_g h(x)u(x) \geq -\alpha(h(x))$, $\forall x \in \mathcal{C}_h$. This implies that, for any $x_0 \in \mathcal{C}_h$, the time derivative of h along the solution $x : \mathbb{R} \to \mathcal{X}$ of (4.1) with initial condition $x(t_0) = x_0$ satisfies,

$$\dot{h}(x(t)) = L_f h(x(t)) + L_g h(x(t))u(x(t)) \geq -\alpha(h(x(t))).$$

Now, since the solution $\bar{h}(x(t))$ of the differential equation $\dot{\bar{h}}(x(t)) = -\alpha(\bar{h}(x(t)))$ with initial condition $\bar{h}(x(t_0)) = h(x(t_0)) \geq 0$ satisfies both

$$\bar{h}(x(t)) > 0, \quad \forall t \geq t_0$$

and

$$\lim_{t \to +\infty} \bar{h}(x(t)) = 0,$$

so, for the comparison lemma [38], we have that $h(x(t)) \geq \bar{h}(x(t)) > 0$, $\forall t \geq t_0$ which means $x(t) \in \mathcal{C}_h$, $\forall t \geq t_0$, i.e. $\mathcal{C}_h$ is forward invariant. □

Before proceeding, we want to highlight the following important property of CBFs.

Theorem 4.1.2. *Given a control barrier function h for the affine system (4.1) any safe controller $u(x)$ renders the set $\mathcal{C}_h$ asymptotically stable for each $x \in \mathcal{D}$.*

Theorem 4.1.2 basically implies that in the case the controlled system finds itself in an unsafe state, the safe controller is able to quickly recover (exponentially fast, actually) and bring the state back to a safe state. The proof of the theorem, based on a simple Lyapunov argument, can be found in [39].

4.1.1 Quadratic Program Safety Filter

Thanks to Theorem 4.1.1 we know that any safe controller keeps the state in the safe set $\mathcal{C}_h$ for a given CBF h. To find a controller $u(x) \in K_{h,\alpha}(x)$ the standard approach is to define a generic control law $u^* : \mathcal{X} \to \mathcal{U}$, which can be used for example to stabilize (4.1) around an equilibrium point or a trajectory, and then to solve the following quadratic program

$$\begin{aligned} \min_{u \in \mathcal{U}} \quad & \|u - u^*(x)\|_2^2 \\ \textit{subject to} \quad & L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0 \end{aligned} \tag{4.3}$$

4.1.2 CLFs

Definition 4.1.7 (Control Lyapunov Fuction)**.** *A continuously differentiable function $V : \mathcal{X} \to \mathbb{R}$ is called a* Control Lyapunov Function *(CLF) for an equibrium point $x^* \in \mathcal{X}$ of the system* (4.1) *if there exist three class $\mathcal{K}$ functions $\gamma, \gamma_1, \gamma_2 : \mathbb{R} \to \mathbb{R}$ such that for every $x \in \mathcal{X}$*

- $\gamma_1\left(\|x - x^*\|_2^2\right) \leq V(x) \leq \gamma_2\left(\|x - x^*\|_2^2\right)$
- $\inf_{u \in \mathcal{U}} L_f V(x) + L_g V(x) + \gamma(V(x)) \leq 0, \forall x \neq x^*$

While CBF can be used as showed in equation (4.5) to render a set forward invariant in time, CLF can be used in the same way to stabilize an affine control system like (4.1) to the corresponding equilibrium point. That is, the control law resulting from the repetitive solution of

$$\begin{aligned} \min_{u \in \mathcal{U}} \quad & \|u\|_2^2 \\ \textit{subject to} \quad & L_f V(x) + L_g V(x) u + \gamma\left(V(x)\right) \leq 0 \end{aligned}$$

is a stabilizing control law for the equilibrium point of the system.

The CBF and CLF constraints can be combined in a single quadratic optimization problem to obtain a stabilizing and safe control law in a so called CBF-CLF-QP control scheme as

$$\begin{aligned} \min_{u \in \mathcal{U}} \quad & \|u\|_2^2 \\ \textit{subject to} \quad & L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq 0 \\ & L_f V(x) + L_g V(x) u + \gamma\left(V(x)\right) \leq 0 \end{aligned} \tag{4.4}$$

However, the two constraints in (4.4) may conflict with each other which may cause the optimization problem to be infeasible. To deal with this problem, in practical applications it is a common practice to add two slack variables to the CBF and CLF constraints to allow for constraints violation, e.g.

$$\begin{aligned} \min_{\substack{u \in \mathcal{U} \\ \delta_V, \delta_h \in \mathbb{R}}} \quad & \|u\|_2^2 + w_V \delta_V^2 + w_h \delta_h^2 \\ \textit{subject to} \quad & L_f h(x) + L_g h(x) u + \alpha(h(x)) \geq \delta_h \\ & L_f V(x) + L_g V(x) u + \gamma\left(V(x)\right) \leq \delta_V \end{aligned} \tag{4.5}$$

where $w_V, w_h \in \mathbb{R}_+$ are two tunable weights that can be used to prioritize either safety or stability in the generation of the control law.

4.1.3 Multiple Constraints

In real world scenarios there may be multiple safety requirements that needs to be satisfied at once. To address this issue, one may define a different CBF for each requirement and add a constraint to problems (4.3) and (4.5) for each function. However, this approach has a problem: two safety constraints may conflict with each other, leading to the infeasibility of the quadratic program.

A common solution to this problem is to add some slack variables to the constraints of the quadratic program to guarantuee the feasibility of the optimization problem. Let us assume, for example, to have $N > 1$ CBFs $h_i : \mathcal{X} \to \mathbb{R}$, $i \in \{1, \ldots, N\}$, then we can modifiy problems (4.3) and (4.5), respectively, as follows

$$\begin{aligned} \min_{\substack{u \in \mathcal{U} \\ \rho_1, \ldots, \rho_N \in \mathbb{R}}} \quad & \|u\|_2^2 + \sum_{i=0}^{N} l_i \rho_i^2 \\ \textit{subject to} \quad & L_f h_i(x) + L_g h_i(x) u + \alpha(h_i(x)) \geq \rho_i \forall i \in \{1, \ldots, N\} \end{aligned} \tag{4.6}$$

$$\begin{aligned} \min_{\substack{u \in \mathcal{U}, \delta \in \mathbb{R} \\ \rho_1, \ldots, \rho_N \in \mathbb{R}}} \quad & \|u\|_2^2 + w\delta^2 + \sum_{i=0}^{N} l_i \rho_i^2 \\ \textit{subject to} \quad & L_f h_i(x) + L_g h_i(x) u + \alpha(h_i(x)) \geq \rho_i, \forall i \in \{1, \ldots, N\} \\ & L_f V(x) + L_g V(x) u + \gamma\left(V(x)\right) \leq \delta \end{aligned} \tag{4.7}$$

N° Constraints	Computation Time
1	1.48×10^{-5} s
5	1.52×10^{-5} s
10	1.58×10^{-5} s
50	4.07×10^{-5} s
100	3.01×10^{-4} s
500	2.26×10^{-2} s
1000	1.65×10^{-1} s

Table 4.1: Average computation time of the quadprog routine to solve a quadratic program with variable number of constraints.

where ρ_i are slack variables that ensure that a solution of the quadratic program always exists, while $l_i \in \mathbb{R}_{++}$ are relative weights used to discourage large values of the slack variables, which may lead to unsafe scenarios. The weights l_i may be chosen, a priori or following some logic during the execution of the task, to give an order of priority to safety requirements.

Remark 4.1.4. *Problem* (4.6) *and* (4.7) *can be solved very efficiently with off-the-shelf solvers such as* quadprog[2]*,* gurobi[3]*,* qpOASES[4]*, and many others. Nonetheless, the designer must take into consideration the time required by the selected solver when implementing the safety filter, which scales with the number of constraints of the optimizazion problem. For example, Table 4.1 shows the mean computation time of the quadprog solver on a test platform for a variable number of constraints. We can see that the computation time is on the order of* 10^{-5} *s for less than 50 constraints, which is compatible with the frequency of control of execution of control commands of common robotic application. However, we can clearly see an exponential like growth of the computation time as the number of constraints increases, which prevents a practical implementation of the control law generated by* (4.6) *and* (4.7) *with an excessive number of contraints.*

4.1.4 Application Example: Obstacle Avoidance

We now present a little example to show how the theory presented so far in this chapter could be applied to a real world application. The scenario we want to present is that of a robot moving in the plane from a point A to a point B that needs to avoid any collision with its environment. So let us set up all the mathematical objects we need to describe the our scenario and we will then explain how to practically implement a CBF safety filter to prevent any collision.

Let us consider an agent moving on the plane, whose position is denoted by $p \in \mathbb{R}^2$ and let us assume that this agent can be modeled a massless point whose velocity can be directly controlled. This means the state x of the agent is equal to p and that the differential equation describing the evolution of the system is $\dot{x} = u$. Suppose moreover that there is an obstacle $\mathcal{O} \in \mathbb{R}^2$ whose profile can encapsuled by a circle centered at $o = \begin{bmatrix}1, 2\end{bmatrix}^T$ with radius $r = 2$, i.e. $\mathcal{O} \subset \mathcal{B}^2(o, r)$. In these terms, a collision corresponds to the condition $x \in \mathcal{O}$, which then implies that our desired safe set corresponds to $\mathbb{R}^2 \setminus \mathcal{O}$, which is the set of points in the plane that is not occupied by the obstacle.

Our goal now is to find a CBF h such that $\mathcal{C}_h \subset \mathbb{R}^2 \setminus \mathcal{O}$, that means, the safe set corresponding to the chosen continously differentiable function does not need to be the entire set of safe configurations,

[2]https://github.com/quadprog/quadprog
[3]https://www.gurobi.com/
[4]https://github.com/coin-or/qpOASES

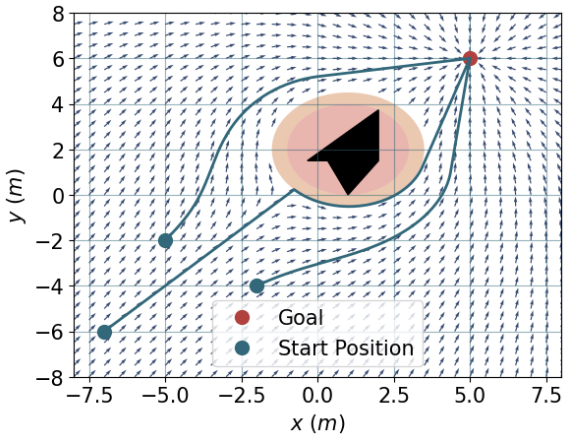


Figure 4.1: The figure shows an agent navigating around an obstacle using a safety filter based on CBF. We can see how the vector field of the closed loop system wrap around the obstacle.

but it is sufficient that it is contained in it. A family of functions that are compliant with our specification is

$$h(x) = \|x - o\|_2^2 - (r + r_s)^2$$

for a given $r_s > 0$, which represent a *safety radius* that can be used to take into account the size of the robot. The safe set $\mathcal{C}_h$ in this case is equivalent to the set $\{x \in \mathbb{R}^2 \mid \|x - o\|_2 \geq r + r_s\}$.

Now that we have a CBF candidate, we need to compute $L_f h$ and $L_g h$. And, in our case, since $f(x) = 0$ and $g(x) = I_(2)$ we have $L_f h = 0$ and $L_g h u = \langle x - o, u \rangle$. So, we now have all the elements to implement the quadratic program safety filter shown in equation (4.3).

Figure 4.1 shows an agent navigating with the proposed safety filter, with $\alpha(h) = 2h$, around an obstacle to reach the goal position $x^* = [6, 5]^T \in \mathbb{R}^2$, whose nominal control law is $u^* = -k_p(x - x^*)$, $k_p = 2$, starting from different initial positions. The figure shows also the vector field $\dot{x}$ resulting from the application of the closed loop control law 4.1. We can see how the vector field wrap around the obstacle, forcing the agent to navigate around it.

To use (4.5) instead of (4.3), we need to find a valid candidate CLF. In this case, it is easy to show that $V(x) = \frac{1}{2}\|x - x^*\|_2^T$ makes x^* a stable equilibrium point for the closed loop system. By noticing that $L_f V(x) = 0$ and $L_g V(x) = x - x^*$, we can easily implement (4.5).

Figure 4.2 shows an agent navigating using (4.5) as a control law, with $\alpha = 10$, $\gamma = 3$ and $w = 10$. The figure shows also the closed loop vector field $\dot{x}$ that, as in previous case, wrap around the obstacle forcing the agent to navigate around it. We can also see that the agent is able to recover from unsafe state and safely navigate to the goal, as predicted by Theorem 4.1.2.

4.2 High Order Control Barrier Functions

Let us consider again the example presented in the previous section, but, this time, let us suppose that our control authority is only on the acceleration of the robot, and not on its velocity. In this case the state of the robot is composed by both its position p and its velocity $v \in \mathbb{R}^2$, i.e. $x = [p, v]^T$,

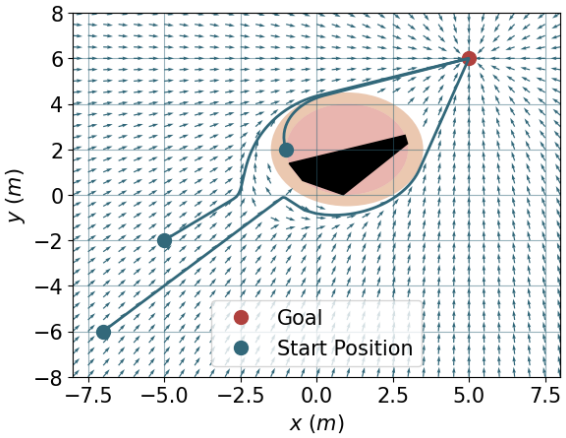


Figure 4.2: The figure shows an agent controlled with a combination of Control Barrier and Control Lyapunov function navigating around an obstacle. We can see how the vector field of the closed loop system wrap around the obstacle.

and its dynamics is described by the following differential equation:

$$\begin{cases} \dot{p} = v \\ \dot{v} = u \end{cases} \tag{4.8}$$

and $u \in \mathbb{R}^2$ is again our control input. If we use again the function $h(x) = \|x - o\|_2^2 - (r + r_s)^2$ as our canditate CBF, we immediately see that there is a problem: $\dot{h} = 2\langle p - o, v \rangle$, which means $L_f h = 2\langle p - o, v \rangle$ and $L_g h = 0$. So this function is not a valid CBF according to defition 4.1.5, despite it encodes the safety specification as intended.

One way to get around this problem is to directly design a valid CBF that depends on the entire state of the system, so that the control input appear in the expression of its first derivative; and this is the approach followed for example by the authors of [40], where they proposed an inter-agent collision avoidance system based on the current velocity of the agents. However, in most application scenario, designing this kink of ad-hoc full state dependent Control Barrier Fuction is not a straightforward task, and it is not the affirmed approach used in practice.

The standard solution to this problem is an iterative procedure that extend Definition 4.1.5 and Theorem 4.1.1 to higher order time derivative to make the effect of the input visible and take the name of *High Order Control Barrier Function*. To get an intution of this technique before presenting the abstract description, let us again discuss the previous example. According to the comparison lemma cited in the proof of Theorem 4.1.1, for any $t_0 \in \mathbb{R}$, the conditions $x(t_0) = x_0$, $h(x_0) \geq 0$, and $\dot{h}(x(t)) + \alpha(h(x(t))) \geq 0$ for all $t \geq t_0$ implies $h(x(t)) \geq t_0$ for all $t \geq t_0$. So, if we were able to impose $\dot{h}(x(t)) + \alpha(h(x(t))) = L_f h(x(t)) + \alpha(h(x(t))) \geq 0$ we could use again the comparison lemma to guarantee safety. This suggests us to define the following auxiliary canditate CBF

$$\psi(x) = L_f h(x) + \alpha(h(x))$$

The intuition here is that if we are able to find a controller $u(x) \in K_{h,\beta}(x)$ for a given class $\mathcal{K}$ function $\beta : \mathbb{R} \to \mathbb{R}$, then by Theorem 4.1.1 we would have $\psi(x(t)) \geq 0$ for each $t \geq t_0$, which in turns will imply the safety $h(x(t)) \geq 0$ by the comparison lemma. We can summarize our discussion

with the implication

$$\mathcal{C}_\psi \ forward\ invariant \implies \mathcal{C}_h \ forward\ invariant.$$

Moreover, since $L_f\psi = \langle v, v\rangle + \frac{d\alpha}{dh}\langle p-o, v\rangle$ and $L_g\psi u = \langle p-o, u\rangle$, $\psi(x)$ is a valid canditate CBF.

We will now better formalize the above discussion to deal with more general cases.

Definition 4.2.1 (Relative Degree)**.** *Given an affine system* (4.1) *and a function* $h : \mathcal{X} \to \mathbb{R}$*, the* relative degree *at* $x \in \mathcal{X}$ *is said to be* $r \in \mathbb{N}$ *if there exists an open set* $\mathcal{O}$ *that contain* x *such that*

- h *is continuously differentiable at least* r *times in* $\mathcal{O}$
- r *is the smallest number such that*

$$L_g L_f^{r-2} h(x) u = 0 \tag{4.9}$$
$$L_g L_f^{r-1} h(x) u = 0 \tag{4.10}$$

We say that a function h *has an* omogeneous relative degree *over the set* $\mathcal{W} \subseteq \mathcal{X}$ *if it has the same relative degree for every* x *contained in* $\mathcal{W}$*.*

Definition 4.2.2 (High Order Control Barrier Functions)**.** *Let* $h : \mathcal{X} \to \mathbb{R}$ *be a function with omogeneous relative degree* r *with respect to* (4.1) *over* $\mathcal{X}$*. Let* $\alpha_i : \mathbb{R} \to \mathbb{R}$ *be* r *class* $\mathcal{K}$ *functions and let us defined the following family of functions*

$$\begin{cases} \psi_0(x) = h(x) \\ \psi_i(x) = L_f\psi_{i-1}(x) + \alpha_i(\psi_{i-1}(x)), \forall i \in \{1, \dots, r-1\} \end{cases}$$

We call h *a* Higher Order Control Barrier Function *(HOCBF) if for each* $x \in \bigcap_{i=0}^{r-1} \mathcal{C}_{\psi_i}$

$$\sup_{u \in \mathcal{U}} L_f\psi_{r-1}(x) + L_g\psi_{r-1}(x)u + \alpha_r(\psi_{r-1}(x))$$

or, equivalently,

$$\sup_{u \in \mathcal{U}} L_f^r h(x) + L_g L_f^{r-1} h(x) u + O(x) + \alpha_r(\psi_{r-1}(x)) \tag{4.11}$$

where $O(x) = \sum_{i=1}^{r} L_f^i \alpha_{r-i}(\psi_{r-i-1}(x))$*.*

Definition 4.2.3 (High Order Safe Controllers)**.** *Let* h *be an HOCBF with relative degree* $r \in \mathbb{N}_0$ *for the system (4.1) for a collection of class* $\mathcal{K}$ *functions* $\mathcal{A} = \{\alpha_i\}_{i=0}^{r}$*. Let us the define the following set*

$$K_{h,\mathcal{A}}(x) = \left\{ u \in \mathcal{U} \mid \sup_{u \in \mathcal{U}} L_f^r h(x) + L_g L_f^{r-1} h(x) u + O(x) + \alpha_r(\psi_{r-1}(x)) \geq 0 \right\}.$$

Then, a controller $u : \mathcal{X} \to \mathcal{U}$ *is said to be a* high order safe controller *if*

$$u(x) \in K_{h,\mathcal{A}}(x), \forall x \in \bigcap_{i=0}^{r-1} \mathcal{C}_{\psi_i}$$

Theorem 4.2.1. *Given a HOCBF* h *for the affine system (4.1) any high order safe controller* $u(x)$ *renders the set* $\mathcal{C}_h$ *forward invariant.*

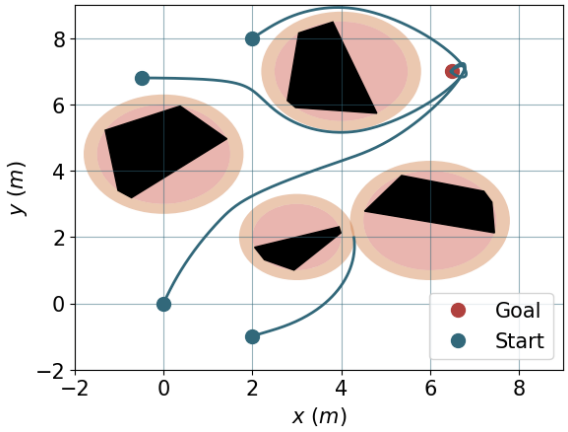


Figure 4.3: The figure shows an agent modeled as a double integrator navigating using a HOCBF based safety filter to reach a goal position in the presence of obstacles in the workspaces.

Proof. Since $u(x) \in K_{h,\mathcal{A}}(x)$, by Theorem 4.1.1 the set $\mathcal{C}_{\psi_{r-1}}$ is forward invariant in time. Using the comparison lemma, it can be proven that, for each $i \in \{0, \dots, r-2\}$,

$$\mathcal{C}_{\psi_{i+1}} \ \textit{forward invariant} \implies \mathcal{C}_{\psi i} \ \textit{forward invariant}.$$

So we can conclude that $\mathcal{C}_{\psi_0} = \mathcal{C}_h$ is forward invariant. □

Armed with the presented theoretical tool, we are now able to adapt (4.3) and (4.5) to the case of HOCBF. For concisiness we show only the adaptation of (4.3) which becomes

$$\begin{aligned} \min_{u \in \mathcal{U}} \quad & \|u - u^*(x)\|_2^2 \\ \textit{subject to} \quad & L_f^r h(x) + L_g L_f^{r-1} h(x) u + O(x) + \alpha_r(\psi_{r-1})(x) \geq 0 \end{aligned} \tag{4.12}$$

4.2.1 Application Example: Obstacle Avoidance

We now present a numerical simulation showcasing the use of HOCBF for obstacle avoidance. Figure 4.3 shows an agent, modeled as in equation (4.8) navigating from a several different initial conditions to the goal state $x^* = [p^*, v^*]^T$, with $p^* = [6.5, 7]^T$ and $v^* = [0, 0]^T$, using the nominal control law $u^*(x) = -k_p(p - p^*) - k_v(v - v^*)$, with $k_p = -10.5$ and $k_v = 4$, filtered by the safety filter (4.12), with $\alpha_1(h) = 5h$ and $\alpha_2(h) = 3h$. We can see that the agent successfully avoid all the obstacles. We can also see that, depending on the initial condition, the agent may not converge to the goal state but remains stuck at the edge of the safe set.

Chapter 5

Safe Navigation in Unknown Environments

In this chapter we investigate the use of Control Barrier Function paradigm and Gaussian Processes to design safe navigation strategies in unknown environments. We start by presenting a direct approach to construct a Control Barrier Function from observed data using Gaussian Process regression and we highlight some of its limitations. We then present a framework based on surface reconstruction [41] that may solve some of the previously presented issues in the case of safe navigation. Lastly, we summarize the work presented in [42] to fit High Order Control Barrier Function constraints using the properties of Gaussian Processes and High Gain Observers.

5.1 Navigation with Gaussian Processes

In this section we show a simple approach to create an environment representation suitable for the the Control Barrier Function [43, 44, 45] paradigm using *Gaussian Process Regression* as our learning approach. Gaussian Process regression is a supervised learning technique that can be used both for regression and classification tasks [46] and, with respect to other learning approaches, can provide both probabilistic[47] and worst case errors bounds[48]. The main elements and concept behind the theory of Gaussian Processes are recalled in Appendix B.

Before delving into the theory, let us introduce the main elements of the discussion. We will denote with $\mathcal{P}$ the workspace, where an agent with position $p \in \mathcal{P}$ lives, and we will identify $\mathcal{P}$ $\mathbb{R}^3$. The velocity and the acceleration of the agent will be denoted, respectively, with $v \in T_p\mathcal{P} = \mathbb{R}^3$ and $a \in T_p\mathcal{P} = \mathbb{R}^3$. In our discussion we will consider the two following dynamical models to describe the dynamics of the agent:

$$\dot{p} = v \tag{5.1}$$

$$\begin{cases} \dot{p} = v \\ \dot{v} = a \end{cases} . \tag{5.2}$$

Notice that both of the above cases are control affine systems such as (4.1). In particular, in (5.1) we have $\mathcal{X} = \mathbb{R}^3$, $x = p$, and $u = v$, while in (5.2) we have $\mathcal{X} = \mathbb{R}^3 \times \mathbb{R}^3$, $x = [p, v]^T$, and $u = a$

In the workspace are placed some obstacles and the set of the points occupied by the obstacles

will be denoted with $\mathcal{O} \subset \mathcal{P}$, which we will assume to be compact. Let

$$d_{\mathcal{O}}(p) = \begin{cases} \min_{q \in \partial\mathcal{O}} \|p - q\|_2, p \notin \mathcal{O} \\ -\min_{q \in \partial\mathcal{O}} \|p - q\|_2, p \in \mathcal{O} \end{cases}$$

denote the *Signed Euclidean Distance* from the obstacles in the workspace. Ideally, we would like to use $d_{\mathcal{O}}$ as our candidate Control Barrier Function, but, in our framework, we will assume $d_{\mathcal{O}}$ to be initially unknown but measurable trough some range sensor at every $p \in \mathcal{P}$.

To construct an approximation of $d_{\mathcal{O}}$ we will rely on the Gaussian Process Regression theory (see Section B.2.2). Let $\mathcal{D} = \{(p_i, y_i)\}_{i=1}^{N}$ denote a set of $N \in \mathbb{N}_0$ tuples containing the position of the agent along with the distance measurements given by

$$y_i = d_{\mathcal{O}}(p_i) + \xi_i \tag{5.3}$$

where ξ_i is a zero mean white noise with variance $\sigma^2 > 0$, as in Section B.2.2. Let $X = p_1, p_2, \ldots, p_N$ denote the set of sample points and $Y = [y_1, \ldots, y_N]^T$ denote the vector containing the distance measurements. Let $k : \mathcal{P} \times \mathcal{P} \to \mathbb{R}$ be the square exponential kernel defined in equation (B.1) and $K_X \in \mathbb{R}^{N \times N}$ the corresponding Kernel matrix (see Definition B.1.2). Then the posterior mean $\mu : \mathcal{P} \to \mathbb{R}$ of the regressor function based on the dataset $\mathcal{D}$, assuming a zero prior mean, is given by:

$$\mu(p) = k_X^T \left(K_X + \sigma^2 I_N\right)^{-1} Y \tag{5.4}$$

where $k_X = [k(p, p_1), \ldots, k(p, p_N)]^T$.

We will now investigate the use of equation 5.4 for the construction of candidate Control Barrier Functions for autonomous navigation.

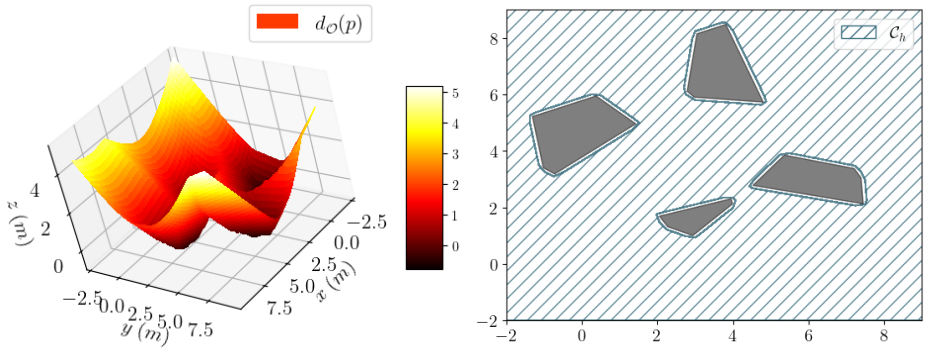
5.1.1 Full Environment Knowledge

To understand the potential and possible drawback of using equation 5.4 as a proxy for the Signed Euclidean Distance, let us first take a qualitative look at his behaviour in a scenario where we already know the exact value of $d_{\mathcal{O}}$ at every point in the workspace, shown in Figure 5.1.

Since during the navigation the agent is able to sample point only outside the obstacles, to estimate the distance function we sampled the environment twice in the set $\mathcal{R} = [-2.0, 8.5] \times [-2.08.5] \subset \mathbb{R}^2 \setminus \mathcal{O}$ to create two dataset and compare two sample densities: $\mathcal{D}_{50}$ which contains the samples taken at points $p = (0.5i, 0.5j) \in \mathcal{R}$ for $i, j \in \mathbb{Z}$, and $\mathcal{D}_{25}$ which contains the samples taken at points $p = (0.25i, 0.25j) \in \mathcal{R}$ for $i, j \in \mathbb{Z}$. Figure 5.2 and Figure 5.3 shows the resulting posterior mean μ on the two respective datasets along with the safe associated safe set $\mathcal{C}_h$ for $h(x) = \mu(p)$. We can see that both regressors give, qualitatively, a good approximation of the distance function $d_{\mathcal{O}}$, but in both cases the resulting safe set intersect the obstacle set, which may results in a collision when used as Control Barrier Functions. A possible workaround is showed in Figure 5.4 where er chose $h(x) = \mu(p) - 0.3$ to restrict the dimension of the safe set, where the value subtracted to the regresson have been chosen empirically.

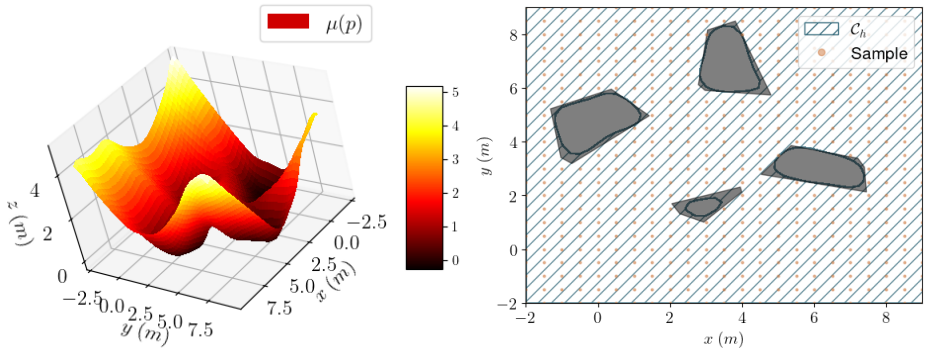
5.1.2 Navigation in Unknown Environments

We now show some simulated scenario showing an agent starting from $p = [0, 0]$ navigating toward one or more goal points $p^* \in \mathcal{P}$ in an unknown environment, while recontructing the distance function $d_{\mathcal{O}}$ by collecting a new sample each time the actual position is more than 0.25 m from all



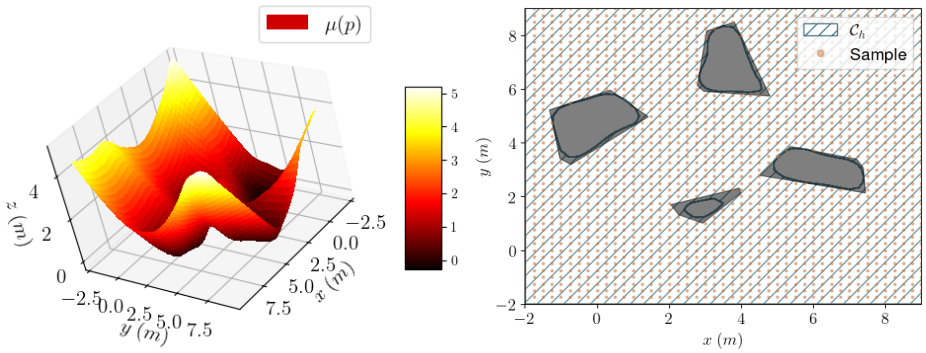
(a) (b)

Figure 5.1: The figures show the *Signed Euclidean Distance Field* of the workspace. Figure (a) shows the tridimensional plot of the function $d_{\mathcal{O}}(p)$, while figure (b) shows the safe set associated with the function $h(x) = d_{\mathcal{O}}(p) - 0.1$.



(a) (b)

Figure 5.2: The figures shows the posterion mean function $\mu(p)$ obtained on the dataset $\mathcal{D}_{50}$. Figure (a) shows the tridimensional plot of $\mu(p)$, while figure (b) shows the safe set associated with the function $h(x) = \mu(p)$. We can see a superposition between the obstacle set $\mathcal{O}$ and the safe set $\mathcal{C}_h$.



(a) (b)

Figure 5.3: The figures shows the posterion mean function $\mu(p)$ obtained on the dataset $\mathcal{D}_{25}$. Figure (a) shows the tridimensional plot of $\mu(p)$, while figure (b) shows the safe set associated with the function $h(x) = \mu(p)$.We can see a superposition between the obstacle set $\mathcal{O}$ and the safe set $\mathcal{C}_h$.

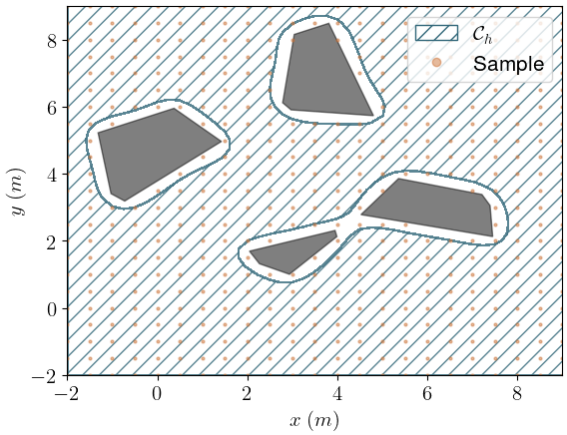
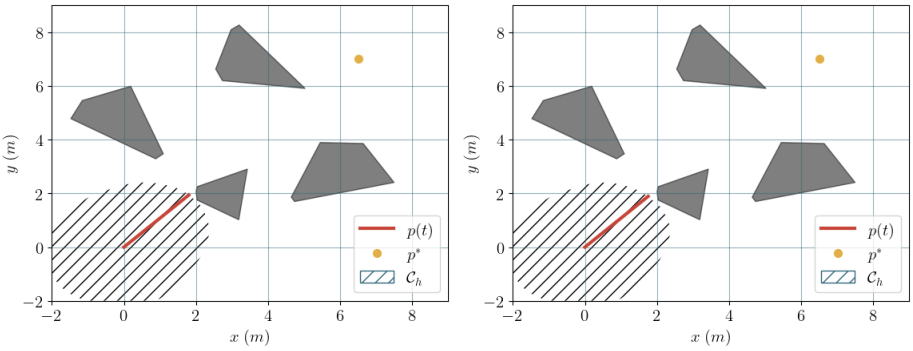


Figure 5.4: The figure shows the safe set $\mathcal{C}_h$ associated with the function $h(x) = \mu(p) - 0.3$, where μ is the posterion mean obtained on the dataset $\mathcal{D}_{50}$



(a) (b)

Figure 5.5: The figures show an agent navigating to a goal in an unknown environment while collecting samples of $d_{\mathcal{O}}(p)$ to learn the Control Barrier Function. In figure (a) the agent is modeled as a single integrator, while in figure (b) the agent is modeled as double integrator. In both cases the agent get stuck after a few seconds trying to reach the goal on the other side of the workspace.

the other sampling points. The agent is modeled either following (5.1) or (5.2), and the candidate Control Barrier Function is chosen to be $h(x) = \mu(p) - 0.3$. In the first case, the nominal control law of the agent is given by $u^*(x) = -k_p(p - p^*)$, with $k_p = 5$, and the class $\mathcal{K}$ function for the safety filter is the constant multiplier $\alpha = 2.5$. In the second case, the nominal control law of the agent is given by $u^*(x) = -k_p(p - p^*) - k_v v$, with $k_p = 5$ and $k_v = 3$, and the class $\mathcal{K}$ functions for the safety filter are the constant multiplier $\alpha_1 = 2.5$ and $\alpha_2 = 2.5$.

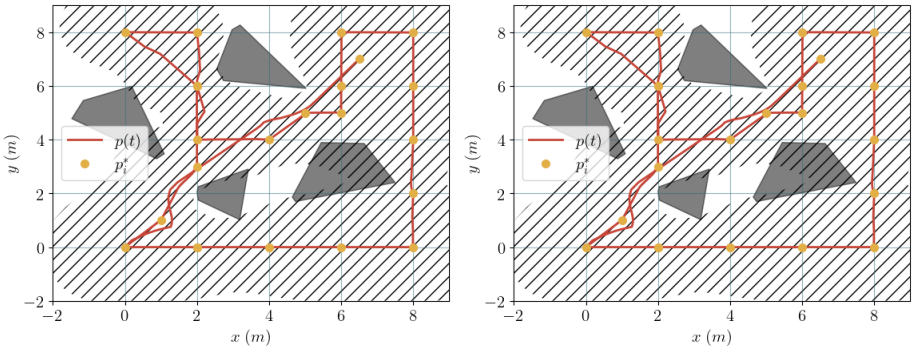
In the first scenario, showed in Figure 5.5, the agent is tasked to reach a goal point on the other side of the workspace located at $p^* = [6.5, 7.0]^T$. We can see that the agent is able to avoid the collision with the obstacles, but it cannot reach the desired goal position and it quickly reaches the boundary of its safe set. That suggests us that, for the agent to be able to navigate in its environment, one first need to design an exploration strategy to sample as much as possible of the workspace.

The second scenario, depicted in Figure 5.6, shows the agent navigating trough a series of hand picked waypoints $p_i^* \in \mathcal{P}$, $i \in \{1, \dots, n_w\}$ with $n_w \in \mathbb{N}_0$, chosen far enough from the obstacles. The figure shows, along with the followed trajectories and the waypoints, the reconstructed safe set at the end of the task and we can clearly see that, due to the non complete coverage of all the workspace, the safe set intersect with the obstacle and this is an critical safety issue. To show the potential danger, we repeated the simulation by changing the location of the waypoints to include a target at $[6, 2]^T$ after a waypoint at $[6, 4]^T$. We can see the results in Figure 5.7 and 5.8 where it is clear that, due to an initiall wrong estimate of the safe set, the agent end up crushing with an obstacle.

Despite its potentiality, we showed that this approach has some critical flaws that need to be addressed. In the next section we will propose an approach, based again on Gaussian Process regression, that aims at addressing the issues that have arisen in these simulations.

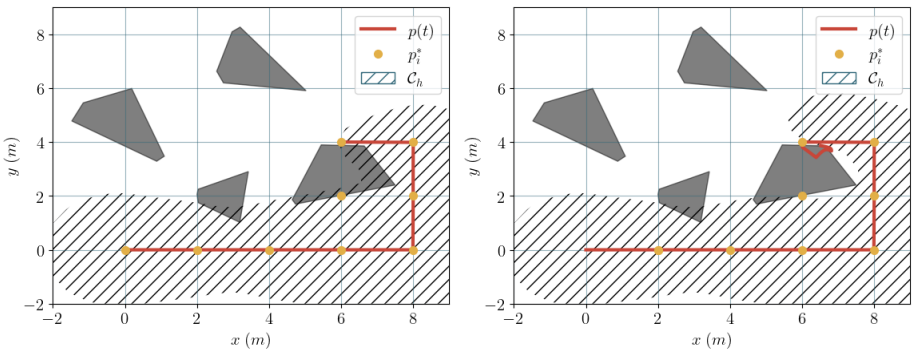
5.2 Navigation with log-GPIS

In this section we introduce an alternative technique to reconstruct $d_{\mathcal{O}}$ using Gaussian Process regression called *log-Gaussian Process Implicit Surface* (log-GPIS)[41, 49], and we propose its use to construct Control Barrier Functions for safe autonomous navigation.



(a) (b)

Figure 5.6: The figures show an agent navigating trough a set of carefully selected waypoints p_i^* while collecting samples of $d_{\mathcal{O}}(p)$. In figure (a) the agent is modeled as a single integrator, while in figure (b) the agent is modeled as a double integrator. We can see clearly see that, due to the low number of collected samples, we have $\mathcal{O} \cap \mathcal{C}_h$, and this may lead to a collision.



(a) (b)

Figure 5.7: The figures show a failure case of the proposed approach. By overestimating the dimension of the safe set, the agent crashes on the obstacle and tries to navigate on it before collecting a new sample and realizing the distance got negative.

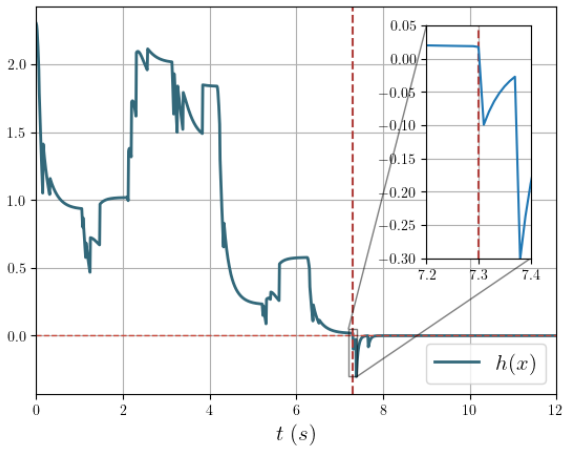


Figure 5.8: The figure shows the value of the barrier function $h(x) = \mu(p)$ over time and gives more insight to explain why the agent crushed with the obstacle. We can see at at time $t = 7.3s$, after collecting a new distance sample, the estimate got negative, implying the agent had already crushed.

To begin with, notice that the Euclidean distance function $d_{\mathcal{O}}$ could be characterized [50] by the following, non linear, differential equation with boundary contraints

$$\begin{cases} \|\nabla\ d_{\mathcal{O}}(p)\|_2 = 1, \forall p \in \mathcal{P} \\ d_{\mathcal{O}}(p) = 0, p \in \partial\mathcal{O} \\ \frac{\partial d_{\mathcal{O}}}{\partial \hat{n}} = 1, p \in \partial\mathcal{O} \end{cases} \tag{5.5}$$

where $\hat{n} \in T_p\mathcal{P}$ is the unit vector perpendicular to $\partial\mathcal{O}$ at p.

Finding an exact solution to (5.5), also known in the literature as the *eikonal equation*, is in general an hard problem due to the non linearity of the $\|\cdot\|_2$ operator, but one can find an approximate solution using the *Varadhan's distance formula* [51] which approximates the function $d_{\mathcal{O}}$ using the *heat kernel* on the set $\mathcal{O}^C$ as briefly summarized in the following discussion.

The heat condution on the set $\mathcal{O}$ can be modeled by the heat kernel $\Upsilon : \mathcal{O}^C \to \mathbb{R}$ solution to the following differential equation, known in the literature as the *screened Poisson equation*,

$$\begin{cases} \Upsilon(p) - t\Delta\Upsilon(p) = \left(\frac{1}{t} - \Upsilon(p)\right) = 0, \forall p \in Int\ \mathcal{O}^C \\ \Upsilon(p) = 1, \forall p \in \partial\mathcal{O} \end{cases} \tag{5.6}$$

where $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$ is the Laplace operator and $t \in \mathbb{R}$ denotes time. Then, it is been proven in [51] the distance function $d_{\mathcal{O}}$ and the heat kernel Υ are related, for each $p \in \mathcal{O}^C$, by

$$d_{\mathcal{O}}(p) = \lim_{t \to 0} -\sqrt{t} \ln \Upsilon(p) \tag{5.7}$$

This means that we can approximate $d_{\mathcal{O}}$ everywhere in our free space using (5.7) if we can find a solution to the linear differential equation (5.6).

To solve (5.6) we rely here on a numerical method based on Gaussian Process regression.[1] In

[1] A precise treatment of this approach is behind the scope of this manuscript, but the interested reader may find more details and references to standard textbooks in [46] and [48]

particular, if $\mathcal{Q} \subset \partial\mathcal{O}$ is a finite subset of $\partial\mathcal{O}$ then the differential equation

$$\begin{cases} \Upsilon(p) - t\Delta\Upsilon(p) = \left(\frac{1}{t} - \Upsilon(p)\right) = 0, \forall p \in Int\ \mathcal{O}^C \\ \Upsilon(p) = 1, \forall p \in \mathcal{Q} \end{cases}$$

has a closed form solution given by the posterior mean μ of the Gaussian Process $\mathcal{GP}(0, k)$ with prior mean $m = 0$ and covariance function $k = k_{1,h}$ for $\mathcal{P} = \mathbb{R}^2$ or $k = k_{1,h}$ for $\mathcal{P} = \mathbb{R}^3$, conditioned on the dataset $\mathcal{D} = \{(p, 1)\}_{p\in\mathcal{Q}}$.The parameter $h \in \mathbb{R}_{++}$ is nothing but $\frac{1}{t}$ and can be adjusted to obtain closer approximation of the distance function.

Summing up the discusion so far, the *logarithmic Gaussian Implicit Surface* over the set $\mathcal{Q}$ is the function

$$\lambda(p) = -\frac{\ln \mu(p)}{h} \tag{5.8}$$

and it is the function we will use as a candidate Control Barrier Function instead of $d_{\mathcal{O}}$.

Before going on, we highlight that in the original paper, i.e. [41], the authors showed emprirically that the Matérn Kernel $k_{\frac{3}{2},l}$ with $l = \frac{\sqrt{6}}{h}$ can be used instead of the Matérn kernel $k_{1,h}$ for the two dimensional case. One of the advantage of using $k_{\frac{3}{2},l}$ as the prior covariance function is that its expression does not involve the evaluation of a Bessel function, and so computing its gradient and Hessian for the implementation of the safety filter is easier. So, in the following simulations, we will adopt $k_{\frac{3}{2},l}$ as the prior covariance function for the Gaussian Process regression.

Remark 5.2.1. *According to [48], the Matérn kernels $k_{1,h}$ and $k_{\frac{1}{2},h}$ are not continouosly differentiable so, according to definition 4.1.5, 5.8 is not a valid candidate Control Barrier Function. From a practical point of view, the only points of non-differentiabilities of the kernels are the sample points of the training set, which in our case are the boundary points of the obstacle. However, $p \in \partial\mathcal{O}$ imply a collision of the agent with the obstacle, so as long as the agent stay away from the obstacle we can use 5.8 as a Control Barrier Function.*

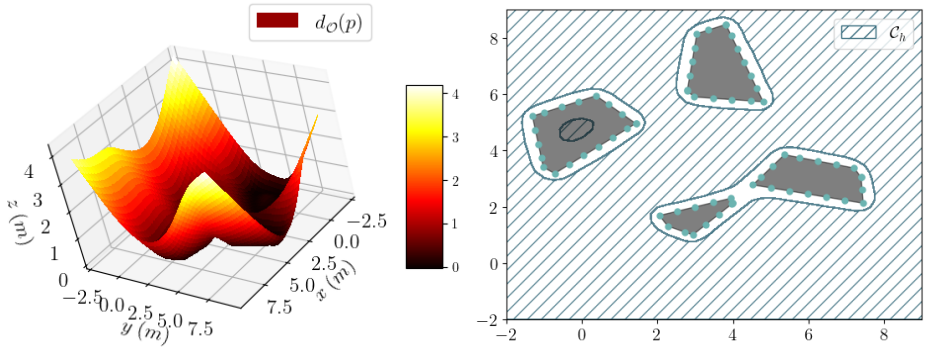
5.2.1 Numerical Simulations

Before showing the numerical simulation involving the safe navigation task, we show the qualitatively behavior of (5.8), shown in Figure 5.9, that can be compared with the real distance distance function in Figure 5.1 and the direct Gaussian Process regression in Figures 5.2 and 5.3. We can see how (5.8) is able to estimate the distance function quite accurately with far fewer samples than the direct Gaussian Process regression.

Figure 5.10 shows an agent navigating in an unknown enviroment, colleting samples from the obstacles when they are less than one meter far from the agent. As we can see from the picture, the control law resulting using (5.8) as a Control Barrier Function keeps the agent from colliding with the obstacles and steer the agent to the goal position.

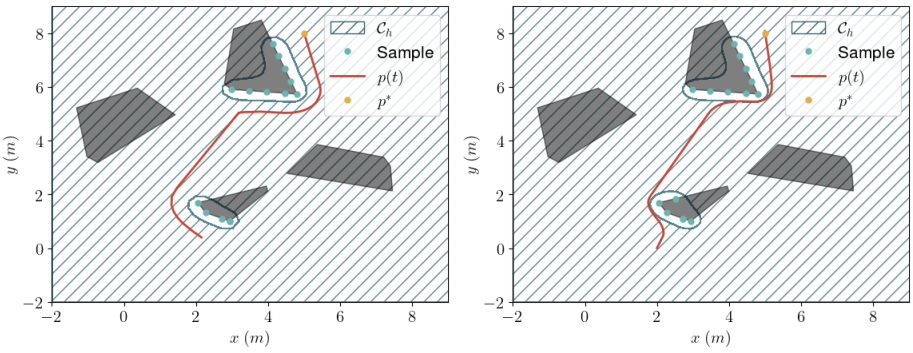
5.3 High Gain Observers for Safety Filters Learning

The content of this section is strongly inspired by the scenario presented in section 5.1. In that section we used Gaussian Process regression to estimate an unknown, but measurable, candidate Control Barrier Function, i.e. the distance from the obstacles, in this section we instead follow a different approach and we investigate the possibility of estimating the terms appearing in equation (4.11).



(a) (b)

Figure 5.9: The figures shows the reconstruction of the distance function trough the use of the log-Gaussian Process Implicit Surface regression, with $h = 3.5$ over a set of obstacle samples that are 0.5 m apart from each other. Figure (a) shows the tridimensional plot of (5.8), while figure (b) shows the corresponding safe set.



(a) (b)

Figure 5.10: The figures shows an agent navigating trough an unkwnown envinroment. In figure (a) the agent is modeled as a single integrator, while Figure(b) it is modeled as a double integrator.

Before introducing the technique we need to state some results about the relation between the Gaussian Process posterior mean and covariance functions and the estimation error, and to recall the definition and the properties of *High Gain Observers*.

Definition 5.3.1 (High Gain Observer)**.** *Let $z \in Z \subset \mathbb{R}^n$ be the state of an autonomous linear system written in canonical observability form*

$$\begin{cases} \dot{z} = Az + d(t) \\ y = Cz + v(t), \end{cases} \tag{5.9}$$

where $y \in \mathbb{R}$ is the measured output of the system, $d \in \mathbb{R}^n$ is a bounded disturbance, $v \in \mathbb{R}$ is the measurement noise and A, C have the form

$$A = \begin{bmatrix} 0_{(n-1)\times 1} & I_{n-1} \\ 0 & 0_{1\times(n-1)} \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0_{1\times(n-1)} \end{bmatrix}.$$

An High Gain Observer *for system (5.9) is given by the dynamical system*

$$\dot{\hat{z}} = A\hat{z} + D_l K(C\hat{z} - y), \tag{5.10}$$

where $\hat{z} \in \mathbb{R}^n$ is the state of the observer, $K = \begin{bmatrix} k_1 & k_2 & \cdots & k_n \end{bmatrix}^T$ is a vector of positive parameters ($k_i > 0$) chosen so that the matrix $A + KC$ is Hurwitz and $D_l = \mathrm{diag}(l, l^2, \ldots, l^n)$ is a diagonal matrix parameterized by $l > 0$.

Lemma 5.3.1 ([52])**.** *Given the system (5.9) with observer (5.10), if d and v are bounded, then there exists $l^* \in \mathbb{R}$ such that for every $l > l^*$ there exists $c_1, c_2, c_3, c_4 > 0$ such that for every $t > 0$ we have*

$$\begin{aligned} |\hat{z}_i(t) - z_i(t)| \leq c_1 l^{i-1} e^{-c_2 l t} |\hat{z}_i(0) - z_i(0)| \\ + \frac{c_3}{l^{n+1-i}} \|d\|_\infty + c_4 l^{i-1} \|v\|_\infty \end{aligned}$$

Lemma 5.3.2 ([47, 53])**.** *Let $f : \mathcal{X} \to \mathbb{R}$ be a Lipschitz continuous function with Lipschizt constant $L_f > 0$. Consider a zero mean Gaussian process with a Lipschitz continuous kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, with Lipschitz constant $L_k > 0$. Then the posterior mean μ and posterior variance κ conditioned on the training data $\mathcal{D} = \{(x^1, y^1), \ldots, (x^N, y^N)\}$, $N \in \mathbb{N}_0$ are Lipschizt continuous with Lipschitz constants L_μ and L_κ on $\mathcal{X}$, respectively, satisfying*

$$L_\mu \leq L_k \sqrt{N} \left\| \left(K_X + \sigma^2 I_N\right)^{-1} Y \right\|,$$
$$L_\kappa \leq 2\rho L_k \left(1 + N \left\| \left(K_X + \sigma^2 I_N\right)^{-1} \right\| \max_{x,x' \in X} k(x, x')\right),$$

Moreover, pick $\delta \in (0, 1)$, $\rho > 0$ and set

$$\beta(\rho) = 2 \log\left(\frac{M(\rho, \mathcal{X})}{\delta}\right),$$
$$\alpha(\rho) = (L + L_\mu)\rho + \sqrt{\beta(\rho) L_\kappa \rho},$$

with $M(\rho, \mathcal{X})$ the ρ-covering number [2] *related to the set $\mathcal{X}$. Then, the bound*

$$|f(x) - \mu(x)| \leq \sqrt{\beta(\rho)}\kappa(x) + \alpha(\rho), \quad \forall x \in \mathcal{X}$$

holds with probability at least $1 - \delta$.

In the following we will denote with $\mathcal{E}_{f,\mathcal{S}}(x) : \mathcal{X} \to \mathbb{R}$ the regressor fitted with the Gaussian process to the data set $\mathcal{D}$. In particular, our analysis is limited to case $\mathcal{E}_{f,\mathcal{S}}(x) = \mu(x)$.

We are now ready to present the proposed approach. Consider a nonlinear system of the form

$$\dot{x} = f(x) + g(x)u, \quad y = h(x) + \varepsilon(t), \tag{5.11}$$

where $x \in \mathcal{X} \subset \mathbb{R}^{n_x}$, $u \in \mathcal{U} \subset \mathbb{R}^n_u$, $f : \mathcal{X} \to \mathcal{X}$ and $g : \mathcal{X} \to \mathbb{R}^n_x \times n_u$, $h : \mathcal{X} \to \mathbb{R}$, and $\varepsilon(t) : \mathbb{R}_+ \to \mathbb{R}$ is a measurement noise. Let us assume that f, g, h are smooth functions of the state. Let the initial state $x(0) = x_0$ and the control input $u(t)$ be fixed and let $\Phi : \mathbb{R} \to \mathcal{X}$ be the resulting solution of (5.11). We introduce the set

$$\mathcal{T}_{\delta,T_1,T_2} = \left\{ x \in \mathbb{R}^n_x \ : \ x \in \bigcup_{t\in[T_1,T_2]} \delta\,\mathcal{B}(\Phi(t)) \right\} \tag{5.12}$$

where $T_2 > T_1 > 0$ and $\delta > 0$.

We suppose that both $x(t)$ and $y(t)$ are measurable for each $t > 0$. In this framework, we are interested in obtaining estimates of h and its first $r - 1$ functional derivatives (i.e. h, $L^1_f h$, $\ldots$, $L^{r-1}_f h$), along the trajectory of (5.11), where $r \in \mathbb{N}$, $r < n$, is given. The estimates are functions $\hat{h}^{(k)} : \mathcal{X} \to \mathbb{R}$, $k = 0, \ldots, r-1$, to be computed so that $|\hat{h}^{(k)}(x) - L^k_f h(x)|$ is small in some sense. We suppose the first $r-1$ time derivatives are not affected by output, namely, we assume the following.

Assumption 5.3.1. *$L_g L^k_f h(x) = 0$ for each $k < r$.*

Assumption 5.3.2. *h is a realization of a Gaussian process*

$$h \sim \mathcal{GP}(0, \kappa_0(\cdot, \cdot)).$$

Following [46], the previous assumption implies that also the higher derivatives of h are realizations of Gaussian processes with certain covariance κ_k, namely

$$L^k_f h \sim \mathcal{GP}(0, \kappa_k(\cdot, \cdot)), \ \forall k = 0, 1, \ldots r-1.$$

Let $\mathcal{S}^k_j = \left\{ t^k_{j-(N-1)}, t^k_{j-(N-2)}, \ldots t^k_j \right\}$ be a sliding time window of $N \in \mathbb{N}$ time instants $t^k_i \in \mathbb{R}_+$, where $t_j > t_{j-1}$. The strategy presented later tunes the Gaussian process linked to $\hat{h}^{(k)}$ with a data set obtained by sampling the available measures $x(t)$, $y(t)$, and the state of the high gain observer introduced later, at the time instances in $\mathcal{S}^k_j$, $j > 0$. The window is updated when the value of the state $x(t)$ fulfills $\|x(t) - x(t^k_j)\| > \tau > 0$, with $\tau > 0$, taking $t^k_{j+1} = t$.

Ideally, setting $\hat{h}^{(k)} = \mathcal{E}_{L^k_f,\mathcal{S}^k_j}$ would guarantee a probabilistic bound on the estimation error[47][53]. However, the values of $L^k_f h$ are not measurable; thus, we cannot construct the needed dataset for training. A first option could be to set $\hat{h} = \mathcal{E}_{h,\mathcal{S}^0_j}$ and to take $\hat{h}^{(k)} = L^k_f \hat{h}$ as estimates for $L^k_f h$.

[2] The minimum number such that there exists a finite set $\mathcal{X}_\rho$ so that its cardinality is equal to $M(\rho, \mathcal{X})$ and $\max_{x\in\mathcal{X}} \min_{x'\in\mathcal{X}_\rho} \|x - x'\| \leq \rho$.

This approach, however, has no theoretical guarantees and leads to an accumulation of errors in the process [54]. Moreover, any uncertainty of the system dynamics f would also compromise the quality of the estimate. In this work, we rather propose a technique to model $\hat{h}^{(k)}$ fulfilling

$$\left|\hat{h}^{(k)}(x) - \mathcal{E}_{L_f^k,\mathcal{S}_j^k}(x)\right| < \epsilon \quad \forall x \in \mathcal{X}$$

where $\epsilon > 0$ is a bound which depends on the noise of the available data and not relying on the knowledge of the vector field f. Then, we use this property to compute a probabilistic bound of convergence of this estimate to $L_f^k h$.

The core of the proposed approach is to use a high gain observer to generate an approximation of $L_f^k h$. Following the structure of (5.10)

$$\begin{aligned}
\dot{\hat{z}}_1 &= \hat{z}_2 + lk_1(\hat{z}_1 - y(t)) \\
\dot{\hat{z}}_2 &= \hat{z}_3 + l^2 k_2(\hat{z}_1 - y(t)) \\
&\vdots \\
\dot{\hat{z}}_r &= l^r k_r(\hat{z}_1 - y(t)),
\end{aligned} \tag{5.13}$$

where k_i, $i = 1, \ldots, r$, are chosen as in 5.3.1.

By using the property that the state $\hat{z}_{k+1}$ of (5.13) practically converges to $L_f^k h$, we compute the estimate $\hat{h}^{(k)}$ as $\mathcal{E}_{\hat{z}_{k+1},\mathcal{S}_j^k}$. The following theorem can then be proved.

Theorem 5.3.3. *Let $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a bounded and Lipschitz continuous function. Then there exist $\bar{t} > 0$ and $l^* > 0$ such that for each $t > \bar{t}$ and $l > l^*$, there exist constants $c_1, c_2, c_3 > 0$ such that, for all $x \in \mathcal{X}$,*

$$|\hat{h}^k(x) - \mathcal{E}_{L_f^k h,\mathcal{S}_j^k}(x)| \leq c_1 N \max\{c_2 l^k \|\varepsilon(t)\|_\infty, c_3 l^{k-r}\}.$$

Proof. Let $Y = \left[z_{t_{j-(N-1)}}, \cdots, z_{t_j}\right]^T$ and $\hat{Y} = \left[\hat{z}_{t_{j-(N-1)}}, \cdots, \hat{z}_{t_j}\right]^T$ then

$$\begin{aligned}
&|\mathcal{E}_{\hat{z}_{k+1},\mathcal{S}_j^k}(x) - \mathcal{E}_{L_f^k h,\mathcal{S}_j^k}(x)| = \\
&= |\kappa(x)^T (K_X + \sigma_n^2 I_N)^{-1}(\hat{Y} - Y)| \leq \\
&\leq \|\kappa(x)\| \|(K_X + \sigma_n^2 I_N)^{-1}\| \|\hat{Y} - Y\|.
\end{aligned}$$

Since κ is bounded, we have $\|\kappa(x)\| \leq \kappa_{max} > 0$. It follows that there exists a $K > 0$ so that $\|(K_X + \sigma_n^2 I_N)^{-1}\| \leq K$. Let us set $c_1 = \kappa_{max} K$, then

$$|\mathcal{E}_{\hat{z}_{k+1},\mathcal{S}_j^k}(x) - \mathcal{E}_{L_f^k h,\mathcal{S}_j^k}(x)| \leq c_1 \|\hat{Y} - Y\| \leq c_1 \sum_{i=j-N+1}^{j} \|\hat{z}_{k+1}(t_i) - z_{k+1}(t_i)\|$$

By means of 5.3.1, there exist a time instant $\bar{t} > 0$ so that, for all $t_j > \bar{t}$ the following holds

$$|\mathcal{E}_{\hat{z}_{k+1},\mathcal{S}_j^k}(x) - \mathcal{E}_{L_f^k h,\mathcal{S}_j^k}(x)| \leq c_1 N \max\{c_2 l^k \|\varepsilon(t)\|_\infty, c_3 l^{k-r}\}, \forall x \in \mathcal{X}$$

from which the result follows. □

Theorem 5.3.3 yields a bound on the difference between the ideal and the proposed estimate. 5.3.2, then, can be used to establish a probabilistic guarantee of convergence of $\hat{h}^{(k)}$ to $L_f^k h$ on the set $\mathcal{T}_{\delta,t_\epsilon,t}$, where t_ϵ is an arbitrarily small time and t is the current time, as formalized in the following theorem.

Theorem 5.3.4. *Pick $\eta \in (0,1)$ and $\rho > 0$. Let W_k^j, $W_{\mu_k^j}$ and $W_{\sigma_k^{2j}}$ be the Lipschitz constants of, respectively, $L_f^k h$, of the mean μ_k^j and variance $\sigma_k^{2,j}$ of the Gaussian process linked to $L_f^k h$ on the set $\mathcal{T}_{\delta,t_{j-(N+1)},t_j}$. Furthermore, let*

$$\beta(\rho) = 2\log\left(\frac{M(\rho, \mathcal{T}_{0,t_{j-(N+1)},t_j})}{\eta}\right)$$
$$\alpha(\rho) = (W_k^j + W_{\mu_k^j})\rho + \sqrt{\beta(\rho) W_{\sigma_k^{2j}} \rho}$$

For all $t_\epsilon > 0$ there exist $l^* > 0$ such that for all $l \geq l^*$ and for each j such that $t_{j-(N+1)} > t_\epsilon$ and $t_j \leq t$ the following hold with probability $1-\eta$

$$|\mathcal{E}_{\hat{z}_{k+1},\mathcal{S}_j^k}(x) - L_f^k h(x)| \leq \sqrt{\beta(\rho)}\sigma_k^{2,j}(x) + \alpha(\rho) + c_1 N \max\{c_2 l^k \|\varepsilon(t)\|_\infty, c_3 l^{k-r}\|\}$$

for all $x \in \mathcal{T}_{\delta,t_{j-(N+1)},t_j}$ and c_i, $i = 1,2,3$ are positive.

Proof.

$$\begin{aligned}
&\|\mathcal{E}_{\hat{z}_{k+1},\mathcal{S}_j^k}(x) - L_f h^k(x)\| \leq \\
&\leq \|\mathcal{E}_{\hat{z}_{k+1},\mathcal{S}_j^k}(x) - \mathcal{E}_{L_f h^k,\mathcal{S}_j^k}(x) + \mathcal{E}_{L_f h^k,\mathcal{S}_j^k}(x) - L_f h^k(x)\| \leq \\
&\leq \|\mathcal{E}_{\hat{z}_{k+1},\mathcal{S}_j^k}(x) - \mathcal{E}_{L_f h^k,\mathcal{S}_j^k}(x)\| + \|\mathcal{E}_{L_f h^k,\mathcal{S}_j^k}(x) - L_f h^k(x)\|,
\end{aligned}$$

then, applying Theorem 5.3.3 to the first term of the right side of the above equation, and 5.3.2 to the second term, the initial statement can be recovered. □

Chapter 6

Navigation With Vision Constraints

This chapter differs somewhat from the rest of the thesis, while maintaining its relevance to the main topic. While in the previous chapters we assumed that the agent could always precisely know its position in its environment, here our focus is on vision based navigation, which implies that the agent under control must be able to navigate or to localize itself trough the use of cameras, such as in the example showed in Figure 6.1 in which a quadrotor needs to race trough the gates of a track.

Our discussion will focus on the two major drawbacks of visual sensors, namely the unreliability of distance estimation computed from them and their limited field of view. To deal with the first, we will aim at designing control laws that are robust to distance estimation errors or that rely only on bearing measurements, which can more reliably measured with the use of cameras; while to deal with the second, we will study how to use Control Barrier Functions to impose field of view constraints.

We start in Section 6.1 by presenting a Control Barrier Function based control law that imposes the field of view constraints while being robust to distance estimation errors [55], and in Section 6.3 we show some preliminary results on the use of Control Lyapunov Function to design bearing based trajectory tracking control laws.

6.1 Control Barrier Functions for Visual Constraints

In the following, we will consider an autonomous agent moving in space modeled as a rigid body whose pose with respect to an inertial reference frame is denoted as $x = (p, R) \in \mathbb{R}^3 \times SO(3)$, where $p \in \mathbb{R}^3$ denotes the position of the rigid body and $R \in SO(3)$ its rotation transforming coordinates from the body to the inertial frame.

The agent is equipped with a visual sensor whose field of view is modeled as an infinite half-cone centered in p and with total aperture $2\psi_F$, with $\psi_F \in \left(0, \frac{\pi}{2}\right)$ (see Figure 6.1 for reference). We denote with $e_c \in \mathbb{R}^3$ the orientation of the optical axis of the visual sensor in the body frame. Formally, we denote the field of view of the agent $\mathcal{F}(x) \subset \mathbb{R}^3$ as:

$$\mathcal{F}(x) = \left\{ q \in \mathbb{R}^2 \mid \angle(\beta(p), Re_c) < \psi_F \right\}.$$

We suppose that a reference controller $u^*(x) : \mathcal{X} \to \mathbb{R}^m$ is given, able for example to track trajectories with the use of visual features. We let $\{q_i\}_{i=0}^N \subset \mathbb{R}^3$ denote a set of N scene features the agent needs to keep in sight during its operation, that may be useful either for localization or

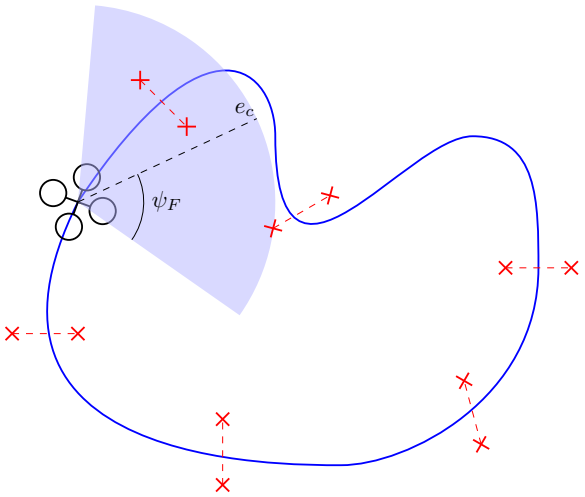


Figure 6.1: A quadrotor traversing a race circuit needs to keep the gates (depicted in red) as long as possible in its field of view (the blue cone) to orient itself and race through the circuit.

navigation. We use $d_i(p) = \|q_i - p\|_2$ to denote the distance of point i from the agent position p, and $\beta_i(p) = \frac{q_i - p}{\|q_i - p\|_2}$ for its associated bearing. Let $\hat{d}_i(p,t) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be an estimate of the quantity $d_i(p)$ available to the agent; we then define the relative error as $\tilde{d}(p,t) = \frac{d(p)}{\hat{d}(p,t)}$.

Problem 6.1.1. *The goal of the agent is to compute a control input u that is as close as possible to u^* while keeping each point of interest inside its field of view, i.e., at every time instant t:*

$$q_i \in \mathcal{F}(x), \quad \forall i \in \{1, \dots, N\}. \tag{6.1}$$

Remark 6.1.1. *In the following discussion, we will prove that the proposed approach is robust to bounded distance readings error, that is, if the multiplicative $\tilde{d}(p,t)$ is contained in an interval (which may be estimated from the sensor datasheet or by computing some bounds from the algorithm used to calculate the distance).*

Condition (6.1) for each feature point q_i can be equivalently characterized by a function $h_i : \mathbb{R}^3 \times SO(3) \to \mathbb{R}$ defined as

$$h_i(x) = \langle \beta_i(p), Re_c \rangle - \cos \psi_F; \tag{6.2}$$

It can be verified that:

- $h_i(x) > 0$, when $q_i \in int\mathcal{F}$
- $h_i(x) = 0$ when $q_i \in \partial\mathcal{F}$
- $h_i(x) < 0$, when $q_i \notin int\mathcal{F}$
- $h_i(x)$ is smooth anywhere except when $p = q_i$, where it is not defined.

and, as such, is a candidate Control Barrier Function. We can then define the safe sets $\mathcal{C}_{h_i} \subset \mathbb{R}^3 \times SO(3)$ according to Definition 4.1.4.

For the sake of exposition, in the following analysis, we will consider $N = 1$ and drop the subscript accordingly. At the end of this section, we will explain how to handle the general case $N > 1$. Furthermore, we will drop the dependence on the agent and feature positions when referring to the bearing β and distance function d associated with the feature.

6.1.1 Velocity Control

We first analyze the case in which we have control authority over the linear and angular velocities of the rigid body. By denoting the linear velocity of the body, expressed in the inertial frame, by $v \in T_p\mathbb{R}^3$ and its angular velocity, expressed in the body frame, by $\omega \in T_R SO(3)$, the equations of motion become

$$\begin{aligned} \dot{p} &= v \\ \dot{R} &= R[\omega]_\times \end{aligned}$$

Comparing with (4.1), we then have $x = x$ and $u = (v, \omega)$.

As is usual in the CBF literature, we aim to solve Problem 6.1.1 by solving the following optimization problem at each time instant:

$$\min_{u \in \mathbb{R}^3} \quad \|u - u^*\|_2^2 \tag{6.3a}$$

$$s.t. \quad \langle \nabla h, u \rangle + \gamma(h(x)) \geq 0. \tag{6.3b}$$

for a given class $\mathcal{K}$ function $\gamma : \mathbb{R} \to \mathbb{R}$. The constraint in (6.3b) expands to

$$\langle \nabla_p h, v \rangle + \langle \nabla_R h, \omega \rangle + \gamma(h(x)) \geq 0,$$

which can be further expanded (see [55]) as

$$-\frac{1}{d} e_c^T R^T P_\beta^T v - \beta^T R[e_c]_\times \omega + \gamma(h(x)) \geq 0.$$

Notice that the term that multiplies v depends explicitly on the multiplicative factor $\frac{1}{d}$, which, in our assumptions, is only approximately known through the estimate $\hat{d}$. To get around the issue we propose to substitute the previous constraint with the following triple of constraints:

$$\begin{aligned} -\frac{1}{\hat{d}} e_c^T R^T P_\beta^T v + c_1 \gamma(h(x)) &\geq 0, \\ -\beta^T R[e_c]_\times \omega + c_2 \gamma(h(x)) &\geq 0, \\ c_1 + c_2 = \gamma_0 &> 0. \end{aligned} \tag{6.4}$$

We will now present a theorem that shows that, under appropriate assumptions, there exist coefficients $c_1, c_2 \in \mathbb{R}$ such that enforcing (6.4) implies satisfaction of (6.3b); but first, we need to introduce a technical lemma.

Lemma 6.1.1. *Let consider $x \in \mathbb{R}$ and the first order differential equation*

$$\dot{x} = -\alpha(x, t) \tag{6.5}$$

where $\alpha : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is continuously differentiable and locally Lipschitz in both arguments and, for each $t \in \mathbb{R}$, $\alpha(t, x)$ is a class $\mathcal{K}$ function of x. Then $x = 0$ is a globally uniformly asymptotically stable equilibrium point for (6.5). Moreover, if $x(0) \geq 0$ then $x(t) \geq 0$ for all $t > 0$.

Proof. Let consider the Lyapunov function $V = \frac{1}{2}x^2$. The time derivative of V is

$$\dot{V} = -\alpha(t, x)x.$$

Since $\alpha(t, x)$ is a class $\mathcal{K}$ function for any t, we have that $\dot{V} = 0$ if $x = 0$, and $\dot{V} < 0$ otherwise. The first part of the claim directly follows from the Lyapunov theorem for nonlinear autonomous systems [38]. For the second part of the claim, assume by way of contradiction that there exist $t' > 0$ such that $x(t') < 0$; from the continuity of the solution to the differential equation, there then exist a t'' such that $x(t'') = 0$; however, this implies $\dot{x}(t'') = 0$ (for any value of α), and $x(t) = 0$ for all $t > t''$, leading to a contradiction at t'. □

Theorem 6.1.2. *Let $\gamma : \mathbb{R} \to \mathbb{R}$ be a class $\mathcal{K}$ function. Given a nominal control law $u^*(x) : \mathbb{R}^3 \times SO(3) \to \mathbb{R}^3$, and constants $\gamma_0, d_m, d_M > 0$, such that $d_m < 1 < d_M$, the control law u resulting from the solution of the following quadratic program*

$$\min_{\substack{u \in \mathbb{R}^3 \\ c_1, c_2 \in \mathbb{R}}} \quad \|u - u^*(x)\|_2^2 \tag{6.6a}$$

$$\text{subject to} \quad -\frac{1}{\tilde{d}} e_c^T R^T P_\beta^T v + c_1 \gamma(h(x)) \geq 0 \tag{6.6b}$$

$$-\beta^T R [e_c]_\times \omega + c_2 \gamma(h(x)) \geq 0 \tag{6.6c}$$

$$c_1 + c_2 = \gamma_0 \tag{6.6d}$$

$$\frac{\gamma_0}{1 - d_M} < c_2 < \frac{\gamma_0}{1 - d_m} \tag{6.6e}$$

renders the set $\mathcal{C}_h$ forward invariant whenever $\tilde{d} \in [d_m, d_M]$.

Remark 6.1.2. *At a high level, the idea of the proof is to show imposing the* (6.6b) *and* (6.6c) *is equivalent to imposing the CBF constraint with a time-varying class $\mathcal{K}$ function $\bar{\gamma}$ different from γ. The constraint* (6.6e) *is then necessary to ensure that the CBF constraint from $\bar{\gamma}$ is valid. Finally, constraint* (6.6d) *can be interpreted as fixing the scale for $\bar{\gamma}$.*

Proof. Assume that the optimization problem is feasible, if not one can always adjust the parameters to obtain a feasible optimization problem. Since $\tilde{d} > 0$, we multiply (6.6b) by $\frac{1}{\tilde{d}}$ to obtain

$$-\frac{1}{d} e_c^T R^T P_\beta^T v + \frac{c_1}{\tilde{d}} \gamma(h(x)) \geq 0;$$

summing with (6.6c) we obtain

$$\langle \nabla h, u \rangle + \gamma_0 \left(c_2 + \frac{c_1}{\tilde{d}} \right) \gamma(h(x)) \geq 0.$$

To ensure forward invariance, we need $c_2 + \frac{c_1}{\tilde{d}} > 0$. We can rewrite this condition as a function of c_2 alone by using constraint (6.6e):

$$c_2 + \frac{c_1}{\tilde{d}} = c_2 + \frac{\gamma_0 - c_2}{\tilde{d}} = \frac{c_2(1 - \tilde{d}) + \gamma_0}{\tilde{d}} > 0.$$

For the above to hold, we need to study how the function changes for various values of $\tilde{d}$. One can easily verify that the above condition can be rewritten as:

$$\begin{cases} c_2 \in \mathbb{R} & \text{if } \tilde{d} = 1, \\ c_2 < \frac{\gamma_0}{1 - \tilde{d}} & \text{if } 0 < \tilde{d} < 1, \\ c_2 > \frac{\gamma_0}{1 - \tilde{d}} & \text{if } \tilde{d} > 1, \end{cases}$$

which is equivalent to the constraint (6.6e).

We then have that if $\tilde{d} \in [d_m, d_M]$ then $\bar{\gamma}(h,t) = \left(c_2(t) + \frac{c_1(t)}{\tilde{d}(t)}\right)\gamma(h)$ is a valid CBF function; from Lemma 6.1.1 we then have that, if $h(0) > 0$, the solution of the differential equation $\dot{h} = -\bar{\gamma}(h,t)h$ is strictly positive. From the comparison lemma [38], we then conclude that the set $\mathcal{C}_h$ is forward invariant. □

6.1.2 Acceleration Control

Let us now analyze the case in which a second-order dynamical system governs the pose of the agent. Denoting the linear acceleration of the body in the inertial frame by $a \in T_p\mathbb{R}^3$, and its angular acceleration in the body frame by $\alpha \in T_R SO(3)$, the equations of motion become

$$\begin{aligned}\dot{p} &= v\\ \dot{v} &= a\\ \dot{R} &= R[\omega]_\times\\ \dot{\omega} &= \alpha\end{aligned}$$

Comparing with (4.1), we then have $x = (p, R, v, \omega)$ and $u = (a, \alpha)$. According to Definition 4.2.2, for given class $\mathcal{C}$ functions $\gamma_1, \gamma_2 : \mathbb{R} \to \mathbb{R}$, the constraint we need to satisfy in this case is

$$L_f^2 h(x) + L_g L_f h(x) u + \gamma_1'(h) L_f h(x) + \gamma_2(L_f h + \gamma_1(h(x))) \geq 0$$

which expands to

$$\begin{aligned}&\langle \nabla_p h, a\rangle + \langle \nabla_R h, \alpha\rangle +\\ &+\langle Hess_p h[v], v\rangle + \langle Hess_R h)[\omega], \omega\rangle +\\ &+\langle Hess_p h[v], \omega\rangle + \langle Hess_R h[\omega], v\rangle +\\ &+\gamma_1'(h)\left(\langle \nabla_p h, v\rangle + \langle \nabla_R h, \omega\rangle\right) +\\ &+\gamma_2(\langle \nabla_p h, v\rangle + \langle \nabla_R h, \omega\rangle + \gamma_1(h(x))) \geq 0\end{aligned} \tag{6.7}$$

Similarly to 6.1.2, we split (6.7) to obtain two constraints, one that collects all the terms that contain d in the denominator, and one that collects all the terms that do not contain d. However, the term $\langle Hess_p h[v], v\rangle$ contains d^2 in the denominator, as it expands (see the Appendix) to

$$v^T \frac{2(e_c^T R^T \beta)I - 3(e_c^T R^T \beta)P(\beta) - Re_c\beta^T - \beta e_c^T R^T}{d^2} v. \tag{6.8}$$

As a consequence, this term cannot belong to either of the two constraints. To deal with this problem, we decompose the velocity v along a component parallel to the bearing, $v_\parallel = \langle \beta, v\rangle \beta$, and one perpendicular to it, $v_\perp = P_\beta v$, so that $v = v_\parallel + v_\perp$. Moreover, since $\dot{\beta} = -\frac{1}{d} P_\beta v$, we have

$$v_\perp = -d\dot{\beta}. \tag{6.9}$$

Now, due to the linearity of the Hessian and the metric, we can rewrite (6.8) as follows

$$\begin{aligned}\langle Hess_p h[v], v\rangle = \big\langle Hess_p h[v_\perp + v_\parallel], v_\perp + v_\parallel\big\rangle =\\ = \langle Hess_p h[v_\perp], v_\perp\rangle + \big\langle Hess_p h[v_\perp], v_\parallel\big\rangle\\ + \big\langle Hess_p h[v_\parallel], v_\perp\big\rangle + \big\langle Hess_p h[v_\parallel], v_\parallel\big\rangle.\end{aligned} \tag{6.10}$$

Expanding the computations, we can see that $\langle Hess_p h[v_\parallel], v_\parallel \rangle = 0$, and we can use (6.9) to simplify a distance factor from the remaining terms containing d^2 at the denominator.

As a consequence, under the assumption that $\dot{\beta}$ is measurable (e.g., through numerical differentiation or the use of an observer), we can compute the various terms in (6.10) without requiring the square of the distance. This leads us to the following claim.

Theorem 6.1.3. *Let $\gamma_1, \gamma_2 : \mathbb{R} \to \mathbb{R}$ be class $\mathcal{K}$ functions. Without loss of generality, assume $\gamma_1'(0) \geq \gamma_2'(0)$ (otherwise swap the role of the two functions). Define $K = \frac{4\gamma_1'(0)\gamma_2'(0)}{(\gamma_1'(0)+\gamma_2'(0))^2}$. Given a nominal control law $u^*(x) : \mathbb{R}^3 \times SO(3) \to \mathbb{R}^3$ and constants $\gamma_0, d_m, d_M > 0$, such that $\gamma_0 > K$ and $d_m < 1 < d_M$, then there exists $M > 0$ such that the control law resulting from the solution of the following quadratic program*

$$\min_{\substack{u \in \mathbb{R}^3 \\ c_1, c_2 \in \mathbb{R}}} \quad \|u - u^*(x)\|_2^2 \tag{6.11a}$$

$$s.t. \quad \tilde{d}\left(\langle grad_p h, a\rangle + 2\langle Hess_p[v_\perp], v_\parallel\rangle\right) + \tag{6.11b}$$
$$c_1\left(\gamma_1' L_f h(x) + \gamma_2(L_f h(x) + \gamma_1(h(x)))\right) \geq 0$$

$$\langle grad_R h, \alpha\rangle + \langle Hess_R h[\omega], \omega\rangle + \tag{6.11c}$$
$$2\langle Hess_R h[\omega], v\rangle + \langle Hess_p h[v_\perp], v_\perp\rangle +$$
$$c_2\left(\gamma_1' L_f h(x) + \gamma_2(L_f h(x) + \gamma_1(h(x)))\right) \geq M$$

$$c_1 + c_2 = \gamma_0 \tag{6.11d}$$

$$\frac{K d_M - \gamma_0}{d_M - 1} \leq c_2 \leq \frac{K d_m - \gamma_0}{d_m - 1} \tag{6.11e}$$

renders the set $\mathcal{C}_h$ forward invariant whenever $\tilde{d} \in [d_m, d_M]$.

The terms in (6.11) are given by (see the Appendix for the full derivation):

$$\begin{aligned}
\langle Hess_p h[v_\perp], v_\perp\rangle &= -\left(\beta^T R e_c\right)\left(\dot{\beta}^T \dot{\beta}\right) \\
\left\langle Hess_p h[v_\perp], v_\parallel\right\rangle &= \left\langle Hess_p h[v_\parallel], v_\perp\right\rangle = \frac{(v^T\beta)\dot{\beta}^T P(\beta) R e_c}{d} \\
\nabla_p h(x) &= -\frac{P_\beta R e_c}{d} \\
\nabla_R h(x) &= -\beta^T R [e_c]_\times \\
\langle Hess_R h[\omega], v\rangle &= \langle Hess_p h[v], \omega\rangle = v^T \frac{P_\beta R [e_c]_\times}{d}\omega \\
\langle Hess_R h[\omega], \omega\rangle &= \omega^T [R^T\beta]_\times [e_c]_\times \omega.
\end{aligned}$$

Remark 6.1.3. *At a high level, the proof follows the same reasoning of Theorem 6.1.2. However, it also need to take into account the time-varying nature of γ_1 when defining the HOCBF constraint ψ_2; this, in turn, depends on $\dot{\tilde{d}}$, which need to be bounded by M.*

Before considering the proof of 6.1.3, we prove the following intermediate results:

Lemma 6.1.4. *Let $L = \gamma_0\left(c_2 + \frac{c_1}{\tilde{d}}\right)$. There exists two functions α_1, α_2 such that*

$$\alpha_1'(h)\dot{h} + \alpha_2(\dot{h} + \alpha_1(h)) = L\left(\gamma_1'(h)\dot{h} + \gamma_2(\dot{h} + \gamma_1(h))\right) \tag{6.12}$$

Proof. Since (6.12) must hold for each h and $\dot{h}$, it must be true also for the special cases where $h = 0$ or $\dot{h}$. Remembering that, from the properties of class-$\mathcal{K}$ functions, $\gamma_1(0) = \alpha_1(0) = 0$, equation (6.12) implies

$$h = 0 \Rightarrow L\left(\gamma_1'(0)\dot{h} + \gamma_2(\dot{h})\right) = \alpha_1'(0)\dot{h} + \alpha_2(\dot{h}), \tag{6.13}$$

$$\dot{h} = 0 \Rightarrow L\gamma_2(\gamma_1(h)) = \alpha_2(\alpha_1(h)). \tag{6.14}$$

From the above, the following must hold for any $z \in \mathbb{R}$:

$$\alpha_1(z) = \alpha_2^{-1}(\gamma_2(\gamma_1(z))) \tag{6.15a}$$

$$\alpha_2(z) = L\left(\frac{d\gamma_1}{dh}(0)z + \gamma_2(z)\right) - \alpha_1'(0)z \tag{6.15b}$$

This provides an expression of α_1 as a function of α_2, and of α_2 as a function of the scalar $\alpha_1'(0)$.

To compute $\alpha_1'(0)$ (together with $\alpha_2'(0)$), let us take the derivative of both of the equations in (6.15) by their argument:

$$\alpha_1'(z) = \frac{\gamma_2'(\gamma_1(z))\gamma_1'(z)}{\alpha_2'(\alpha_2^{-1}(\gamma_2(\gamma_1(z))))} \tag{6.16a}$$

$$\alpha_2'(z) = L(\gamma_1'(0) + \gamma_2'(z)) - \alpha_1'(0) \tag{6.16b}$$

Let us define $G = \gamma_1'(0) + \gamma_2'(0)$. We then obtain:

$$\alpha_{1,2}'(0) = \frac{LG \mp \sqrt{L^2G^2 - 4L\gamma_1'(0)\gamma_2'(0)}}{2} \tag{6.17}$$

□

Lemma 6.1.5. *If $L \geq K$ and $\gamma_1'(0) \geq \gamma_2'(0)$, then the two functions α_1, α_2 from 6.1.4 are class $\mathcal{K}$.*

Proof. Using the condition $L \geq K$ in (6.17) implies the realness of $\alpha_1'(0), \alpha_2'(0)$.

We also need to ask both $\alpha_1'(z)$ and $\alpha_2'(z)$ to be positive for each $z \in \mathbb{R}$. First of all, by looking at equations (6.16) we can notice that $\alpha_1'(z)$ has the same sign as $\alpha_2'(z)$, so we just need to ask for $\alpha_2'(z) > 0$ for each z. This is true if $\gamma_2'(z) > \frac{\alpha_1'(0)}{L} - \gamma_1(0)$, so a sufficient condition to ask is $\frac{\alpha_1'(0)}{L} - \gamma_1(0) < 0$. Now, taking the convention that $\alpha_1'(0)$ is obtained using the solution with the minus sign and expanding equation (6.17), we can see that this last condition is true when

$$\gamma_1'(0) \geq \gamma_2'(0) - \sqrt{G^2 - \frac{4}{L}\gamma_1'(0)\gamma_2'(0)} \tag{6.18}$$

which is trivially true if $\gamma_1'(0) \geq \gamma_2'(0)$. □

Proof of Theorem 6.1.3. As in the proof of Theorem 6.1.2, the sum of the first two constraints recovers a formula similar to the second-order barrier function constraint, namely:

$$\ddot{h} + \gamma_0\left(c_2 + \frac{c_1}{\tilde{d}}\right)\left(\gamma_1'\dot{h} + \gamma_2(\dot{h} + \gamma_1(h(x))\right) - M \geq 0.$$

To make use of the comparison lemma, we need to prove that there exists two, time-varying, class $\mathcal{K}$ functions α_1, α_2 such that:

$$\frac{\partial\alpha_1}{\partial t} + \alpha_1'(h)\dot{h} + \alpha_2(\dot{h} + \alpha_1(h)) \geq L\left(\gamma_1'(h)\dot{h} + \gamma_2(\dot{h} + \gamma_1(h))\right) - M \tag{6.19}$$

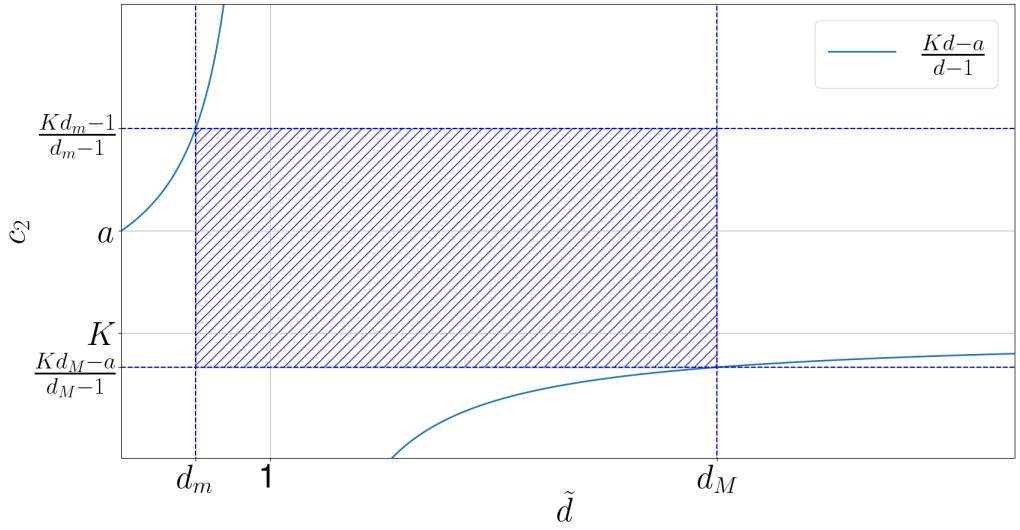


Figure 6.2: The figure shows the relation between the allowable error ration $\tilde{d}$ and the resulting range for c_2 in Theorem 6.1.3. As we can see, a trade-off exists between the bounds on $\tilde{d}$ and the resulting range for c_2.

where the explicit time dependence has been omitted for conciseness, and M is a design parameter that will be used to deal with the unknown $\frac{\partial \alpha_1}{\partial t}$; in fact, considering the results of Lemmata 6.1.4 and 6.1.5, equation (6.19) holds if $L \geq K$ and if $\frac{\partial \alpha_1}{\partial t} \geq M$.

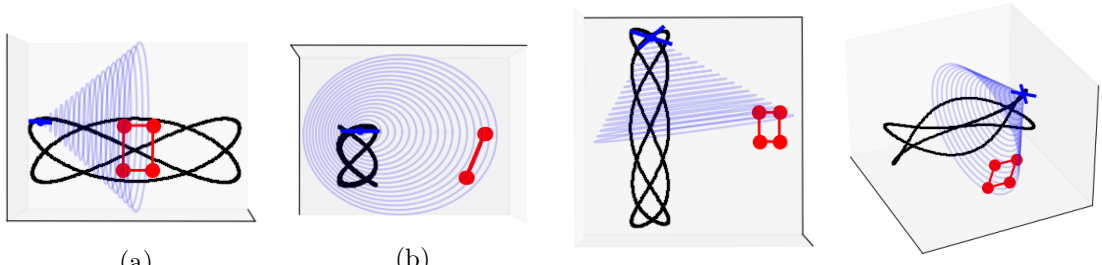
Using (6.11d) to substitute c_1 in $L \geq K$, we get: $c_2 + \frac{\gamma_0 - c_2}{\tilde{d}} \geq K$, which is equivalent to $c_2(1-\tilde{d}) \geq K\tilde{d} - \gamma_0$, since $\tilde{d} > 0$. As in the case analyzed in Theorem 6.1.2, based on the value of $\tilde{d}$ we get

$$\begin{cases} \gamma_0 \geq K, & \tilde{d} = 1 \\ c_2 \geq \frac{K\tilde{d}-\gamma_0}{\tilde{d}-1}, & \tilde{d} \geq 1 \\ c_2 \leq \frac{K\tilde{d}-\gamma_0}{\tilde{d}-1}, & \tilde{d} \leq 1 \end{cases} \tag{6.20}$$

Now, under the assumption that $\gamma_0 > K$, one can prove the function $\frac{Kd-\gamma_0}{d-1}$ is monotonically increasing over the intervals $]-\infty, 1[$ and $]1, +\infty[$, and that it has the shape showed in Figure 6.2, which means $\frac{Kd-\gamma_0}{d-1} > a$ over the interval $]-\infty, 1[$ and $\frac{Kd-\gamma_0}{d-1} < K$ over the interval $]1, +\infty[$. This means that, given a relative error range $[d_m, d_M]$, we can impose constraint 6.20 as long as $d_M > 1 > d_m > 0$.

We now need to prove that $\frac{\partial \alpha_1}{\partial t} \geq -M$ for each t. This is equivalent to $M \geq \max_t \left|\frac{\partial \alpha_1}{\partial t}\right|$, which can be imposed true whenever $\frac{\partial \tilde{d}}{\partial t}$ is bounded.

Now, as in Theorem 6.1.2, we can invoke Theorem 6.1.1 and the comparison lemma [38] to conclude the proof. □



(a) (b) (c) (d)

Figure 6.3: Snapshot of the simulation scenario of the numerical experiments. We can see that agent (represented by a blue cross) needs to turn to keep all the features (red points) in its field of view (blue cone) to follow the reference trajectory (black line). (a) Front view. (b) Side view (c) Top-down view. (d) Tilted view.

6.1.3 Multiple Features

When $N > 1$ features need to be tracked, we propose to extend the quadratic programming of Theorems 6.1.2 and 6.1.3 by adding for each feature a couple of optimization variables c_{i1}, c_{i2} and the same set of constraints proposed for the case of the single features.

Remark 6.1.4. *Having multiple constraints may lead to the infeasibility of the optimization problem. To overcome these issues, and improve the numerical stability of the optimization, one may add some slack variables $\delta_{i1}, \delta_{i2} > 0$ to each couple of constraints to allow for constraints violation. The square of the norm of these variables should be added to the cost function, weighted by some constant chosen by the designer.*

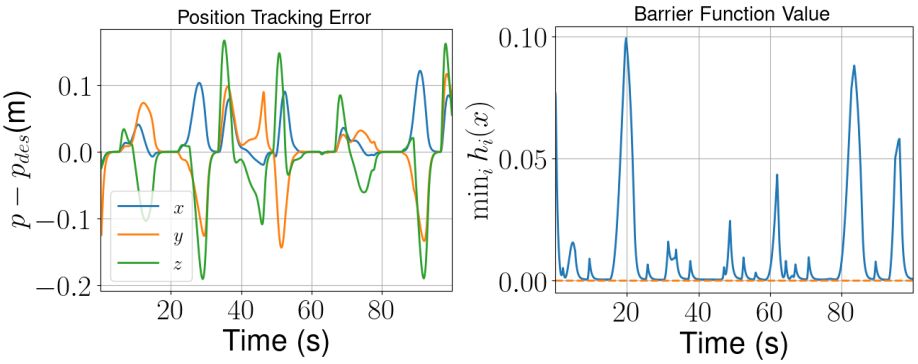
6.2 Numerical Simulation

To validate the proposed approach, in this section, we present two numerical simulations[1] involving the control of a double integrator system, as presented in Section 6.1.2, and the control of a quadrotor. In both simulations, the task of each agent is to follow the same trajectory $p_{ref}(t) = [\cos(0.3t), 10\cos(0.2t), 2\cos(0.2t)]^T$ (depicted in Figure 6.3). The features to keep in the field of view, $q_1 = [7.0, -1.5, 1.5]^T$, $q_2 = [7.0, 1.5, 1.5]^T$, $q_3 = [6.0, 1.5, -1.5]^T$, and $q_4 = [6.0, -1.5, -1.5]^T$ form the corners of a rectangle. We use a conic field of view with half-aperture $\psi_F = \frac{\pi}{6}$. The two simulations show an extreme case where the agents cannot estimate the distance from the features (i.e., $\hat{d} = 1$). The parameters mentioned in Theorem 6.1.3 are $\gamma_0 = 3$, $d_M = 15$, $d_m = 0.5$ and $M = 0$.

Figure 6.5(a) and 6.5(b) show, respectively, the tracking error of the proposed control scheme and the minimum of the Control Barrier Functions $h_i(x)$ values in the quadrotor scenario. The nominal control law of the agent is the one proposed in [13], with $k_p = 20.8$, $k_v = 13.3.$, $k_R = 54.81$, and $k_\omega = 10.54$.

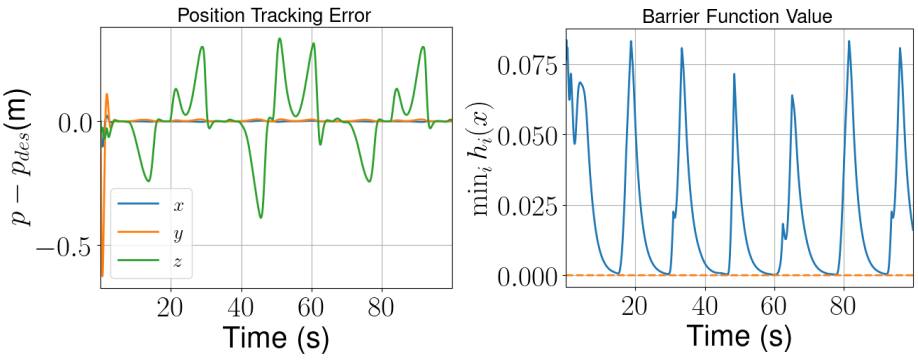
We can see, in both cases, that the constraint of imposed by the Control Barrier Function is always satisfied, i.e. the features never leave the field of view of the agent, and the reference trajectory is tracked with a bounded tracking error.

[1]The code used to implement the simulation can be found at https://github.com/biagio-trimarchi/A-Control-Barrier-Function-Candidate-for-Limited-Field-of-View-Sensors.git



(a) (b)

Figure 6.4: *Acceleration Control:* Plot (a) shows the position tracking error of a rigid body actuated both in linear and angular acceleration along the prescribed path. Plot (b) shows the minimum value among the barrier functions associated with the features.



(a) (b)

Figure 6.5: *Quadrotor:* Plot (a) shows the position tracking error of a quadrotor actuated both in trust and torques along the prescribed path. Plot (b) shows the minimum value among the barrier functions associated with the features.

6.3 Control Lyapunov Functions for Visual Servoing

Inspired by the approach presented above, in this section we propose a trajectory tracking CLF robust to distance measurements errors. To start, let us consider the same scenario of Section 6.1.1 and let us suppose to have designed a trajectory $p^* : \mathbb{R}_+ \to \mathbb{R}^3$ from which we can compute the *desired bearings* $\beta_i^*(t) = \frac{q_i - p^*(t)}{\|q_i - p^*(t)\|_2}$ to a set of $N \in \mathbb{N}_0$ visual features with known position $q_i \in \mathbb{R}^3$. Let us define the folllowing candidate Control Lyapunov Function to track $p^*(t)$:

$$V(x) = \sum_{i=1}^{N} 1 - \langle \beta_i, \beta_i^* \rangle \tag{6.21}$$

which, clearly, at each $t \in \mathbb{R}$ has a minimum whenever, for all $i \in N$, $\beta_i = \beta_i^*$.

Now, notice that equation 6.21 can be decomposed in n components $V_i(x) = 1 - \langle \beta_i, \beta_i^* \rangle$, one for each visual feature, with gradient

$$\nabla V_i = \frac{\langle P_{\beta_i} \beta_i^*, v \rangle}{d_i} + \frac{\left\langle P_{\beta_i^*} \beta_i, v^* \right\rangle}{d_i^*}$$

To converge to the desired trajectory, we should constraint the control input to satisfy, for each visual feature q_i,

$$\frac{\langle P_{\beta_i} \beta_i^*, v \rangle}{d_i} + \frac{\left\langle P_{\beta_i^*} \beta_i, v^* \right\rangle}{d_i^*} + \alpha\left(V(x)\right) \leq 0 \tag{6.22}$$

for a given class $\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$. Since our we only have access to the noisy measurements $\hat{d}$, the constraint we can actually impose is

$$\frac{\langle P_{\beta_i} \beta_i^*, v \rangle}{\hat{d}_i} + \frac{\left\langle P_{\beta_i^*} \beta_i, v^* \right\rangle}{d_i^*} + \alpha\left(V(x)\right) \leq 0 \tag{6.23}$$

which is equivalent to

$$\frac{\langle P_{\beta_i} \beta_i^*, v \rangle}{\tilde{d}_i} + \frac{\left\langle P_{\beta_i^*} \beta_i, v^* \right\rangle}{d_i^* d} + \frac{\alpha}{d}\left(V(x)\right) \leq 0 \tag{6.24}$$

Remark 6.3.1. *Equations 6.22 and 6.24 are not equivalent. By dividing 6.24 by d_i we get, first of all, that the class $\mathcal{K}$ function of the new constraint is $\frac{\alpha}{d_i}$, which is a time varying class $\mathcal{K}$ function; moreover, the term $\frac{\left\langle P_{\beta_i^*} \beta_i, v^* \right\rangle}{d_i}$ implies that each constraint is tracking a different desired velocity $\frac{v^*}{d_i}$.*

Armed with the above considerations, we propose the following CBF-CLF-QP control scheme to

track p^*

$$\min_{\substack{u\in\mathbb{R}^3\\ c_1,c_2\in\mathbb{R}}} \quad \|u\|_2^2 + \sum_{i=1}^{N} \left(w_{1,i}\delta_{1,i}^2 + w_{i,2}\delta_{2,i}^2\right) \tag{6.25a}$$

$$\textit{subject to} \quad d_i^* \langle P_{\beta_i}\beta_i^*, v\rangle + \left\langle P_{\beta_i^*}\beta_i, v^*\right\rangle + d_i^*\alpha V_i(x) \leq 0 \tag{6.25b}$$

$$-\frac{1}{\hat{d}} e_c^T R^T P_{\beta_i}^T v + c_{i,1}\gamma(h_i(x)) \geq \delta_{i,1} \tag{6.25c}$$

$$-\beta_i^T R[e_c]_\times \omega + c_{i,2}\gamma(h_i(x)) \geq \delta_{2,i} \tag{6.25d}$$

$$c_{i,1} + c_{i,2} = \gamma_0 \tag{6.25e}$$

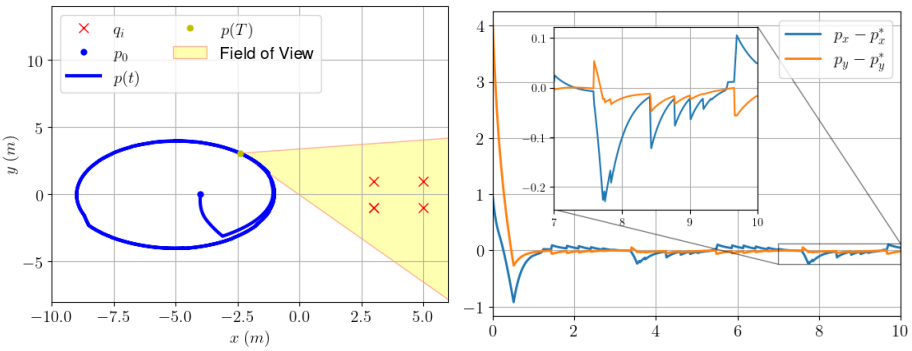
$$\frac{\gamma_0}{1-d_M} < c_{i,2} < \frac{\gamma_0}{1-d_m} \tag{6.25f}$$

$$\forall i \in 1,\dots,N$$

In the above equation, $\delta_{i,1}, \delta i, 2, \in \mathbb{R}$ are slack variables that are to ensure the existence of a solution, and $w_{i,1}, w_{i,2} \in \mathbb{R}_+$ are adjustable weights to discourage high values of the slack variables.

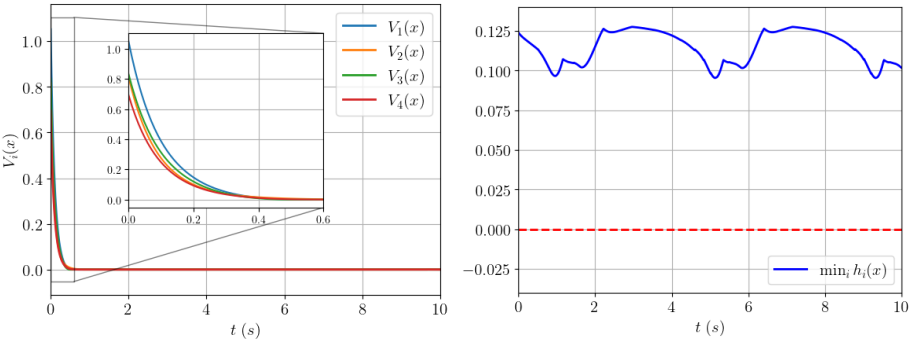
6.3.1 Numerical Simulation

We now show a numerical simulation to validate the approach proposed in this section. An agent starts from $p_0 = [-4.0, 0.0, 0.0]^T$ and it is tasked to follow the trajectory $p^*(t) = [-5.0, 0.0, 0.0]^T + 4.0[\sin(1.5t), -\cos(1.5t), 0.0]^T$. In the workspace are presents four visual features located at $q_1 = [3.0, -1.0, 0.0]$, $q_2 = [3.0, 1.0, 0.0]$, $q_3 = [5.0, -1.0, 0.0]$, $q_4 = [5.0, 1.0, 0.0]$ that the agent must keep in its field of view and need to track the reference trajectory p^*. Figures 6.6 and 6.7 show the the performance of the control law obtained by recursively solving 6.25 starting from $p = p_0$ and $R = I_3$, with $\alpha = 7.5$, $\gamma = 1.5$, $\gamma_0 = 1$, $d_m = 0.5$, $d_M = 20$, and, for each $i \in \{1, 2, 3, 4\}$, $w_{i,1} = w_{i,2} = 10.0$; morever we set $\hat{d} = d^*$ to simulate a bearing only scenario, in which we use the desired distance as a proxy for the real distance. We can see how the tracking error remains bounded while the features always remains in the field view.



(a) (b)

Figure 6.6: Figure (a) shows the scenario of the simulation. $p(T)$ denotes the position of the agent at $T = 10\ (s)$, and the yellow cone represents its field of view in that instant. Figure (b) shows the position tracking error: p_x denotes the x component of the position vector, and p_y denotes the y component.



(a) (b)

Figure 6.7: Figure (a) show the evolution in time of the various Lyapunov functions V_i. We can see that each of them converge exponentially fast to 0. Figure (b) shows the minimum value of the Control Barrier Functions; as predicted by the theory, it stays positive during the execution of the task.

Chapter 7

Conclusions

The primary objective of this thesis was to develop robust algorithms for safe navigation and collision avoidance to enable autonomous quadrotors to operate reliably without risk of harmful crashes. To address this, we explored multiple strategies and approaches. We began by introducing a path-planning framework based on Bézier curves, which generates optimal, feasible paths for quadrotors navigating through cluttered environments. This method provides a foundation for efficiently avoiding obstacles while minimizing path length and smoothness constraints.

We then delved into data-driven approaches for designing safety filters by leveraging Gaussian Processes. This exploration led to several innovative methods aimed at creating adaptive safety filters capable of handling dynamic environments and offering robust performance even when the environment is only partially known. The versatility of Gaussian Processes allowed us to design navigation strategies that adapt to uncertain surroundings, providing an essential step toward safer quadrotor autonomy.

In the final section, we presented preliminary work utilizing Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs) to formulate vision-based control laws using bearing measurements. This approach introduces the potential to bypass full reliance on precise localization data by using only visual bearings, which is crucial for quadrotors operating in unknown environments or when GPS signals are unavailable.

Despite these advancements, most of the approaches developed in this thesis rely on knowledge of the entire environment or the exact position of the quadrotor for collision avoidance. A promising direction for future research is to extend the work presented in the final chapter by designing Control Barrier Functions that can be computed directly from visual measurements. This would allow for real-time obstacle avoidance without requiring explicit localization, thus expanding the quadrotor's autonomy and applicability in a variety of real-world scenarios.

In summary, this thesis has contributed novel algorithms and methodologies for safer, more autonomous quadrotor navigation. By combining optimal path planning, adaptive safety filters, and vision-based control laws, we have laid a foundation for further advancements in the field of autonomous aerial robotics. These contributions hold promise for a future where quadrotors operate seamlessly and safely in complex, real-world environments.

Appendix A

Bézier Curves

In this chapter we review the definition of Bézier curve and Bernstein polynomial, and we recall the main properties of these mathematical objects used in our treatise. Let $n, i \in \mathbb{N}$ with $n > i$. The i^{th} *Bernstein Polynomial* of order n is the function $b_i^n(t) : [0,1] \to \mathbb{R}$ defined as

$$b_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

Given $n \in \mathbb{N}, d \in \mathbb{N}_0$ and $n+1$ points $P_i \in \mathbb{R}^d$, for $i \in \{0, 1, \ldots, n\}$, we call a *Bézier Curve* of order n in $\mathbb{R}^d$ the curve

$$pt = \sum_{i=0}^{n} b_i^n(t) P_i$$

defined for $t \in [0,1]$. The points P_i are called the *control points* of the curve.

A.1 Relation to Monomial Basis

A *Bézier Curve* of order n is equivalent to a polynomial with real coefficients of the same order over the interval $[0,1]$. To make this more evident, let us expand the rewrite the expression of a generic *Bernstein Polynomial*

$$\begin{aligned} b_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} &= \binom{n}{i} t^i \sum_{k=0}^{n-i} \binom{n-i}{k} (-1)^k t^k = \\ &= \sum_{j=i}^{n} \binom{n}{i} \binom{n-i}{j-i} (-1)^{j-i} t^j \end{aligned}$$

Now, given a *Bézier Curve* $p(t)$ of order n with control points $P_i \in \mathbb{R}$, we can expand it as

$$p(t) = \sum_{i=0}^{n} b_i^n(t) P_i = \sum_{i=0}^{n} P_i \sum_{j=i}^{n} \binom{n}{i} \binom{n-i}{j-i} (-1)^{j-i} t^j = \sum_{k=0}^{n} a_k t^k \tag{A.1}$$

which is indeed a polynomial of order n written in the canonical monomial basis, with coefficients

$$a_k = \sum_{i=0}^{k} (-1)^{k-i} \binom{n-i}{k-i} \binom{n}{i} P_i \tag{A.2}$$

If we stack all the *Control Points* in a vector[1] $\mathbf{P} = \begin{bmatrix} P_0 & P_1 & \cdots & P_n \end{bmatrix}^T$, and all the monomial coefficients in another vector $\mathbf{a} = \begin{bmatrix} a_0 & a_1 & \cdots & a_n \end{bmatrix}^T$ we can construct the change of basis matrix $M \in \mathbb{R}^{n+1 \times n+1}$ that relates the coordinates of a polynomial in the *Bézier Curve* basis to the monomial ones

$$\mathbf{a} = M\mathbf{P}$$
$$\mathbf{P} = M^{-1}\mathbf{a}$$

where M is defined as

$$M_{(i,j)} = \begin{cases} \sum_{j=0}^{i} (-1)^{i-j} \binom{n-j}{i-j} \binom{n}{j}, & j \leq i \\ 0, & j > i \end{cases}$$

In the general cases of a *Bézier Curve* belonging to a d space, i.e. $P_i \in \mathbb{R}^d$, equations (A.1) and (A.2) still holds, and we can construct $\mathbf{P}$ and $\mathbf{a}$ by stacking the coefficients as previously mentioned, which means $\mathbf{P} = \begin{bmatrix} P_0^T & P_1^T & \cdots & P_n^T \end{bmatrix}^T$ and $\mathbf{a} = \begin{bmatrix} a_0^T & a_1^T & \cdots & a_n^T \end{bmatrix}^T$. In this case, the change of basis matrix $\bar{M} \in \mathbb{R}^{d(n+1) \times d(n+1)}$ can be obtained with a Kronecker product between the matrix M and the identity matrix of dimension d

$$\bar{M} = M \otimes I_d$$

A.2 Remarkable Properties

Theorem A.2.1 ([36]). *Let $p(t) : [0,1] \to \mathbb{R}^d$ be a* Bézier Curve *of order n. Then $p(t) = P_0$ and $p(1) = P_n$.*

Proof. The statement is a direct consequence of the definition of the *Bernstein Polynomials*; by substitution in their formula we have

$$\begin{cases} b_0^n(0) = 1 \\ b_i^n(0) = 0, \quad i \neq 0 \end{cases} \qquad \begin{cases} b_n^n(1) = 1 \\ b_i^n(1) = 0, \quad i \neq n \end{cases}$$

This means that

$$p(0) = \sum_{i=0}^{n} b_i^n(0) P_i = P_0$$
$$p(1) = \sum_{i=0}^{n} b_i^n(1) P_i = P_n$$

[1] The set of polynomial of order n over the interval $[0,1]$ forms indeed a vector space and the *Bernstein Polynomial* of order n forms one of its orthonormal bases, so the collection of control points are the coordinates of an element of this polynomial vector space expressed with respect this basis. Also the monomials of degree $0 \leq i < n$ forms an ortonormal basis for this vector space, the canonical basis, so the collection of the multiplicative coefficients forms another set of coordinates.

□

Theorem A.2.2 ([36]). *Let* $p(t) : [0,1] \to \mathbb{R}^d$ *be a* Bézier Curve *of order* n. *Then* $\dot{p}(t)$ *is* Bézier Curve *of order* $n-1$ *and so that the* i^{th}, $i \in \{0,1,\ldots,n-1\}$, *control point is equal to* $n\,(P_{i+1} - P_i)$.

Corollary A.2.2.1. *Let* $p(t) : [0,1] \to \mathbb{R}^d$ *be a* Bézier Curve *of order* n. *Then* $\frac{d^r p}{d^r t}(t)$ *is* Bézier Curve *of order* $n-r$, $0 \geq r \geq n$. *Moreover, if we define the matrices* $D_n^r \in \mathbb{R}^{n+1-r \times n+1}$ *as*

$$D_n^1 = \begin{bmatrix} -1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & -1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & -1 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & 0 & 0 & \ldots & -1 & 1 \end{bmatrix}$$

$$D_n^r = D_r^{r-1}, n \geq r \geq 2$$

and we also define $\bar{D}_n^r = D_n^r \otimes I_d$, *then the control points of* $\frac{d^r p}{d^r t}(t)$ *are the elements of the vector* $\bar{D}_n^r \mathbf{P}$.

(A.3)

Theorem A.2.3 ([36]). *Let* $p(t) : [0,1] \to \mathbb{R}^d$ *be a* Bézier Curve *of order* n. *Then* $p(t) \in conv\left(\{P_i\}_{i=0}^n\right)$

Appendix B

Gaussian Processes

This chapter recalls some facts and definitions, taken from [48], about Gaussian Processes and Gaussian Process Regression that are used in the manuscript; the interested reader may find a more exhaustive introduction to the subject in [46].

Before delving into the subject, we want to highlight that Gaussian Processes have also an interesting interpretation that stems from the *Reproducing Kernel Hilbert Space* (RKHS) theory, that treats them as functional basis for regression and interpolation without invoking any probabilistic arguments. Readers that wants to learn more can find in [48] a discussion about the connections of the two subjects, and in [56] a complete introduction to the RKHS theory.

B.1 Gaussian Processes

Definition B.1.1 (Positive Definite Kernels)**.** *Let $\mathcal{X}$ be a nonempty set. A* symmetric[1] *function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a* positive definite kernel*, if for any $n \in \mathbb{N}$, $(c_1, \dots, c_N) \subset \mathbb{R}$ and $(x_1, \dots, x_N) \subset \mathcal{X}$*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j k(x_i, x_j) \geq 0$$

Definition B.1.2 (Kernel Matrix)**.** *Let $\mathcal{X}$ be a nonempty set, let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel, and let $X = (x_1, \dots, x_n) \subset \mathcal{X}$ be a finite set with $N \in \mathbb{N}$ elements. Then, we call* Kernel matrix*, or* Gram matrix*, the matrix $K_X \in \mathbb{R}^{N \times N}$ defined by $K_{X(i,j)} = k(x_i, x_j)$.*

The following are two of the most used kernel families for regressions [46].

Definition B.1.3 (Matérn Kernels)**.** *Let $\mathcal{X} \in \mathbb{R}^d$. For constants $\alpha > 0$ and $h > 0$, the* Matérn Kernel *$k_{\alpha,h} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$k_{\alpha,h}(x, x') = \frac{1}{2^{\alpha-1}\Gamma(\alpha)} \left(\frac{\sqrt{2\alpha}\|x - x'\|_2}{h} \right)^{\alpha} K_\alpha \left(\frac{\sqrt{2\alpha}\|x - x'\|_2}{h} \right)$$

where Γ is the Gamma Function*[57, 58], and K_α is the* modified Bessel Function of the second kind *of order α[57, 59].*

[1] A function $k : \mathcal{X} \to \mathcal{X} \to \mathbb{R}$ is said to be symmetric if for any $x_1, x_2 \in \mathcal{X}$ we have $k(x_1, x_2) = k(x_2, x_1)$

When $\alpha = m + \frac{1}{2}$ with $m \in \mathbb{N}$, then the above expression reduces to

$$k_{\alpha,h}(x,x')=\exp\left(-\frac{\sqrt{2\alpha}\|x-x'\|_2}{h}\right)\frac{\Gamma(m+1)}{\Gamma(2m+1)}\sum_{i=1}^{m}\frac{(m+1)!}{i!(m-1)!}\left(\frac{\sqrt{8\alpha}\|x-x'\|_2}{h}\right)^{m-i}$$

For instance, when $\alpha = \frac{1}{2}$ or $\alpha = \frac{3}{2}$, we have

$$k_{\frac{1}{2},h}(x,x') = \exp\left(-\frac{\|x-x'\|_2}{h}\right)$$

$$k_{\frac{3}{2},h}(x,x') = \left(1+\frac{\sqrt{3}\|x-x'\|_2}{h}\right)\exp\left(-\frac{\|x-x'\|_2}{h}\right)$$

Definition B.1.4 (Square Exponential Kernel)**.** *Let $\mathcal{X} \in \mathbb{R}^d$. For constant $\sigma > 0$ the* Square Exponential Kernel *$k_\sigma : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined by*

$$k_\sigma(x,x') = \exp\left(-\frac{\|x-x'\|_2^2}{2\sigma^2}\right) \tag{B.1}$$

We can now state the formal defition of *Gaussian Process*.

Definition B.1.5 (Gaussian Processes)**.** *Let $\mathcal{X}$ be a nonempty set, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel and $\mu : \mathcal{X} \to \mathbb{R}$be any real-valued function. Then a random function $f : \mathcal{X} \to \mathbb{R}$ is said to be* Gaussian Process *(GP) with mean function μ and covariance kernel k, denoted by $\mathcal{GP}(\mu, k)$, if for any finite set $X = (x_1, \ldots, x_N) \subset \mathcal{X}$ of any size $N \in \mathbb{N}$, the random vector*

$$f_X = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix} \in \mathbb{R}^N$$

follows the multivariate normal distribution $\mathcal{N}(\mu_X, K_X)$ with where k_X is the kernel matrix of k and $\mu_X = \left[\mu(x_1), \ldots, \mu(x_N)\right]^T$ is the mean vector of the distribution.

B.2 Gaussian Process Regression

B.2.1 Regression

Let $\mathcal{X}$ be a nonempty set and $f : \mathcal{X} \to \mathbb{R}$ be a function. Assume that one is given a set of pairs $(x_i, y_i)_{i=1}^N \subset \mathcal{X} \times \mathbb{R}$ for $N \in \mathbb{N}$, which will be referred to as the *training data*, such that

$$y_i = f(x_i) + \xi_i$$

where $\xi_i \sim \mathcal{N}\left(0, \sigma^2\right)$ is a zero-mean random variable, with variance $\sigma^2 > 0$, that models the noise on the output measurements of the function. The *regression* problem is to estimate the unknown function f based on the training data $(x_i, y_i)_{i=1}^N$. The function f is called the *regression function*, and is the conditional expectation of the output given an input

$$f(x,y) = \mathbb{E}\left[y|x\right]$$

We will denote the set of input data points as $X = \{x_1, \ldots, x_N\} \in X^N$ and the set of collected outputs as the vector $Y = [y_1, \ldots, y_N]^T \in \mathbb{R}^N$.

B.2.2 Gaussian Process Regression

Gaussian Process Regression is a tecnique to solve the regression problem based on the Bayes Rules for conditional probability. This techinique assume that the unknown regression function f can be modeled as a Gaussian Process with prior mean $m : \mathcal{X} \to \mathbb{R}$ and covariance kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, i.e. $f \sim \mathcal{GP}(m, k)$, which can be used to encode prior knowledge about the regression function. Due to the properties of Gaussian Processes (see [46, 48] for a more detailed explanation), the posterior distribution of f given the dataset $(x_i, y_i)_{i=1}^N$ is again a Gaussian Process with posterion mean $\mu : \mathcal{X} \to \mathbb{R}$ and covariance $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which are given by

$$\mu(x) = m(x) + k_X(x)^T \left(K_X + \sigma^2 I_N\right)^{-1} (y - m(x))$$
$$\kappa(x, x') = k(x, x') - k_X(x)^T \left(K_X + \sigma^2 I_N\right)^{-1} k_X(x'),$$

where K_X is the the Kernel Matrix of the collected samples and $k_X(x) = [k(x, x_1), \ldots, k(x, x_N)]^T$.

Bibliography

[1] N. Boumal, *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[2] W. G. Walter, "An imitation of life," *Scientific american*, vol. 182, no. 5, pp. 42–45, 1950.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[4] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in applied mathematics*, vol. 11, no. 4, pp. 412–442, 1990.

[5] E. Rimon, *Exact robot navigation using artificial potential functions*. Ph.D. Thesis, Yale University, 1990.

[6] E. Rimon and D. E. Koditschek, "Exact robot navigation in geometrically complicated but topologically simple spaces," in *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 1937–1942, IEEE, 1990.

[7] E. Rimon and D. E. Koditschek, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Transactions of the American Mathematical Society*, vol. 327, no. 1, pp. 71–116, 1991.

[8] S. G. Loizou, "The navigation transformation," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1516–1523, 2017.

[9] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," in *2014 IEEE international conference on robotics and automation (ICRA)*, pp. 6567–6572, IEEE, 2014.

[10] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[11] J.-C. Latombe, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[13] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*, pp. 5420–5425, IEEE, 2010.

[14] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*, pp. 2520–2525, IEEE, 2011.

[15] D. Mellinger, *Trajectory generation and control for quadrotors.* Ph.D Thesis, University of Pennsylvania, 2012.

[16] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE conference on decision and control*, pp. 6271–6278, IEEE, 2014.

[17] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[18] 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)*, pp. 3420–3431, IEEE, 2019.

[19] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *2016 American Control Conference (ACC)*, pp. 322–328, IEEE, 2016.

[20] W. Xiao and C. Belta, "High-order control barrier functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3655–3662, 2021.

[21] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8129–8136, IEEE, 2021.

[22] M. Krstic, "Inverse optimal safety filters," *IEEE Transactions on Automatic Control*, vol. 69, no. 1, pp. 16–31, 2023.

[23] G. Wu and K. Sreenath, "Safety-critical control of a planar quadrotor," in *2016 American control conference (ACC)*, pp. 2252–2258, IEEE, 2016.

[24] G. Wu and K. Sreenath, "Safety-critical control of a 3d quadrotor with range-limited sensing," in *Dynamic Systems and Control Conference*, vol. 50695, p. V001T05A006, American Society of Mechanical Engineers, 2016.

[25] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE transactions on robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[26] A. Lakshmanan, *Piecewise Bézier curve trajectory generation and control for quadrotors.* Master's Thesis, University of Illinois at Urbana-Champaign, 2016.

[27] B. Sabetghadam, R. Cunha, and A. Pascoal, "Real-time trajectory generation for multiple drones using bézier curves," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9276–9281, 2020.

[28] J. Park and H. J. Kim, "Online Trajectory Planning for Multiple Quadrotors in Dynamic Environments Using Relative Safe Flight Corridor," *IEEE Robotics and Automation Letters*, vol. 6, pp. 659–666, Apr. 2021. Conference Name: IEEE Robotics and Automation Letters.

[29] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, "Online Distributed Trajectory Planning for Quadrotor Swarm With Feasibility Guarantee Using Linear Safe Corridor," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4869–4876, Apr. 2022.

[30] H. Zhu, B. Brito, and J. Alonso-Mora, "Decentralized Probabilistic Multi-Robot Collision Avoidance Using Buffered Uncertainty-Aware Voronoi Cells," *arXiv:2201.04012 [cs]*, Jan. 2022. arXiv: 2201.04012.

[31] H. Zhu and J. Alonso-Mora, "B-UAVC: Buffered Uncertainty-Aware Voronoi Cells for Probabilistic Multi-Robot Collision Avoidance," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 162–168, Aug. 2019.

[32] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, On-line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1047–1054, Apr. 2017. Conference Name: IEEE Robotics and Automation Letters.

[33] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[34] Y. Wu, X. Sun, I. Spasojevic, and V. Kumar, "Deep learning for optimization of trajectories for quadrotors," *IEEE Robotics and Automation Letters*, 2024.

[35] L. Carlone, K. Khosoussi, V. Tzoumas, G. Habibi, M. Ryll, R. Talak, J. Shi, and P. Antonante, "Visual navigation for autonomous vehicles: An open-source hands-on robotics course at mit," in *2022 IEEE Integrated STEM Education Conference (ISEC)*, pp. 177–184, IEEE, 2022.

[36] R. T. Farouki, "The bernstein polynomial basis: A centennial retrospective," *Computer Aided Geometric Design*, vol. 29, no. 6, pp. 379–419, 2012.

[37] P. Ong and J. Cortés, "Universal formula for smooth safe stabilization," in *2019 IEEE 58th conference on decision and control (CDC)*, pp. 2373–2378, IEEE, 2019.

[38] H. K. Khalil, *Nonlinear Systems Third Edition*. Pearson; 3rd edition, 2001.

[39] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[40] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[41] L. Wu, K. M. B. Lee, L. Liu, and T. Vidal-Calleja, "Faithful euclidean distance field from log-Gaussian process implicit surfaces," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2461–2468, 2021.

[42] B. Trimarchi, L. Gentilini, F. Schiano, and L. Marconi, "Data-driven analytic differentiation via high gain observers and gaussian process priors," in *2023 American Control Conference (ACC)*, pp. 3056–3061, IEEE, 2023.

[43] M. Khan and A. Chatterjee, "Gaussian control barrier functions: Safe learning and control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3316–3322, IEEE, 2020.

[44] M. Khan, T. Ibuki, and A. Chatterjee, "Safety uncertainty in control barrier functions using Gaussian processes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6003–6009, IEEE, 2021.

[45] M. A. Khan, T. Ibuki, and A. Chatterjee, "Gaussian control barrier functions: Non-parametric paradigm to safety," *IEEE Access*, vol. 10, pp. 99823–99836, 2022.

[46] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[47] A. Lederer, J. Umlauft, and S. Hirche, "Uniform error bounds for Gaussian process regression with application to safe control," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[48] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, "Gaussian processes and kernel methods: A review on connections and equivalences," *arXiv preprint arXiv:1807.02582*, 2018.

[49] L. Wu, K. M. B. Lee, C. Le Gentil, and T. Vidal-Calleja, "Log-GPIS-MOP: A unified representation for mapping, odometry, and planning," *IEEE Transactions on Robotics*, 2023.

[50] S. Mauch, "A fast algorithm for computing the closest point and distance transform, 2000," *Source: http://www-cgrl.cs.mcgill. ca/˜ godfried/teaching/cgprojects/98/normand/main.html*.

[51] S. R. S. Varadhan, "On the behavior of the fundamental solution of the heat equation with variable coefficients," *Communications on Pure and Applied Mathematics*, vol. 20, no. 2, pp. 431–455, 1967.

[52] A. Tornambè, "High-gain observers for non-linear systems," *International Journal of Systems Science*, vol. 23, no. 9, pp. 1475–1489, 1992.

[53] A. Lederer, J. Umlauft, and S. Hirche, "Uniform error and posterior variance bounds for Gaussian process regression with application to safe control," *arXiv preprint arXiv:2101.05328*, 2021.

[54] E. Solak, R. Murray-Smith, W. Leithead, D. Leith, and C. Rasmussen, "Derivative observations in Gaussian process models of dynamic systems," *Advances in neural information processing systems*, vol. 15, 2002.

[55] B. Trimarchi, F. Schiano, and R. Tron, "A control barrier function candidate for limited field of view sensors," *arXiv preprint arXiv:2410.01277*, 2024.

[56] J. H. Manton, P.-O. Amblard, *et al.*, "A primer on reproducing kernel Hilbert spaces," *Foundations and Trends in Signal Processing*, vol. 8, no. 1–2, pp. 1–126, 2015.

[57] R. A. Askey and R. Roy, "Gamma function," in *NIST handbook of mathematical functions* (F. W. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, eds.), ch. 5, Cambridge university press, 2010.

[58] P. Sebah and X. Gourdon, "Introduction to the gamma function," *American Journal of Scientific Research*, pp. 2–18, 2002.

[59] E. W. Weisstein, "Modified Bessel function of the second kind," *https://mathworld. wolfram. com/*, 2002.