# Safe Input Regulation for Robotic Systems

Thesis by
Andrew Wills Singletary

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy


CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2022
Defended May 19, 2022

Andrew Wills Singletary
ORCID: 0000-0001-6635-4256

# ACKNOWLEDGEMENTS

To Winston and Daisy.

# ABSTRACT

The safety of robotic systems is paramount to their continued emergence into our lives. From collaborative industrial manipulators to drone deliveries to autonomous vehicles, safety is the primary concern when it comes to the continued adoption of these technologies. While a number of techniques can be used to design safe controllers and planners that govern the actions of these robots, few are able to provide the type of safety guarantee needed to bring these technologies into reality.

The goal of this thesis is to provide a framework for regulating, or filtering, existing control inputs before they are applied by the robot, in order to ensure that safety is upheld. To illustrate this, consider one of the primary applications for this method: human-operated robotic platforms. For vehicles, this framework would modify the throttle, braking, and steering commands from a human driver to prevent him from driving off the road or into other cars. However, when the human is operating the vehicle safely, his commands should go unaltered. This illustrates the idea of a minimally invasive safety regulator: one that only engages when absolutely necessary to ensure safety.

Within the last decade, the mathematical framework that allows us to achieve this result, control barrier functions, was introduced. Its adoption among the nonlinear controls community has been rapid, and the method has been used to create controllers that guarantee safety on a large class of systems. Despite this, real-world implementations of control barrier functions are less common, since they require a very accurate model of the system, and they can be difficult to formulate properly. This work provides several major extensions, improvements, and modifications of control barrier functions that allow them to be utilized on a variety of real-world robotic systems.

The first major contribution of this thesis is a set of formulations for safety regulators that do not depend on complete knowledge of the underlying dynamical systems. Three unique formulations are proposed, whose usages depend on the level of knowledge of the underlying system. The resulting performance and safety guarantees are analyzed in real-world applications of quadrotor collision avoidance and fast-food frying with industrial manipulators. The second major contribution is a set of two safety filtering frameworks that utilize knowledge of the full-order dynamics, but allow for guaranteed safety in the presence of input constraints on

high-dimensional systems. Two formulations are given, with one designed for use on microcontrollers with minimal computational resources. Both formulations utilize the knowledge of an existing "backup controller" that attempts to take the system into a small, safe "backup set". This method is demonstrated in simulation on a robotic manipulator and a Segway robot, and on hardware for collision avoidance and geofencing of single and multi-agent racing drones. The third major contribution is a novel discrete-time formulation of control barrier functions that allow for safety regulation of discrete-time systems. We show how safety constraints can be encoded as temporal logic specifications that are enforced over discrete-time models of the systems and their environments. The fourth and final major contribution is a unified, multi-rate control framework that guarantees safety at both the high-level, in discrete-time, and the low-level, in continuous-time. A mid-level Model Predictive Controller (MPC) is used to generate reference signals based on the high-level planner which are tracked by the low-level controller.

Together, these four major contributions result in safe input regulation on a wide variety of robotic systems. Since no single method can reliably enforce safety on such a wide range of systems with different requirements, this thesis provides the smallest collection of methods that applies to the largest classes of systems.

# PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Ryan K Cosner et al. "Measurement-robust control barrier functions: Certainty in safety with uncertainty in state". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6286–6291. DOI: 10.1109/IROS51168.2021.9636584.
AWS led in the creation of the simulation environment, led the generation of the hardware results, and participated in the writing of the manuscript.

[2] Andrew Singletary et al. "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)*. IEEE, pp. 8129–8136. DOI: 10.1109/IROS51168.2021.9636670.
AWS led in the conception of the project, led in the development of the theory, led the implementation of the software and hardware results, generated the safety sets, and led in the writing of the manuscript.

[3] Andrew Singletary, Mohamadreza Ahmadi, and Aaron D Ames. "Safe Control for Nonlinear Systems with Stochastic Uncertainty via Risk Control Barrier Functions". In: *arXiv preprint arXiv:2203.15892* (2022). DOI: 10.48550/arXiv.2203.15892.
AWS participated in the conception of the project, created the simulation environment, helped generate the theory, and led in the writing of the manuscript.

[4] Andrew Singletary et al. "Onboard safety guarantees for racing drones: High-speed geofencing with control barrier functions". In: *IEEE Robotics and Automation Letters* (2022). DOI: 10.1109/LRA.2022.3144777.
AWS led in the conception of the project, created the simulation environment, led the hardware implementation, led the theory development, and led in the writing of the manuscript.

[5] Andrew Singletary et al. *Safety-Critical Manipulation for Collision-Free Food Preparation*. 2022. DOI: 10.48550/arXiv.2205.01026. URL: https://arxiv.org/abs/2205.01026.
AWS led the conception of the project, led in the implementation in the simulation and hardware, helped generate the theory, and led in the writing of the manuscript.

[6] Tamas G Molnar et al. "Model-free safety-critical control for robotic systems". In: *IEEE Robotics and Automation Letters* 7.2 (2021), pp. 944–951. DOI: 10.1109/LRA.2021.3135569.
AWS participated in the conception of the project, participated in the software and hardware implementation, generated the safety sets, and participated in the writing of the manuscript.

[7] Andrew Singletary, Shishir Kolathaya, and Aaron D Ames. "Safety-critical kinematic control of robotic systems". In: *IEEE Control Systems Letters* 6 (2021), pp. 139–144. DOI: 10.1109/LCSYS.2021.3050609.
AWS led in the conception of the project, helped generate the theory results, generated the simulation environment, led the implementation, and participated in the writing of the manuscript.

[8] Mohamadreza Ahmadi et al. "Barrier functions for multiagent-pomdps with dtl specifications". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 1380–1385. DOI: 10.1109/CDC42340.2020.9304266.
AWS participated in the conception of the project, created the simulation environment, generated the safety sets, and participated in the writing of the manuscript.

[9] Aaron D Ames et al. "Safety-critical control of active interventions for COVID-19 mitigation". In: *Ieee Access* 8 (2020), pp. 188454–188474. DOI: 10.1109/ACCESS.2020.3029558.
AWS participated in the conception of the project, created the dynamical system models, analyzed the real-world data, and participated in the writing of the manuscript.

[10] Yuxiao Chen, Andrew Singletary, and Aaron D Ames. "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: a control barrier function approach". In: *IEEE Control Systems Letters* 5.1 (2020), pp. 127–132. DOI: 10.1109/LCSYS.2020.3000748.
AWS participated in the conception of the project, created the simulation environment, generated the backup controllers and backup sets, and participated in the writing of the manuscript.

[11] Yuxiao Chen, Andrew W Singletary, and Aaron D Ames. "Density functions for guaranteed safety on robotic systems". In: *2020 American Control Conference (ACC)*. IEEE. 2020, pp. 3199–3204. DOI: 10.23919/ACC45564.2020.9147265.
AWS participated in the conception of the project, created the simulation environment, led the development of the hardware platform and results, and participated in the writing of the manuscript.

[12] Ruben Grandia et al. "Nonlinear model predictive control of robotic systems with control lyapunov functions". In: *arXiv preprint arXiv:2006.01229* (2020). DOI: 10.15607/RSS.2020.XVI.098.
AWS participated in the conception of the project, helped create the simulation environment, created the hardware demonstration, and participated in the writing of the manuscript.

[13] Thomas Gurriet et al. "A scalable safety critical control framework for nonlinear systems". In: *IEEE Access* 8 (2020), pp. 187249–187275. DOI: 10.1109/ACCESS.2020.3025248.

AWS participated in the conception of the project, led in the creation and development of many of the software and hardware examples, and participated in the writing of the manuscript.

[14] Tamás G Molnár et al. "Safety-critical control of compartmental epidemiological models with measurement delays". In: *IEEE Control Systems Letters* 5.5 (2020), pp. 1537–1542. DOI: 10.1109/LCSYS.2020.3040948.
AWS participated in the conception of the project, helped in the creation of the dynamical system models, and participated in the writing of the manuscript.

[15] Ugo Rosolia, Andrew Singletary, and Aaron D Ames. "Unified multi-rate control: from low level actuation to high level planning". In: *arXiv preprint arXiv:2012.06558* (2020). DOI: 10.48550/arXiv.2012.06558.
AWS participated in the conception of the project, created the simulation environment, helped in the implementation of the code and hardware, and participated in the writing of the manuscript.

[16] Andrew Singletary, Yuxiao Chen, and Aaron D Ames. "Control barrier functions for sampled-data systems with input delays". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 804–809. DOI: 10.1109/CDC42340.2020.9304281.
AWS led in the conception of the project, led in the generation of the simulation and hardware results, led the theory development, and led in the writing of the manuscript.

[17] Andrew Singletary et al. "Safety-critical rapid aerial exploration of unknown environments". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 10270–10276. DOI: 10.1109/ICRA40945.2020.9197416.
AWS led in the conception of the project, created the simulation environment, generated the backup control law, and led in the writing of the manuscript.

[18] Andrew Taylor et al. "Learning for safety-critical control with control barrier functions". In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 708–717.
AWS participated in the conception of the project, helped created the simulation environment, generated the CBFs, led in the hardware development and testing, and participated in the writing of the manuscript.

[19] Andrew J Taylor et al. "A control barrier perspective on episodic learning via projection-to-state safety". In: *IEEE Control Systems Letters* 5.3 (2020), pp. 1019–1024. DOI: 10.1109/LCSYS.2020.3009082.
AWS participated in the conception of the project, helped create the simulation environment, led in the generation of the hardware results, and participated in the writing of the manuscript.

[20] Mohamadreza Ahmadi et al. “Safe policy synthesis in multi-agent POMDPs via discrete-time barrier functions”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 4797–4803. DOI: 10.1109/CDC40024.2019.9030241.
AWS participated in the conception of the project, created the simulation environment, generated the safety conditions, and participated in the writing of the manuscript.

[21] Thomas Gurriet et al. “A scalable controlled set invariance framework with practical safety guarantees”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 2046–2053. DOI: 10.1109/CDC40024.2019.9030159.
AWS participated in the conception of the project, helped create the simulation environment, helped create the backup controller and backup sets, and participated in the writing of the manuscript.

[22] Thomas Gurriet et al. “Realizable set invariance conditions for cyber-physical systems”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 3642–3649. DOI: 10.23919/ACC.2019.8815332.
AWS participated in the conception of the project, helped design the digital controller implementation, helped create the simulation environment, and participated in the writing of the manuscript.

[23] Andrew Singletary et al. “Online active safety for robotic manipulators”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 173–178. DOI: 10.1109/IROS40897.2019.8968231.
AWS led in the conception of the project, created the simulation environment, generated the safety sets, and led in the writing of the manuscript.

[24] Thomas Gurriet et al. “Towards a framework for realizable safety critical control through active set invariance”. In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE. 2018, pp. 98–106. DOI: 10.1109/ICCPS.2018.00018.
AWS participated in the conception of the project, performed the safety set computations, helped prepare the hardware, and participated in the writing of the manuscript.

# TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

# LIST OF TABLES

*Chapter 1*

# INTRODUCTION

Ensuring safety of robotic systems is the primary focus of this work. To achieve this, we plan to leverage results from control theory.

## 1.1 Safety-critical control

Safety-critical control is a subset of control theory that focuses on providing mathematical guarantees of safety for dynamical systems. This can be in the form of formal verification of an autonomous system, or in the form of a controller that is guaranteed to keep a control system safe.

Frequently, the notion of safety in robotics involves the avoidance of obstacles in the environment. Artificial Potential Fields (APFs) have been utilized for over thirty years in the context of real-time obstacle avoidance. They were first in the seminal paper by Khatib [81], and has since been developed further, beginning with computational methods [23]. Of particular importance to this work, there has been significant work in applying APFs to obstacle avoidance [160, 93], including dynamic obstacles [55]. Work has also been been done on improving the behavior of artificial potential fields, particularly in dealing with undesirable oscillation behavior [122]. Finally, the search for effective methods for path planning using APFs has continued [96], including application to UAV path planning [34]. While potential fields have been shown to work well in practice, they are heavily dependent on tuning parameters, and offer no formal verification of safety for robotic systems.

Safety verification for a robotic system can be encoded as checking whether the system remains inside a pre-specified safe set or alternatively avoids a pre-defined unsafe set. Then, a natural method for checking safety is to compute the reachable set of a system subject to disturbances and controls [102, 1, 12]. However, for complex and high-dimensional systems such methods are either intractable, or overly conservative. Alternative approaches to reachability date back to the pioneering works of Nagumo [107] to study the set invariance of ordinary differential equations (ODEs). Nagumo's works were extended to ODEs with inputs by Aubin *et. al.* [22] in the context of viability theory. The interest in hybrid systems in the 2000's led to the introduction of barrier certificates for safety verification [117]. However, the

construction of these barrier certificates require solving a set of polynomial optimization problems that become intractable for high-dimensional systems (despite some promising recent directions [4]).

The recently proposed notion of control barrier functions [15] circumvent the computational bottleneck of barrier certificates inasmuch as the closed-form expression for a barrier function can be derived from the definition of the safe set. By taking advantage of this property, barrier functions have been used for designing safe controllers (in the absence of a nominal controller) and safety filters (in the presence of a nominal controller) for dynamical systems, such as biped robots [108] and trucks [33], with guaranteed performance and robustness [166, 83].

Control barrier functions are similar to control Lyapunov functions [143, 51], which have been used to guarantee asymptotic stability for robotic systems [111]. However, control barrier functions allow for freedom of movement inside of the sets that they define, whereas a control Lyapunov function will enforce convergence to an equilibrium point.

The field of formal methods is also used to verify safety properties of autonomous systems. As shown in [163], correctness guarantees can be formalized over specifications that define safe behaviors, and control strategies can be synthesized that meet these specifications. More generally, methods like abstract interpretation [46] and model-checking [42] can be used to automatically generate proofs of correctness for software systems. There is not a framework, however, for regulating control inputs of general robotic systems utilizing techniques from formal methods.

While the concept of minimally invasive input regulation is less studied than system verification, several approaches exist in the literature. In [30], the authors propose a sampling-based MPC approach that generates many possible safe trajectories at each time-step, and chooses the one closest to the user's desired input subject to safety conditions. Another MPC-based approach is demonstrated in [148], which uses learning to minimize conservatism, but neither of these approaches are able to run in real-time on a microcontroller. Explicit Reference Governor (ERG) schemes modifies the derivative of the applied inputs subject to safety constraints utilizing Lyapunov functions, as shown in [66]. While this approach is optimization-free and could be implemented online, it is difficult to find the required upper-bound of the Lyapunov function that guarantees constraint satisfaction.

More extensive literature reviews and comparisons to existing methods can be found

in the individual sections of this thesis, to better relate each individual contribution to works that relate to it.

## 1.2 Control barrier functions

To achieve the safety guarantees desired in this work, we will extensively utilize control barrier functions. Before we define a control barrier function, we must first define several other properties.

To begin, we look at the class of systems that we will be studying: nonlinear dynamical systems in control-affine form.

**Definition 1** (Control system)**.** *A **nonlinear, control-affine system** is a dynamical system such that*

$$\dot{x} = f(x) + g(x)u, \tag{1.1}$$

*where $x \in \mathcal{D} \subset \mathbb{R}^n$ is the state, and $u \in U \subseteq \mathbb{R}^m$ the input. Here, $\mathcal{D}$ is the domain of the state space, and $U$ is the admissable input set. Assume that the functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are continuously differentiable.*

Next, we must formally define our notion of safety. The goal is to keep a system inside of some safe set $S \subset \mathcal{D} \subset \mathbb{R}^n$. This can be formalized through the notion of set invariance.

**Definition 2** (Set invariance)**.** *A set $\mathcal{S}$ is called **invariant** if the system's state stays in $\mathcal{S}$ for all time, i.e., $\forall t \geq t_0,\ x(t) \in \mathcal{S}$. Moreover, for* (1.1)*, a set $\mathcal{S}$ is **control invariant** if, for all time, there always exists an input $u \in U$ such that the system stays in $\mathcal{S}$ for all time, i.e. $\forall t \geq t_0, \exists u \in U$ s.t. $x(t) \in \mathcal{S}$.*

With the system and the concept of set invariance defined, we can now define a control barrier function.

### 1.2.1 Continuous-time

First, we begin with the (more common) continuous-time formulation.

**Definition 3** (Control barrier function)**.** *Let the safe set $\mathcal{S} \subset \mathcal{D} \subset \mathbb{R}^n$ be the **0-superlevel set** of a continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$:*

$$\begin{aligned} \mathcal{S} &= \{x \in \mathbb{R}^n \ :\ h(x) \geq 0\}, \\ \partial\mathcal{S} &= \{x \in \mathbb{R}^n \ :\ h(x) = 0\}, \\ \mathrm{Int}(\mathcal{S}) &= \{x \in \mathbb{R}^n \ :\ h(x) > 0\}. \end{aligned} \tag{1.2}$$

*Then $h$ is a* ***control barrier function (CBF)*** *if $\frac{\partial h}{\partial x}(x) \neq 0$ for all $x \in \partial\mathcal{S}$ and there exists an* extended class $\mathcal{K}$ function *([15, Definition 2]) $\alpha$ such that for the control system* (1.1) *and for all $x \in \mathcal{S}$:*

$$\sup_{u \in U} \Big[ \underbrace{L_f h(x) + L_g h(x) u}_{\dot{h}(x,u)} \Big] \geq -\alpha(h(x)), \tag{1.3}$$

*where $L_f h(x) = \frac{\partial h}{\partial x} f(x)$ and $L_g h(x) = \frac{\partial h}{\partial x} g(x)$. We say that $h$ is a* ***control barrier function (CBF) on $\mathcal{S}$*** *if* (1.3) *holds for all $x \in \mathcal{S}$ (but not necessarily on all of $\mathcal{D}$).*

The main control barrier function result is that this class of functions give sufficient (and necessary) conditions on set invariance, i.e., safety of the system relative to $\mathcal{S}$. Note that, in the literature, this formulation may sometimes be referred to as a "zeroing" control barrier function, but this terminology is not utilized in this work, as we do not consider reciprocal CBFs.

**Theorem 1** ([15])**.** *Given a control barrier function $h : \mathbb{R}^n \to \mathbb{R}$ together with the associated set $\mathcal{S}$, for any Lipschitz continuous controller satisfying:*

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

*the set $\mathcal{S}$ is forward invariant, i.e,. safe. Additionally, the set $\mathcal{S}$ is asymptotically stable.*

Since the constraint (1.3) is affine in $u$, the above definition can be used to construct a quadratic program that functions as a safety filter, guaranteeing the safety of the system by enforcing it to stay inside of $\mathcal{S}$. This quadratic program is given by

$$\begin{aligned} u^*(x) = \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \quad & \|u - u_{\text{des}}(x,t)\|^2 \\ \text{s.t.} \quad & L_f h(x) + L_g h(x) u \geq -\alpha(h(x)) \end{aligned} \tag{CBF-QP}$$

Importantly, this QP has an explicit solution given by

$$u^*(x,t) = u_{\text{des}}(x,t) + u_{\text{safe}}(x,t) \tag{1.4}$$

where $u_{\text{safe}}$ is added to $u_{\text{des}}$ if the nominal controller would not keep the system safe, which is determined by the sign of $\Psi(x,t;u_{\text{des}}) := \dot{h}(x, u_{\text{des}}(x,t)) + \alpha(h(x))$ via:

$$u_{\text{safe}}(x,t) = \begin{cases} -\frac{L_g h(x)^T}{L_g h(x) L_g h(x)^T} \Psi(x,t;u_{\text{des}}) & \text{if } \Psi < 0 \\ 0 & \text{if } \Psi \geq 0 \end{cases} \tag{1.5}$$

### 1.2.2 Discrete-time

The use of discrete-time control barrier functions is significantly more limited, due to their computational complexity. While the continuous-time CBF is an affine constraint for a nonlinear system, the same is not true for discrete-time CBFs. One formulation exists in the literature [3], based on the reciprocal CBF formulation, but its formulation will not be copied here.

Instead, in this thesis, we will extend the zeroing CBF formulation to discrete-time systems, and present algorithms to utilize them on robotic systems.

## 1.3 Safety regulation

The specific focus of this work is the idea of regulating a desired control input in a minimally-invasive way, subject to safety constraints. This is not a typical task in control theory or robotics, as it is generally preferred to incorporate the safety constraints into the original controller or planner for the system. However, this approach to filter a desired signal, rather than generate safe signals from the start, has a number of unique benefits.

- Easy integration with human operators. In this case, human operators provide the desired input signal that is tracked. Humans operators are generally not able to guarantee the safety of the systems they operate themselves, so this framework uniquely allows them to operate the system while providing formal safety guarantees.
- Verification of systems running complicated ML-based algorithms. It is very difficult to verify the safety of these controllers generated with machine learning, but by filtering the outputs of these models, we can guarantee safety while maintaining their performance advantages.

## 1.4 Outline

This thesis is broken down into four major chapters, all of which are centered around the use of CBFs to regulate safety of robotic systems.

Chapter 2 focuses on kinematic and model-free safety regulation. Here, we break the norm of enforcing the CBF condition at the low-level with forces and torques, and instead enforce the conditions at the velocity layer. Section 2.1 compares a simple velocity-layer kinematic barrier to artificial potential fields, one of the staple obstacle avoidance methods in robotics. Section 2.2 proposes a new type of dynamic

extension based on the kinetic energy of robotic systems to guarantee safety utilizing minimal model information. Lastly, Section 2.3 showcases the kinematic CBFs on a full-scale application of robotic cooking, while also providing theoretical results on the full-order dynamics utilizing information on the low-level controller's velocity tracking abilities.

Chapter 3 focuses on utilizing the knowledge of a single "backup controller" that tries to take the system to a small, safe region called the "backup set" to regulate control inputs. Section 3.1 covers how this knowledge can be used to construct "implicit" control invariant sets, allowing CBFs with input constraints to scale to higher-dimensional systems. This section details applications for robotic manipulators, sampled-data systems with input delays, and multi-agent drone systems. Section 3.2 utilizes the same knowledge of backup controllers and sets, but implements a gradient-free CBF approach that is much more computationally efficient. This is demonstrated on several drone platforms in applications of obstacle avoidance, geofencing, and multi-agent collision avoidance.

Chapter 4 proposes a new formulation for discrete-time barrier functions (DTBFs), utilizing the dynamics of belief-space updates of Multi-agent Partially Observable Markov Decision Processes (MPOMDPs). Section 4.1 formalizes this notion of DTBFs, while Section 4.2 shows how safety constraints can be formulated as linear distrubutional temporal logic (LDTL) specifications. Lastly, Section 4.3 shows how uncertainty can be formulated as risk using coherent risk measures, and how safety can be enforced over these risk measures using risk control barrier functions.

Finally, Chapter 5 offers a brief perspective on how safety can be guaranteed at all levels of a robotic system: the low-level control layer, the high-level decision-making layer, and the "mid-level" planning layer. The chapter concludes with a summary of the work, and possible future directions for this research area.

*Chapter 2*

# KINEMATIC AND MODEL-FREE SAFETY REGULATION

## 2.1 Comparison between potential fields and kinematic barrier functions

As mentioned in the introduction, Artificial Potential Fields (APFs) have been a staple for safety in the field of robotics for decades. Given their historic use and the recent popularity of control barrier functions, the natural question to ask is: *How do control barrier functions compare to artificial potential fields?* Since the main application of control barrier functions is to provide critical safety guarantees for the systems in which they are implemented, their use in the literature is generally heavily model-dependent. In contrast, we show in this work just how effectively control barrier functions can be utilized even without a model of the system dynamics.

The major contribution of this section is a comparative analysis of CBFs and APFs—both theoretically and through simulation and experimental results on obstacle avoidance. At a formal level, we establish that a broad class of APFs can be used to synthesize a specific instance of a CBF. Additionally, this translation results in beneficial properties: it (pointwise) optimally balances avoidance and goal attainment, is well defined if the system leaves the safe set, and allows one to easily generalize these APFs to nonlinear control systems. From a comparative perspective, we begin with simple examples to illustrate the beneficial properties of CBFs vs. APFs, followed by high-fidelity simulations of a quadrotor for different obstacle avoidance scenarios. Finally, these same scenarios are carried out experimentally on a quadrotor with onboard sensing. In these experiments, the CBFs outperform APFs in the context of providing smooth behaviors that are minimally invasive while avoiding the obstacles, even though the quadrotor dynamics were not utilized.

We begin by considering artificial potential fields (APFs) in the setting of obstacle avoidance. A variety of potential functions can be utilized, but this section will look at the original formulation [81]. In this context, consider a control system described by a single integrator:

$$\dot{x} = v, \tag{2.1}$$

with $x \in \mathbb{R}^n$ is the position, and $v \in \mathbb{R}^n$ is the velocity. Here $v$ is viewed to be the control input to the system. The goal is to synthesize a desired velocity profile that

reaches a goal position while avoiding one or multiple obstacles. The motivation for considering a single integrator is that the resulting behavior of this system can, for example, be utilized as desired velocity profiles for end-effector positions for a robot manipulator ($n = 3$) wherein classic Jacobian methods can be utilized [160].

To explicitly present artificial potential fields, per the original formulation in [81], let $x_{\text{goal}}$ the goal position. This exerts an attractive potential to the system given by:

$$U_{\text{att}}(x) = \frac{1}{2} K_{\text{att}} \left\| x - x_{\text{goal}} \right\|^2 . \tag{2.2}$$

Any obstacles in the area assert a repulsive potential, given by:

$$U_{\text{rep}}(x) = \begin{cases} 0 & \rho(x) > \rho_0 \\ \frac{1}{2} K_{\text{rep}} \left( \frac{1}{\rho(x)} - \frac{1}{\rho_0} \right)^2 & \rho(x) \leq \rho_0, \end{cases} \tag{2.3}$$

where $\rho(x)$ is the distance to the obstacle or the distance from a safe region around the obstacle, e.g.:

$$\rho(x) = \|x - x_{\text{obs}}\| - D_{\text{obs}}, \tag{2.4}$$

for $D_{\text{obs}} > 0$, and $\rho_0$ is the region of influence. The potential function is set to zero outside of this region to allow the attractive potential to dominate over large distances.

To obtain a feedback controller that pushes the system to the goal while avoiding obstacles, the attractive and repulsive potentials are combined and the gradient is taken

$$F_{\text{APF}}(x) = -\nabla U_{\text{att}}(x) - \nabla U_{\text{rep}}(x), \tag{2.5}$$

with $\nabla U_{-}(x) = \frac{\partial U_{-}}{\partial x}(x)^T$ and

$$\nabla U_{\text{att}}(x) = K_{\text{att}}(x - x_{\text{goal}}) \tag{2.6}$$

$$\nabla U_{\text{rep}}(x) = \frac{K_{\text{rep}}}{\rho(x)^2} \left( \frac{1}{\rho(x)} - \frac{1}{\rho_0} \right) \frac{(x - x_{\text{goal}})}{\rho(x)}. \tag{2.7}$$

For a single integrator (2.1), where one directly controls velocity, we simply apply this expression as the velocity input

$$\dot{x} = F_{\text{APF}}(x),$$

yielding a gradient dynamical system with respect to the attractive and repulsive potentials.

**Example 1.** *Consider a mobile robot modeled as a single integrator travelling in the plane:* $n = 2$. *The initial position is* $x_0 = (0, 0)$, *and the goal position is* $x_{goal} = (3, 5)$. *There are two obstacles, at* $x_{O1} = (1, 2)$ *and* $x_{O2} = (2.5, 3)$, *that the mobile robot must not come within 0.5 meters of these obstacles.*

*The new distance function for obstacle i is* $\rho = \|x - x_{Oi}\| - 0.5$. *Using* $K_{att} = K_{rep} = 1$, *with varying values of* $\rho_0$, *we have the result shown in Figure 2.1(a). As can be seen in this figure, the potential field works well at* $\rho_0$ *values of 1 and 0.25, but it gets stuck in a local minimum at* $\rho_0 = 0.5$, *and oscillations start to occur at* $\rho_0 = 0.1$ *at a controller frequency of 1000 Hz. To remove these oscillations, one would have to either increase the control frequency beyond 1000 Hz or decrease the maximum speed of the system.*

We again consider the the single integrator in (2.1), this time in the context of control barrier functions. For this system, we wish to formulate a safety-critical controller. We define the safe set $\mathcal{S}$ as the complement of any obstacles in the space. That is, one can utilize $\rho$ and define:

$$h(x) = \rho(x) = \|x - x_{\text{obs}}\| - D_{\text{obs}} \geq 0$$

where now the safe set $\mathcal{S}$ is the set we wished to render safe with the potential fields.

For the single integrator, the control barrier condition becomes

$$\dot{h}(x, v) = \nabla h(x)^T \dot{x} = \nabla h(x)^T v \geq -\alpha h(x) \tag{2.8}$$

for $\alpha > 0$.

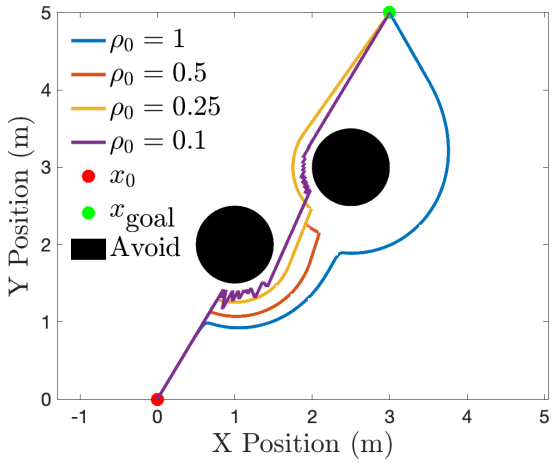
This can be framed as an optimization problem:

$$\begin{aligned} v^*(x, t) = \underset{v \in \mathbb{R}^n}{\text{argmin}} \;\; & \|v - v_{\text{des}}(x, t)\|^2 \\ \text{s.t.} \;\; & \nabla h(x)^T v \geq -\alpha h(x), \end{aligned} \tag{2.9}$$

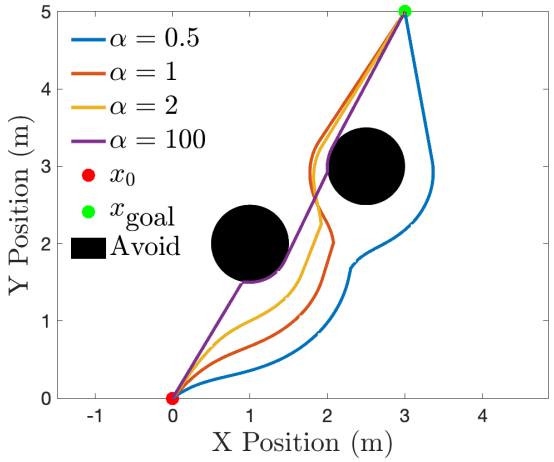
where $v^*(x, t)$ is the pointwise optimal controller. This is an important and substantial divergence from potential fields in that gradients are no longer used for synthesis. Rather, one can optimize over controllers that satisfy the safety constraint. To see how this difference manifests itself, we return to Example 1 but instead apply control barrier functions.

**Example 2.** *Consider the same setup as Example 1. The desired velocity command is a simple P controller on position,*

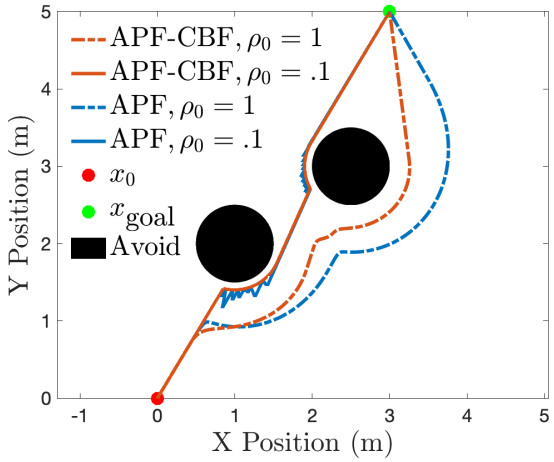
$$v_{des}(x, t) = -K(x - x_{goal}), \tag{2.10}$$




(a) Artificial Potential Field




(b) Control Barrier Function




(c) APF-based CBF

Figure 2.1: Comparison of APFs and CBFs at various values of $\rho_0$ and $\alpha$ at a control frequency of 1000 Hz

*with $K = 1$. Note that this is equivalent to the $\nabla U_{att}$ from the previous example, as the attractive force functions as a P controller on position. The control barrier function, as inspired by $\rho$ in* (2.4)*, is given by*

$$h(x) = \min_{i \in \{1,2\}} \|x - x_{Oi}\| - D_{obs}, \tag{2.11}$$

*where $\nabla h(x) = \frac{x - x_{Oi}}{\|x - x_{Oi}\|}$, for the closer obstacle $i$, and here we pick $D_{obs} = 0.5$. Note that, technically, this barrier function is non-smooth, but the methods from [58] can be employed.*

*The simulation is run for varying values of $\alpha$, and the results are shown in Figure 2.1(b). For all values of $\alpha$, the robot safely completes the mission and suffers from no oscillations. Additionally, one can see the minimally invasive behavior of* (2.9) *in that the nominal trajectories to goal are modified to a much smaller degree when compared against potential fields.*

Now, we show the main result of this section: that many potential fields are specific instances of control barrier functions. This will be demonstrated by explicitly constructing a CBF from an APF. Importantly, this transformation results in additional beneficial properties that the original controller did not benefit from. Thus, control barrier functions generalize these types of potential fields.

**Definition 4.** *Given a goal state, $x_{goal} \in \mathbb{R}^n$, an **attractive potential** is a positive definite continuously differentiable function $U_{att} : \mathbb{R}^n \to \mathbb{R}$, such that there exists $\underline{c}, \overline{c} > 0$ such that, $\forall\, x \in \mathbb{R}^n$:*

$$\underline{c}\|x - x_{goal}\|^2 \leq U_{att}(x) \leq \overline{c}\|x - x_{goal}\|^2.$$

*Given an obstacle at $x_{obst}$ and minimum distance $D_{obst} > 0$, a **repulsive potential** is a continuously differentiable positive semi-definite function $U_{rep} : \mathbb{R}^n \to \mathbb{R}$, strictly increasing, that "blows up" at the minimum distance:*

$$\begin{aligned} \textit{Positive semi-definite:} \quad & U_{rep}(x) \geq 0, \\ \textit{Strictly increasing:} \quad & \nabla U_{rep}(x) > 0 \text{ if } \|x - x_{obst}\| \leq \rho_0 \\ \textit{"Blows up" at obstacle:} \quad & \lim_{\|x - x_{obst}\| \to D_{obst}} U_{rep}(x) = \infty. \end{aligned}$$

*An **artificial potential field**: $U(x) := U_{att}(x) + U_{rep}(x)$, yields a controller:*

$$k(x) = -\nabla U(x) \qquad \Rightarrow \qquad \dot{x} = -\nabla U(x).$$

The CBF paradigm includes these types of APFs as a special case. In this case, rather than combining the attractive and repulsive potentials into a single function, we utilize the attractive potential as the desired velocity, and the repulsive potential as the control barrier function.

**Theorem 2.** *Consider an artificial potential field with repulsive potential $U_{rep}$ meeting Definition 4. The function:*

$$h(x) = \frac{1}{1 + U_{rep}(x)} - \delta, \tag{2.12}$$

*with $\delta \in (0, \delta_0)$ a small constant, is a control barrier function for the single integrator: $\dot{x} = v$. Additionally, given any feedback controller $v = k(x)$ satisfying:*

$$\dot{h}(x, k(x)) = \nabla h(x)^T k(x) \geq -\alpha(h(x))$$

*the set*

$$\mathcal{S} = \{x \in \mathbb{R}^n : h(x) \geq 0\} \subset \{x \in \mathbb{R}^n : \|x - x_{obst}\| \geq D_{obst}\}$$

*is forward invariant, i.e., safe, and asymptotically stable.*

**Remark 1.** *Note that the set $\mathcal{S}$ depends on the choice of $\delta$ with:*

$$\lim_{\delta \to 0} \mathcal{S} = \{x \in \mathbb{R}^n : \|x - x_{obst}\| \geq D_{obst}\}.$$

*Thus, the smaller the choice of $\delta$, the closer the safe set, $\mathcal{S}$, to the complement of the obstacles. Additionally, unlike potential fields, in the case when the system starts with an initial condition outside of $\mathcal{S}$, the CBF will be well defined and the system will asymptotically stabilize back to $\mathcal{S}$.*

*Proof.* Taking the gradient of $h(x)$ yields

$$\nabla h(x) = -\frac{\nabla U_{\text{rep}}(x)}{(1 + U_{\text{rep}}(x))^2}.$$

To be a control barrier function, we first require that $\nabla U_{\text{rep}} \neq 0$ when $h(x) = 0$. When $h(x) = 0$, we have that

$$\frac{1}{1 + U_{\text{rep}}(x)} - \delta = 0 \quad \Longrightarrow \quad U_{\text{rep}}(x) = \frac{1}{\delta} - 1.$$

Since $U_{\text{rep}}$ "blows up" with proximity to obstacles, one can pick $\delta_0 > 0$ such that for all $\delta \in (0, \delta_0)$ it follows that $h(x) = 0$ implies that $\|x - x_{\text{obst}}\| \leq \rho_0$. Therefore, by the strictly increasing assumption, $\nabla U_{\text{rep}} \neq 0$.

For the single integrator dynamics, the CBF requirement given in Equation (1.3) is trivially met, as $L_f h(x) = 0$ and $L_g h(x) = \nabla h(x)^T v$, so there always exists a velocity such that $-\frac{\nabla U_{\text{rep}} v}{(1+U_{\text{rep}}(x))^2} \geq -\alpha(h(x))$. Therefore, $h$ is a CBF and the remaining statements follow from Theorem 1. □

The advantage of CBFs is that they allow for controller synthesis where the attractive and repulsive potentials are combined in a pointwise optimal fashion. Specifically, using $-\nabla U_{\text{att}}(x)$ as the desired velocity, we have:

$$\begin{aligned} v^*(x) = \underset{v\in\mathbb{R}^n}{\text{argmin}} \ \ & \|v + \nabla U_{\text{att}}(x)\|^2 \\ \text{s.t.} \ \ & -\frac{\nabla U_{\text{rep}}^T v}{(1+U_{\text{rep}}(x))^2} \geq -\alpha(h(x)). \end{aligned} \tag{2.13}$$

Importantly, this controller has an explicit solution:

$$v^*(x) = -\nabla U_{\text{att}}(x) + \begin{cases} -\frac{\nabla h(x)}{\nabla h(x)^T \nabla h(x)} \Psi(x; U_{\text{att}}) & \text{if } \Psi < 0 \\ 0 & \text{if } \Psi \geq 0 \end{cases} \tag{2.14}$$

for $\Psi = \Psi(x; U_{\text{att}}) := -\nabla h(x)^T \nabla U_{\text{att}}(x) + \alpha(h(x))$.

Note the parallels between this function and the original artificial potential field, where the repulsive potential only plays a difference when within a certain radius $\rho_0$. Now, the attractive potential is used unless $-\nabla h(x)^T \nabla U_{\text{att}} + \alpha(h(x)) < 0$, in which case the CBF minimally alters the velocity inputs in order to maintain safety.

**Example 3.** *Consider the APF given in Example 1. The repulsive potential is given by* (2.3)*, which is used to make the barrier function* (2.12)*, with value* $\delta = 0.001$*. The attractive potential given by* (2.3) *with gradient* (2.6) *is used in the desired velocity controller.*

*By applying the APF-CBF QP given in* (2.13) *or the explicit solution* (2.14)*, the path shown in 2.1(c) is obtained for tuning parameters* $K_{rep} = K_{att} = \alpha = 1$*. One can see improved performance for the APF-CBF QP obtained from the APF when compared against the original APF, wherein the APF-CBF QP gets closer to the obstacles with fewer oscillations. Moreover, when the robot has passed the obstacle and is moving towards the goal, the APF-CBF converges more quickly to the desired path.*

It is important to note that the connection between APFs and CBFs allows for potential fields to be generalized to a nonlinear setting with ease. In particular, since

CBFs are defined for general nonlinear control systems, as in (1.1), we can use the instantiation of APFs as CBFs to easy extend APFs to a nonlinear setting.

**Proposition 1.** *Consider a nonlinear control system of the form:* $\dot{x} = f(x) + g(x)u$. *Assume the existence of a potential field as given in Definition 4 with the associated safety constraint, h, given in* (2.12)*. If the repulsive potential,* $U_{rep}$*, satisfies the CBF condition*

$$\nabla U_{rep}(x)^T g(x) = 0 \implies -\frac{\nabla U_{rep}(x)^T f(x)}{(1 + U_{rep}(x))^2} \geq -\alpha(h(x)),$$

*for some extended class* $\mathcal{K}$ *function* $\alpha$ *then the controller*

$$\begin{aligned} u^*(x) = \underset{v \in \mathbb{R}^n}{\operatorname{argmin}} \ & \|u + \nabla U_{att}(x)\|^2 \\ \text{s.t.} \ & -\frac{\nabla U_{rep}^T(f(x) + g(x)u)}{(1 + U_{rep}(x))^2} \geq -\alpha(h(x)), \end{aligned} \tag{2.15}$$

*renders the set* $\{x \in \mathbb{R}^n : \|x - x_{obst}\| \geq D_{obst}\}$ *forward invariant, i.e., a controller that ensures safety.*

*Proof.* The CBF condition is simply: $L_g h(x) = 0$ implies that $L_f h(x) \geq -\alpha(h(x))$. One can verify that this implies that (1.5) is well defined, thus (1.3) is satisfied and $h$ is a CBF. The result then follows by combining Theorem 1 with Theorem 2. □

To illustrate the issue with utilizing model-free collision avoidance on systems with non-trivial dynamics, we will compare the performance of the same artificial potential field and control barrier function described in Examples 1 and 2, but applied to a double integrator:

$$\dot{x} = v, \quad \dot{v} = u \tag{2.16}$$

The velocity outputs of the safety filters, denoted $v^*(x)$, are now tracked by the velocity-based controller:

$$u(x, v) = -K(v - v^*(x)). \tag{2.17}$$

The velocity-based controller is implemented for $v^*$ obtained from APFs and CBFs (per Examples 1 and 2). For the APF, the $\rho_0$ values of 1 and 2, the APF is able to keep the system safe while reaching the goal. However, large oscillations occur while approaching the first obstacle. The oscillations are not present with

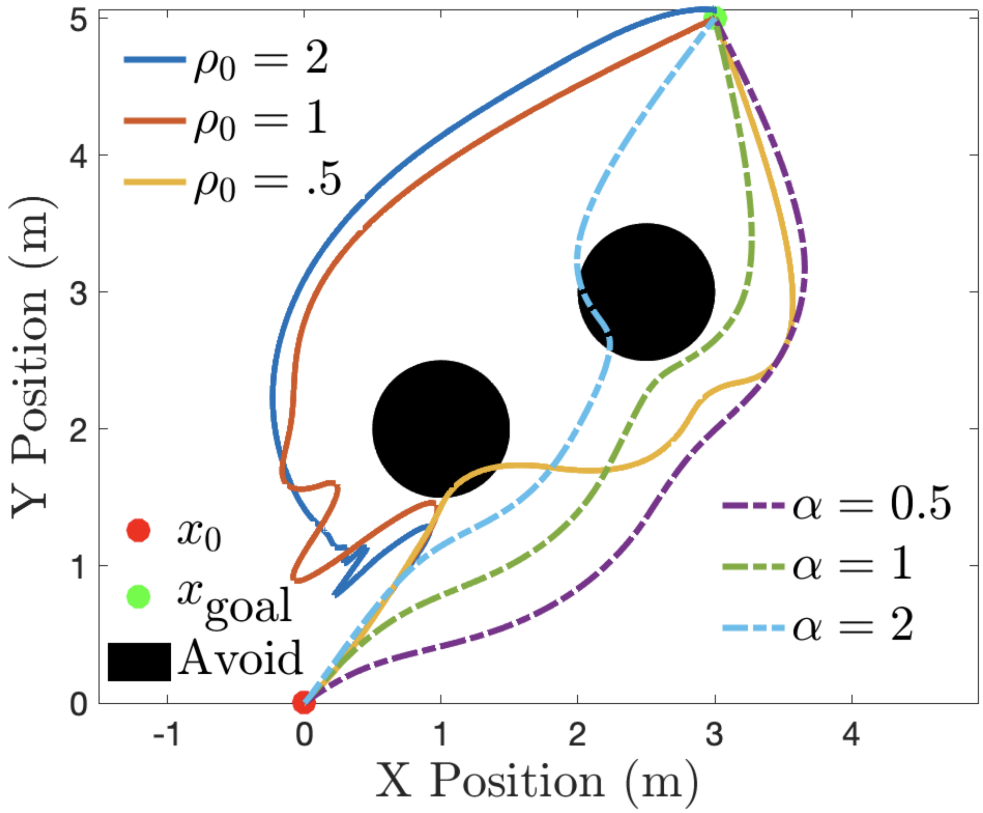



Figure 2.2: Potential Fields and CBFs on a double integrator.

$\rho_0$ equal to 0.5, but safety is no longer maintained. Eliminating these oscillations and maintaining safety is possible, but would require additional tuning. For CBFs, safety could be trivially guaranteed by utilizing the double integrator dynamics in the CBF-QP, but we instead utilize the velocity-based controller from Example 2. Safety is maintained for $\alpha$ values of 0.5 and 1, but it is not maintained for $\alpha = 2$. This biggest difference between the performance of the CBF and the APF is the lack of oscillations when approaching the obstacles, with CBFs resulting in smoother controllers. This is a trend that will be seen in simulation and experimentally on the quadrotor.

### 2.1.1 Applications: APFs vs CBFs on quadrotor

To provide a more realistic comparison of APFs and CBFs realized as velocity-based controllers (2.17), we consider their application to quadrotors in the context of obstacle avoidance. In particular, we compare the APF and CBF velocity-based controllers in a high-fidelity simulation environment based on the physical hardware

that will be detailed after. The dynamics and low-level control are provided by the ArduPilot SITL simulator, and the velocity commands are produced and filtered in ROS nodes using simulated LIDAR sensor data from Gazebo.

In the context of the APF velocity-based controller, we will utilize the general form given in Example 1, with the same attractive potential as given in (2.2). The repulsive potential is replaced with a more modern potential field formulation that has been tuned for a quadrotor in simulation—this was done to avoid the repulsive potential from becoming ill-defined when realized in practice. In particular, the repulsive potential is taken from [54], yielding the repulsive force

$$F_{\text{rep}} = (x - x_{Oi})K_{\text{rep}} \exp\left(-\frac{\rho^2}{\rho_0}\right) \tag{2.18}$$

with $\rho = \|x - x_{Oi}\|$. The values of $K_{\text{rep}}$ and $\rho_0$ were tuned until oscillations vanished in practical cases, and safety was achieved, but optimized such that they do not affect flight when collision is unlikely.

For the CBF-based velocity controller, we use a formulation identical to Example 2. In particular, the barrier function is given as in (2.11), where $x_{Oi}$ is the $i^{\text{th}}$ point of the simulated laser scan, and $D_{\text{obs}} = 0.3$ represents the minimum distance that the drone must maintain from the obstacles. Utilizing this with the single integrator dynamics results in the control law in (2.9) which is passed to a low-level velocity tracking controller. The value of $\alpha$ kept to 1, to ensure that no tuning is performed to improve the results.

For each experiment, the quadrotor is given a waypoint 5m ahead in the x-direction. The five tests are described as follows, in order of difficulty for the collision avoidance algorithms: (1) in between the quadrotor and the goal, two obstacles are placed that are offset from the center of the path, but close enough to effect the drone. (2) A single obstacle is placed such that the edge of the obstacle aligns with the center of the drone. This is done to ensure that the drone is able to find a path around it, but to strongly obstruct the drone. (3) Two obstacles are placed with edges 1 m apart, to mimic a 1 m wide doorway, and the drone has to fly through this to reach the goal. (4) The doorway from the previous test is reduced to 0.7 m in width. (5) In between the starting position and the goal is a large wall that the drone is not able to pass. This is to test the oscillations that may occur when running directly at an obstacle that is in front of the goal.

Each of the five tests are run for both the artificial potential field, as well as the control barrier function. The setups and paths are shown in Figure 2.3, along with

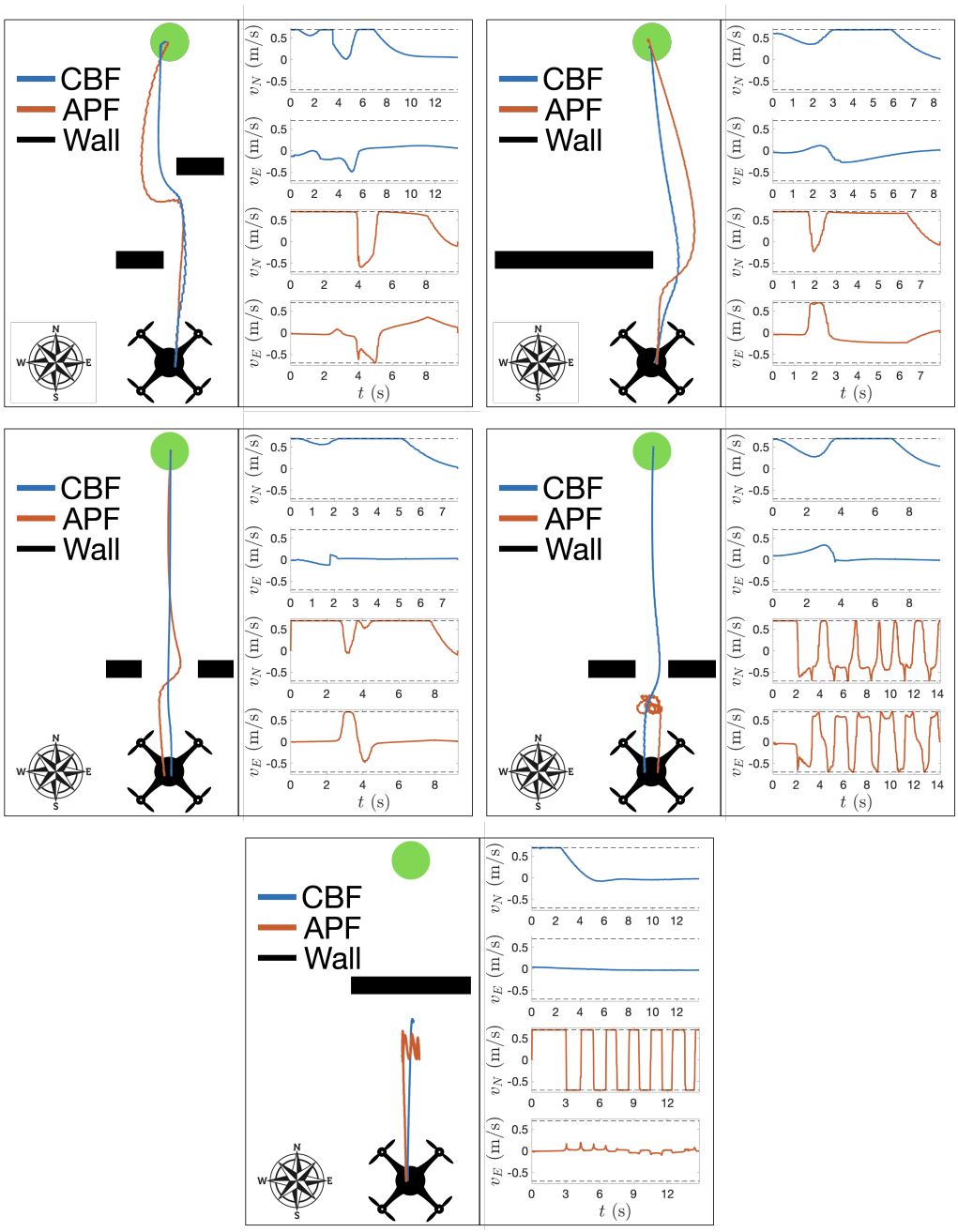



Figure 2.3: Simulation results for the quadrotor with APFs and CBFs for the five scenarios considered in this section.

the velocities. Oscillations do not occur for the CBF velocity-based controller, and only occur for the APF based controller situations where the drone is unable to get to the goal, e.g., the wall and the 0.7 m doorway. In all cases, the CBF is able to get closer to the obstacles while staying safe due to its pointwise optimally.

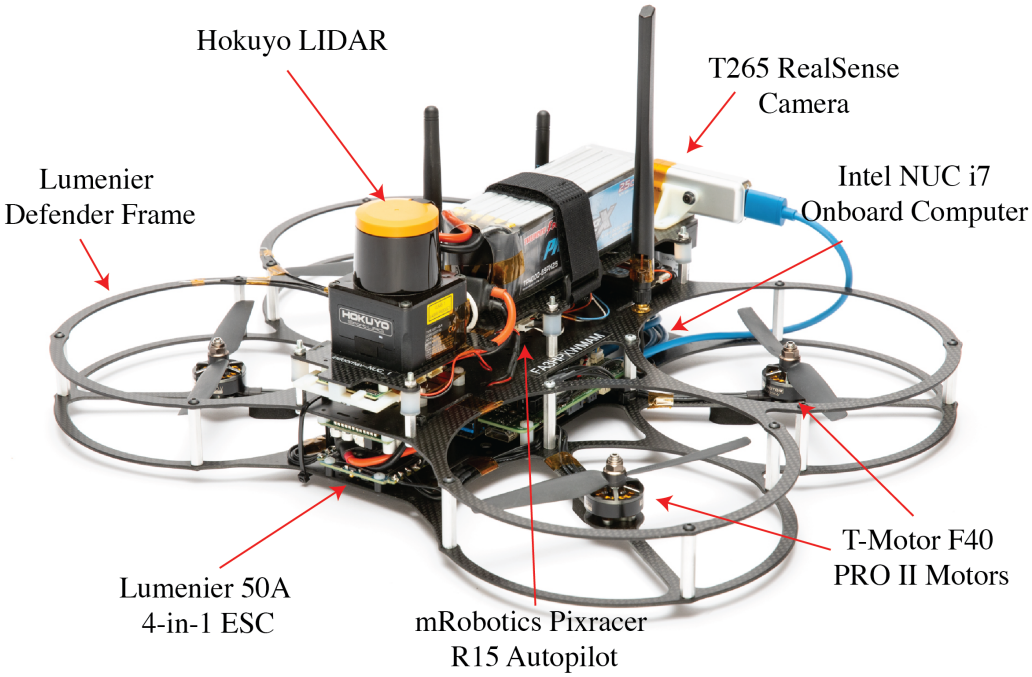



Figure 2.4: The quadrotor used in the experiments.

**Hardware setup** The quadrotor used for experiments is shown in Figure 2.4. It consists of a Lumenier Defender frame, four T-Motor F40 PRO II 1600 KV brushless motors, a Lumenier 50A 4-in-1 ESC, a mRobotics Pixracer R15 autopilot, a T265 RealSense camera, a Intel NUC i7 onboard computer, and a Hokuyo UST-10LX LIDAR. The Hokuyo UST-10LX 2D LIDAR gives 1080 points in front of the quadrotor in a 270° field of view along the XY plane. Google's Cartographer SLAM package was used with the Hokuyo LIDAR and the RealSense camera for localization. Additionally, the Hokuyo LIDAR is used for obstacle detection and avoidance. An Intel NUC i7 onboard computer is used to run the ROS nodes that perform the localization and collision avoidance. The high-level velocity commands are passed from the onboard computer to the Pixracer flight controller. The flight controller utilizes a cascading PID control structure of velocity, acceleration, attitude, and angular rate.

The same five tests described in the previous section (see Fig. 2.3) were implemented on the hardware, and the results are shown in Fig. 2.5. The only difference in the setup was that the drone is now commanded to yaw 180° and return along the same path, in order to maximize the amount of data for the analysis. While the hardware results are similar to simulation for CBFs, the artificial potential field suffers from

significantly more oscillations than in the high-fidelity simulation environment. This suggests that the CBF implementation is more robust to model uncertainty and noise, as the APF would need to be tuned again to eliminate oscillations due to the differences between the simulation and reality. Finally, APFs fail to reach the goal in the case of the narrow door, while the CBFs succeeds. Thus, the CBFs outperformed APFs on hardware.

## 2.2 Energy-based CBFs

Kinematic control provides a powerful method for achieving desired behaviors on a large class of robotic systems [136, 164, 18]. Ensuring safety for these kinematic systems is widely researched area. Artificial potential field methods were formulated as a way to reach goal positions while avoiding obstacles utilizing an attractive force from the goal and a repulsive force from the obstacles [81]. In [48], the authors improve upon this idea by constructing the problem as a quadratic program (QP), where the objective is to track the desired goal subject to geometric constraints on the velocities to prevent collisions. While this work is effective in practice, and has been extended to multi-objective task structures [75], it can be made more general and more formal through control barrier functions (CBFs).

Recently, energy-based reciprocal control barrier functions were introduced [84] as a means to provide robust safety guarantees for fully-actuated robotic platforms with model uncertainty. This was done by utilizing bounds on the inertia and Coriolis-centrifugal matrices, as well as the gravity vector, and providing safety guarantees for the worst-case scenario. While the resulting QP formulation yielded robustness in safety, it does not have well-defined behavior on the boundary of the set and outside of it, making it difficult to implement in practice.

In this section, an alternative formulation for the energy-based CBFs is introduced for zeroing control barrier functions, which are well defined on the boundary and exterior of the set. Using this formulation, we modify the traditional torque-based formulation into a kinematic control problem, and showcase several simplifications that can be made to reduce model dependence. The resulting formulation allows for formal safety guarantees at the dynamical system level, while allowing for simple implementation with kinematic controllers. This analysis is then extended to the class of underactuated robotic systems. The results are demonstrated in a 6-DOF manipulator and a cart-pole system wherein different levels of uncertainties are incorporated and safety-critical kinematic control laws are applied.

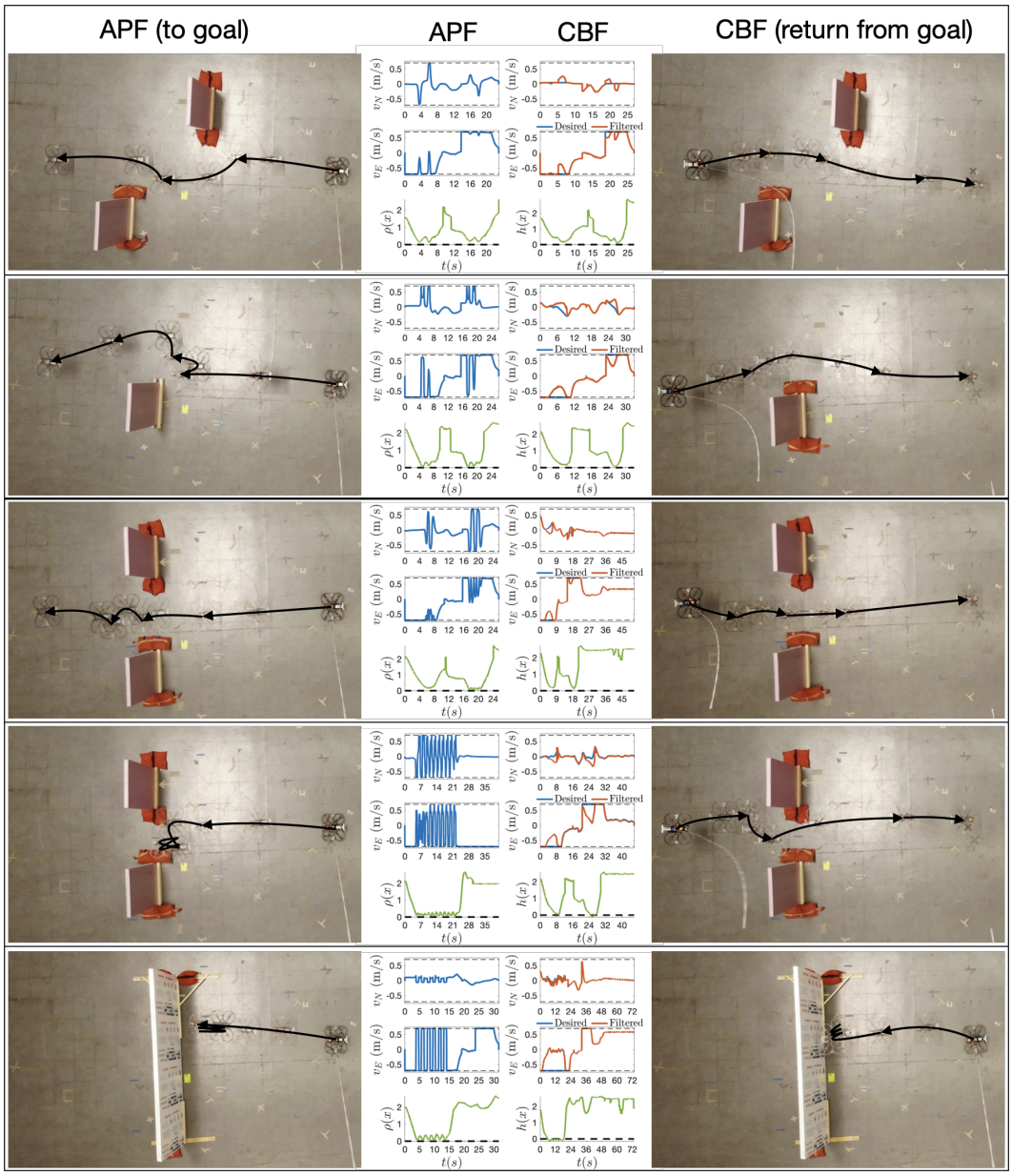



Figure 2.5: Hardware results for the control barrier function and artificial potential field. Note that the APF showcases the drone as it goes to the goal, while the CBF is pictured returning to the start position. Due to symmetry of the setup, these trips almost identical for each method, and the plots contain data for the round trip. The video found here [156] shows both directions for each of the experiments from three different angles. The first two columns show the velocities in the north and east directions, and the final column shows the value of $\rho(x) - 0.3$ for the APF on the left, and $h(x)$ for the barrier on the right.

Before beginning the kinematic formulation, we propose the following theorem for the explicit solution of the CBF-QP (CBF-QP).

**Lemma 1.** *Let $h$ be a control barrier function for the control system* (1.1) *and assume that $U = \mathbb{R}^m$. Then the explicit solution to the QP* (CBF-QP) *is given by:*

$$u^*(x,t) = u_{des}(x,t) + u_{safe}(x,t), \tag{2.19}$$

*where $u_{safe}$ minimally modifies $u_{des}$ depending on if the nominal controller keeps the system safe, i.e., the sign of $\Psi(x,t;u_{des}) := \dot{h}(x,u_{des}(x,t)) + \alpha(h(x))$, according to:*

$$u_{safe}(x,t) = \begin{cases} -\frac{L_g h(x)^T}{L_g h(x) L_g h(x)^T}\Psi(x,t;u_{des}) & \text{if } \Psi(x,t;u_{des}) < 0 \\ 0 & \text{if } \Psi(x,t;u_{des}) \geq 0 \end{cases}. \tag{2.20}$$

We are interested in kinematic mappings of the form $x = y(q)$ where $q \in Q \subset \mathbb{R}^k$, $x \in \mathcal{D} \subset \mathbb{R}^n$ and thus $y : Q \to \mathcal{D}$. Here, we assume that $k \geq n$, i.e., that there are more degrees of freedom than tasks. Here $x$ is the vector of "outputs" or "task" variables, i.e., a vector of elements which we wish to control, and $q$ is a vector consisting of the systems configuration, e.g., angles of the robotic system. The evolution of the task variables is therefore given by:

$$\dot{x} = J_y(q)\dot{q}. \tag{2.21}$$

In kinematic control, we view $\dot{q}$ as the input to the system. That is, we have the dynamics model:

$$\dot{q} = v. \tag{2.22}$$

With this, we wish to determine a feedback control law: $\dot{q} = K(q,t)$ that achieves the desired properties.

Suppose that we have a desired trajectory $x_d(t)$ for the task vector. The goal is to track this trajectory, i.e., for $e(t) = x(t) - x_d(t) \to 0$ with $x(t)$ satisfying (2.21). Differentiating this yields:

$$\dot{e} = J_y(q)\dot{q} - \dot{x}_d(t).$$

Therefore, for $\gamma > 0$, if we choose $\dot{q}$ such that $J_y(q)\dot{q} = \dot{x}_d(t) - \gamma e$, we have $\dot{e} = -\gamma e \implies e(t) = \exp(-\gamma t)e(0)$. As a result, if we wish to track a trajectory, we can pick:

$$\dot{q}(x,t) = J_y(q)^\dagger \left(\dot{x}_d(t) - \lambda e\right), \tag{2.23}$$

with $J_y(q)^\dagger = J_y(q)^T(J_y(q)J_y(q)^T)^{-1}$, the Moore-Penrose (right) pseudoinverse, assumed to be well defined.

Equipped with $\dot{q}$, which will now serve as the desired (potentially unsafe) input $\dot{q}_{\text{des}}$, we can now impose safety. We have the following.

**Lemma 2.** *Consider a kinematic safety constraint $h : Q \subset \mathbb{R}^k \to \mathbb{R}$ and the corresponding safe set $\mathcal{S} = \{q \in Q \ : \ h(q) \geq 0\}$ defined as the 0-superlevel set of $h$. If $J_h(q) \neq 0$, then the following velocity based controller:*

$$\begin{aligned} \dot{q}^*(q,t) = \underset{\dot{q}\in\mathbb{R}^k}{\text{argmin}} \ & \|\dot{q} - J_y(q)^\dagger \left(\dot{x}_d(t) - \lambda(y(q) - x_d(t))\right)\|^2 \\ \text{s.t.} \ & \dot{h}(q,\dot{q}) = J_h(q)\dot{q} \geq -\alpha(h(q)), \end{aligned} \tag{2.24}$$

*ensures safety, i.e., $\mathcal{S}$ is forward invariant. Moreover, this has a closed form solution given by*

$$\dot{q}^*(x,t) = \dot{q}_{des}(q,t) + \begin{cases} -J_h(x)^\dagger \Psi(q,t;\dot{q}_{des}) & \text{if } \Psi(q,t;\dot{q}_{des}) < 0 \\ 0 & \text{if } \Psi(q,t;\dot{q}_{des}) \geq 0 \end{cases}, \tag{2.25}$$

*where $\Psi(x,t;\dot{q}_{des}) = J_h(q)\dot{q}_{des}(q,t) + \alpha(h(q))$.*

Therefore, the controller (2.25) utilizes $\dot{q}_{\text{des}}$ whenever it is safe, i.e., when $\Psi(q,t;\dot{q}_{\text{des}}) \geq 0$. Conversely, in the case when $\dot{q}_{\text{des}}$ is unsafe the controller takes over and enforces $\dot{h} = J_h(q)\dot{q}^*(q,t) = -\alpha(h)$ until $\dot{q}_{\text{des}}$ is safe again.

**Example 4** (**Manipulator Obstacle Avoidance**). *Consider a 6-DOF industrial manipulator attempting to track a desired trajectory $x_d(t)$ using the desired velocity given in* (2.23) *with its end-effector. Note that CBFs have been successfully applied to robot manipulators in [45, 120, 91] via kinematic control. Suppose that the manipulator needs to complete this trajectory while avoiding an obstacle located at $(x_0, y_0, z_0)$. Thus, in the set $\mathcal{S} = \{q \mid h(q) \geq 0\}$, the end-effector must be at least a distance $d$ from the obstacle. A control barrier function representing this safety constraint is*

$$h(x) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - d^2. \tag{2.26}$$

*By substituting this into* (2.24) *or* (2.25)*, we obtain the results shown in Figures 2.6 and 2.7. Since this CBF does not take into account the system dynamics or the tracking ability of the low-level controller, safety is not guaranteed, but it can be achieved by proper choice of $\alpha$. In this case, with scalar multiple $\alpha \in [0.5, 1]$, the obstacle is avoided, but not for $\alpha \in [2, 3]$.*

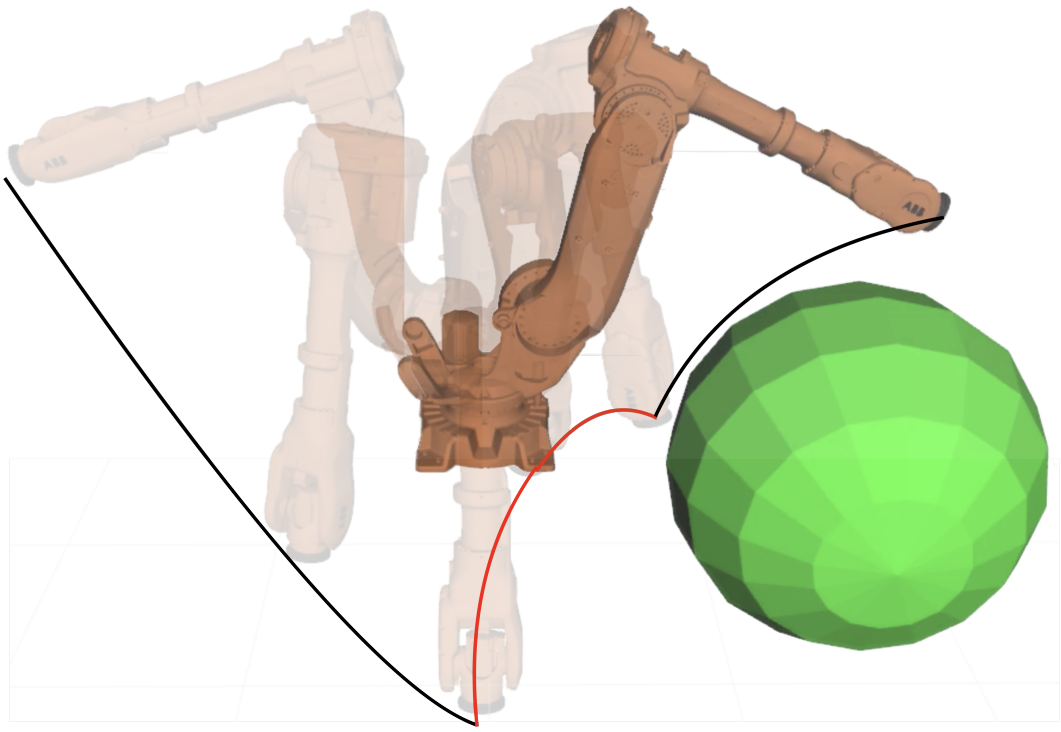


Figure 2.6: A 6-DOF manipulator safely avoiding an obstacle with energy-based control barrier function. The CBF intervention is shown in red.

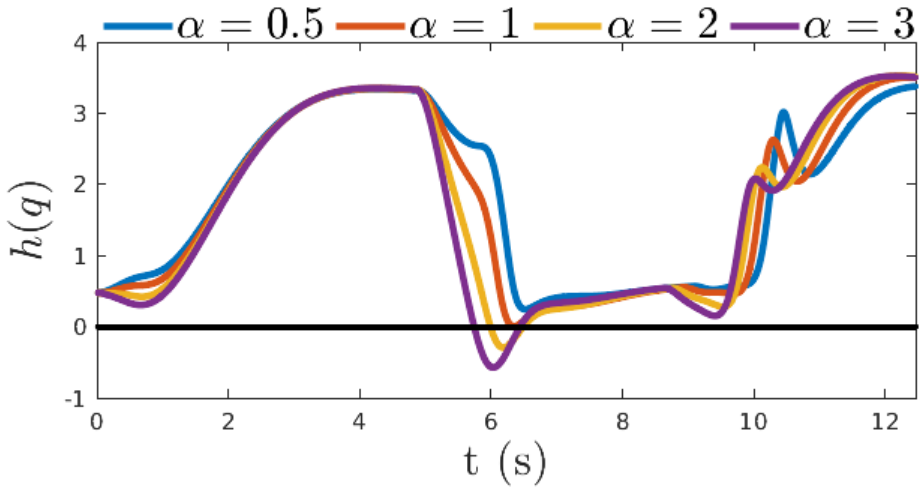



Figure 2.7: Velocity-based kinematic barrier function on the 6 DOF manipulator. Safety depends on choice of $\alpha$.

### 2.2.1 Guarantees on dynamics

We now wish to establish the main result of this section: guarantees safety for the *dynamics* of a robotic system, not just the kinematics. To do this, we first introduce an alternative formulation of the energy-based CBFs shown in [84] for robotic systems.

We consider Euler-Lagrangian dynamics of the form:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu, \tag{2.27}$$

where $B \in \mathbb{R}^{k\times m}$ is the actuation matrix, $D, C, G$ are the inertia, Coriolis-centrifugal and gravity matrices respectively of appropriate dimensions. We assume $m \leq k$, wherein $m = k$ with $B$ invertible corresponds to full actuation. From the equations of motion, we can obtain a control system of the form (1.1). We will first discuss the fully actuated case, and the underactuated case will be discussed after.

We begin by formulating a safety-critical controller for fully actuated robotic systems given kinematic safety constraints—thus bridging the gap from kinematic to dynamics. This will be achieved via a "dynamically consistent" extension to the desired safe set. This is similar to the extensions shown in [16], [109] for higher relative degree systems, but leverages the kinetic energy of the system. Specifically, to dynamically extend the CBF, we note that the inertia matrix, $D(q)$ is a symmetric positive definite matrix, $D(q) = D(q)^T \succ 0$, and thus:

$$\lambda_{\min}(D(q))\|q\|^2 \leq q^T D(q) q \leq \lambda_{\max}(D(q))\|q\|^2,$$

where $\lambda_{\min}$, $\lambda_{\max}$ are the min and max eigenvalues (which are dependent on $q$) of $D(q)$ which are necessarily positive due to the positive definite nature of $D(q)$.

**Definition 5.** *Given a kinematic safety constraint expressed as a function $h : Q \subset \mathbb{R}^k \to \mathbb{R}$ only dependent on $q$, and the corresponding safe set: $\mathcal{S} = \{(q,\dot{q}) \in Q \times \mathbb{R}^k \ : \ h(q) \geq 0\}$, the associated **energy-based safety constraint** is defined as:*

$$h_D(q,\dot{q}) := -\frac{1}{2}\dot{q}^T D(q)\dot{q} + \alpha_e h(q) \geq 0 \tag{2.28}$$

*with $\alpha_e > 0$. The corresponding **energy-based safe set** is:*

$$\mathcal{S}_D := \{(q,\dot{q}) \in Q \times \mathbb{R}^k \ : \ h_D(q,\dot{q}) \geq 0\}. \tag{2.29}$$

This construction is similar to the augmentation of kinetic energy in [84] for reciprocal control barrier functions. While the reciprocal formulation has the advantage

of having no added conservatism, due to the set remaining unchanged, it does not have well-defined behavior on the boundary of the set and outside of it, making it less popular for implementation. In fact, we now will show that the energy based constraint in Definition 5 is a valid (zeroing) control barrier function (CBF), thereby allowing for a new class of QPs that guarantee safety. First, we establish the relationship between $\mathcal{S}_D$ and $\mathcal{S}$.

**Proposition 2.** *Consider a kinematic safety constraint, $h : Q \subset \mathbb{R}^k \to \mathbb{R}$, with corresponding safe set $\mathcal{S}$, and the associated energy-based safety constraint, $h_D$, as given in Definition 5 with corresponding safe set $\mathcal{S}_D$. Then*

$$\text{(i) } \mathcal{S}_D \subset \mathcal{S}, \qquad \text{(ii) } \text{Int}(\mathcal{S}) \subset \lim_{\alpha_e \to \infty} \mathcal{S}_D \subset \mathcal{S}. \tag{2.30}$$

*Proof.* To establish (i), we simply note that

$$\mathcal{S}_D \subset \{(q, \dot{q}) \in Q \times \mathbb{R}^k \ : \ h(q) \geq \frac{1}{2} \frac{\lambda_{\min}(D(q))}{\alpha_e} \|\dot{q}\|^2 \geq 0\} \subset \mathcal{S}.$$

To establish (ii), we first note that

$$S_D(\alpha_e) = \{(q, \dot{q}) \in Q \times \mathbb{R}^k \ : \ h(q) \geq \frac{\frac{1}{2}\dot{q}^T D(q)\dot{q}}{\alpha_e}\},$$

where here we made the dependence of $S_D$ on $\alpha_e$ explicit. Consider an increasing sequence $\alpha_e^i$ where $i \in \mathbb{N}$ and $\lim_{i\to\infty} \alpha_e^i \to \infty$. This results is a nondecreasing sequence of sets: $\{\mathcal{S}_D(\alpha_e^i)\}_{i=1}^{\infty}$:

$$\alpha_e^i < \alpha_e^{i+1} \ \Rightarrow \ \frac{\frac{1}{2}\dot{q}^T D(q)\dot{q}}{\alpha_e^i} > \frac{\frac{1}{2}\dot{q}^T D(q)\dot{q}}{\alpha_e^{i+1}} \ \Rightarrow \ \mathcal{S}_D(\alpha_e^i) \subset \mathcal{S}_D(\alpha_e^{i+1}).$$

As a result:

$$\lim_{i\to\infty} \frac{\frac{1}{2}\dot{q}^T D(q)\dot{q}}{\alpha_e^i} = 0 \ \Rightarrow \ \lim_{i\to\infty} \mathcal{S}_D(\alpha_e^i) = \bigcup_{i\in\mathbb{N}} \mathcal{S}_D(\alpha_e^i) \supset \text{Int}(\mathcal{S}),$$

and $\mathcal{S}_D(\alpha_e^i) \subset \mathcal{S}$ for all $i \in \mathbb{N}$. □

We now have the necessary constructions to present the main result of this section—a largely model independent safety-critical controller that ensures the forward invariance of $\mathcal{S}_D$ and, thus, $\mathcal{S}$ in the limit for $\alpha_e$ sufficiently large. We will establish this by showing that $h_D$ is a valid CBF and that $\dot{h}_D$ only depends on the kinematics, the gravity vector $G(q)$, and the inertial matrix $D(q)$. This makes the controller more robust to uncertainty in the dynamics than full model based controllers—which would require knowledge of the Coriolis-centrifugal matrix, $C(q, \dot{q})$.

**Theorem 3.** *Consider a robotic system* (2.27)*, assumed to be fully actuated with $B$ invertible, and a kinematic safety constraint $h : Q \to \mathbb{R}$ with corresponding safe set $\mathcal{S} = \{(q, \dot{q}) \in Q \times \mathbb{R}^k : h(q) \geq 0\}$. Let $h_D$ be the energy based constraint defined as in* (2.28) *with corresponding safe set $\mathcal{S}_D$ as given in* (2.29)*. Then $h_D$ is a control barrier function on $\mathcal{S}_D$ and given a desired controller $u_{des}(x, t)$, the following controller for all $(q, \dot{q}) \in \mathcal{S}_D$:*

$$\begin{aligned} u^*(q, \dot{q}, t) = \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \ & \|u - u_{des}(q, \dot{q}, t)\|^2 \\ \text{s.t.} \ & \underbrace{-\dot{q}^T B u + G(q)^T \dot{q} + \alpha_e J_h(q)\dot{q}}_{\dot{h}_D(q,\dot{q},u)} \geq -\alpha(h_D(q, \dot{q})), \end{aligned} \tag{2.31}$$

*guarantees forward invariance of $\mathcal{S}_D$, i.e., safety of $\mathcal{S}_D$. Additionally, it has a closed form solution:*

$$u^*(x, t) = u_{des}(q, \dot{q}, t) + \begin{cases} \frac{B^T \dot{q}}{\|B^T \dot{q}\|^2} \Psi(x, t; u_{des}) & \text{if } \Psi(x, t; u_{des}) < 0 \\ 0 & \text{if } \Psi(x, t; u_{des}) \geq 0 \end{cases}, \tag{2.32}$$

*where*

$$\Psi(x, t; u_{des}) := \dot{q}^T (\alpha_e J_h(q)^T + G(q) - B u_{des}(x, t)) + \alpha(h_D(q, \dot{q})).$$

It is interesting to note that $h_D$ is a CBF on $\mathcal{S}_D$ without requiring that $h$ has relative degree 1, i.e., one need not require that $J_h(q) \neq 0$ (except on $\partial\mathcal{S}$) as in Lemma 2. This reinforces the idea that these energy-based control barrier functions are natural extensions for relative-degree 2 systems.

*Proof of Theorem 3.* Differentiating $h_D$ along solutions yields (and suppressing the dependence on $q$ and $\dot{q}$):

$$\begin{aligned} \dot{h}_D &= -\dot{q}^T D \ddot{q} - \frac{1}{2}\dot{q}^T \dot{D} \dot{q} + \alpha_e J_h \dot{q} \\ &= \dot{q}^T (C\dot{q} + G - Bu) - \frac{1}{2}\dot{q}^T \dot{D} \dot{q} + \alpha_e J_h \dot{q} \\ &= \frac{1}{2}\dot{q}^T \left(-\dot{D} + 2C\right) \dot{q} - \dot{q}^T B u + G^T \dot{q} + \alpha_e J_h \dot{q} \\ &= -\dot{q}^T B u + G^T \dot{q} + \alpha_e J_h \dot{q}, \end{aligned} \tag{2.33}$$

where the last equality follows from the fact that $\dot{D} - 2C$ is skew symmetric (see [106, Lemma 4.2]). To establish that $h_D$ is a CBF, we need only show that (2.31) has a solution since the inequality constraint in (2.31) implies that (1.3) is satisfied

in Definition 3. As a result of Lemma 1, the solution to (2.31) is given by (1.4) . Note that

$$L_f h_D(q,\dot{q}) = (\alpha_e J_h(q) + G(q)^T)\dot{q}, \quad L_g h_D(q,\dot{q}) = -\dot{q}^T B.$$

Since (1.4) has a $L_g h L_g h^T$ term in the denominator, to show that (1.4) is well defined, we need to establish that:

$$L_g h_D(q,\dot{q}) = -\dot{q}^T B = 0 \quad \Rightarrow \quad L_f h_D(q,\dot{q}) + \alpha(h_D(q,\dot{q})) \geq 0.$$

Yet $\dot{q}^T B = 0$ implies that $\dot{q}^T = 0$ since $B$ is invertible and therefore $L_f h_D(q,\dot{q}) = 0$ and since $(q,\dot{q}) \in \mathcal{S}_D$ it follows that $h_D(q,\dot{q}) \geq 0$ and hence $\alpha(h_D(q,\dot{q})) \geq 0$ implying that (1.4) is well defined and thus $h_D$ is a CBF. Finally, the forward invariance of $\mathcal{S}_D$ follows from the results of Lemma 1 and Theorem 1. □

Having established Theorem 3, the following corollary demonstrates how to further reduce model dependence.

**Corollary 1.** *Under the conditions of Theorem 3, if there exists a $c_u > 0$ such that $c_u \geq \frac{1}{2}\lambda_{\max}(D(q))$ then replacing the safety constraint* (2.31) *in the safety-critical QP with:*

$$\underbrace{-\dot{q}^T B u + G(q)^T \dot{q} + \alpha_e J_h(q)\dot{q}}_{\dot{h}_D(q,\dot{q},u)} \geq -\alpha(-c_u \|\dot{q}\|^2 + \alpha_e h(q)), \tag{2.34}$$

*implies safety of $\mathcal{S}_D$. Moreover, if in addition $\|G(q)\| \leq c_u$, for a large enough $c_u > 0$ (perhaps larger than previously determined), then the constraint* (2.31) *can be replaced by:*

$$\alpha_e J_h(q)\dot{q} - \dot{q}^T B u - c_u|\dot{q}| \geq -\alpha\left(-c_u \|\dot{q}\|^2 + \alpha_e h(q)\right). \tag{2.35}$$

*wherein safety of $\mathcal{S}_D$ is guaranteed.*

*Proof.* It can be verified that $-\alpha(-c_u \|\dot{q}\|^2 + \alpha_e h(q)) \geq -\alpha(-\frac{1}{2}\lambda_{\max}(D(q)) \|\dot{q}\|^2 + \alpha_e h(q)) \geq -\alpha(h_D(q,\dot{q}))$, which means that (2.34) $\Longrightarrow$ (2.31). The second inequality, (2.35), follows from the bound on the gravity vector $G$. □

The goal is to now connect the previous constructions with the kinematic controllers defined previously. Often, controllers can only be implemented as desired position and velocity commands that are passed to embedded level PD controllers. Moreover,

minimizing the difference between the desired and the safe robot velocities often leads to more desirable behaviors with the lower-level commands, which affect the system in much more complex ways. As such, we consider a controller:

$$u = -K_{\text{vel}}(\dot{q} - \dot{q}_d^*(q, \dot{q}, t)) \tag{2.36}$$

where $\dot{q}_d^*(q,t)$ is a desired velocity signal that enforces safety while trying to achieve tracking as in the case of Lemma 2 wherein we have a desired velocity based tracking controller: $\dot{q}_{\text{des}}(q,t) := J_y(q)^\dagger\,(\dot{x}_d(t) - \lambda(y(q) - x_d(t)))$ for $\lambda > 0$. The following is a result of the direct application of Theorem 3 in the context of the controller (2.36).

**Theorem 4.** *Consider a robotic system* (2.27)*, and assume it is fully actuated. Given a kinematic safety constraint $h : Q \to \mathbb{R}$ and the associated dynamically consistent extended CBF $h_D : Q \times \mathbb{R} \to \mathbb{R}$ as given in* (2.28) *with associated safe set $\mathcal{S}_D$, along with a desired trajectory $x_d(t)$ in the task space $x = y(q)$. The D controller* (2.36) *with $K_{vel} \succ 0$ and the following QP:*

$$\begin{aligned} \dot{q}_d^* = \underset{\dot{q}_d \in \mathbb{R}^n}{\text{argmin}} \; & \|\dot{q}_d - \overbrace{J_y^\dagger\,(\dot{x}_d - \lambda(y - x_d))}^{\dot{q}_{des}(q,t)}\|^2 \\ \text{s.t.} \; & \underbrace{\alpha_e J_h \dot{q} + \dot{q}^T B K_{vel} \dot{q} - \dot{q}^T B K_{vel} \dot{q}_d + G^T \dot{q}}_{\dot{h}_D(q,\dot{q},\dot{q}_d)} \geq -\alpha(h_D), \end{aligned} \tag{2.37}$$

*guarantees forward invariance, i.e., safety, of $\mathcal{S}_D$. Moreover, it has a closed form solution:*

$$\dot{q}_d^* = \dot{q}_{des} + \begin{cases} \frac{K_{vel}^T B^T \dot{q}}{\|K_{vel}^T B^T \dot{q}\|^2} \Psi(q, \dot{q}, t; q_{des}) & \text{if } \Psi(q, \dot{q}, t; q_{des}) < 0 \\ 0 & \text{if } \Psi(q, \dot{q}, t; q_{des}) \geq 0 \end{cases}, \tag{2.38}$$

*where*

$$\Psi(q, \dot{q}, t; \dot{q}_{des}) := \dot{q}^T(\alpha_e J_h^T + B K_{vel} \dot{q} - B K_{vel} \dot{q}_{des} + G) + \alpha(h_D).$$

Proof of Theorem 4 is omitted as it is a straightforward extension of Theorem 3.

It may be the case, as with industrial actuators, that $K_{\text{vel}}$ is not known. In that case, it can typically be determined from experimental data. Formally, one can guarantee safety by utilizing adaptive control barrier functions [147]. Similar to Remark 1, we can reformulate the constraints to eliminate the $D$ and $G$ matrices to yield robust QPs.

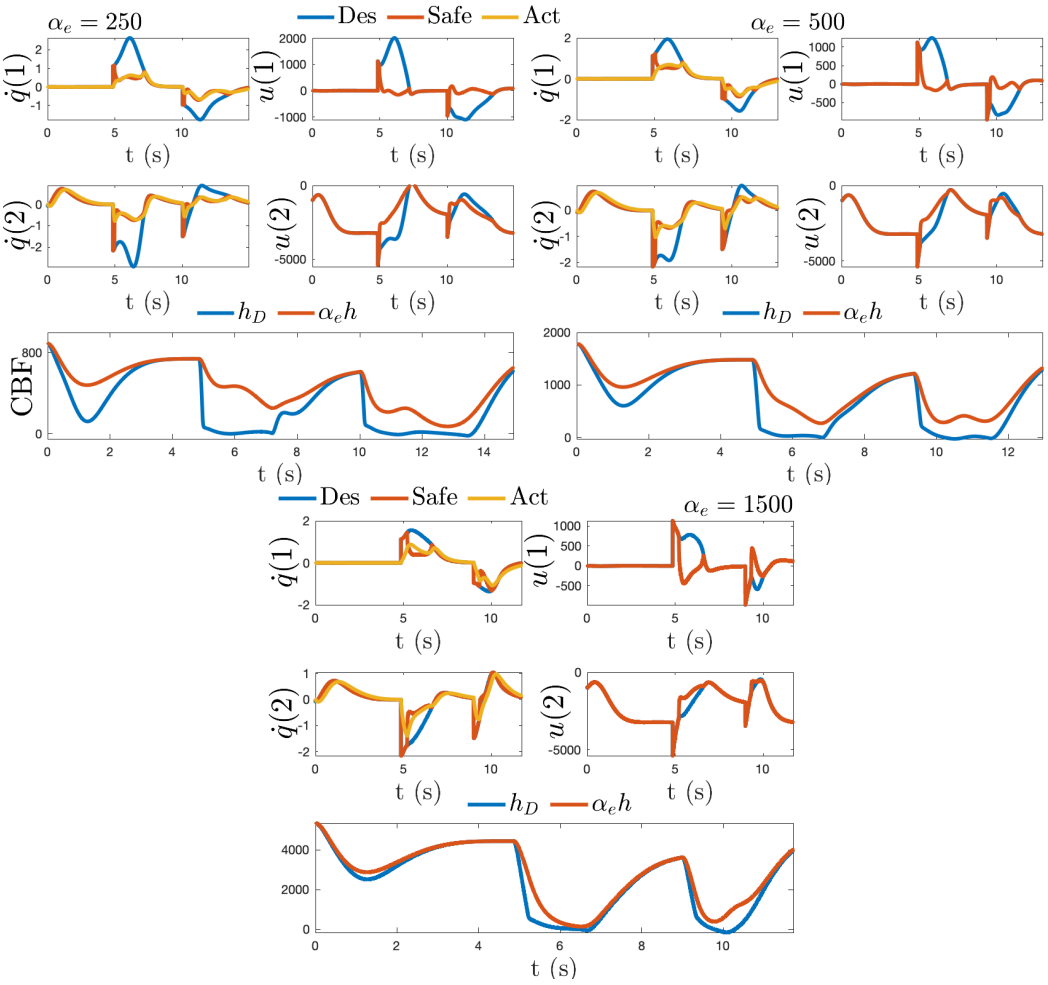



Figure 2.8: Energy-based kinematic CBF on the 6 DOF manipulator. Safety is guaranteed regardless of the choice of $\alpha_e$, but performance improves as $\alpha_e$ increases. The times taken to complete the second portion of the task, near the obstacle, are 10.07, 8.06, and 6.86 sec for $\alpha_e = \{250, 500, 1500\}$, compared to values of 7.60 to 7.94 for the purely kinematic case and 5.79 for CBF-free case.

**Example 5 (Energy-based kinematic CBF).** *The 6 DOF manipulator from Example 4 is now filtered with the constraint given in* (2.34)*, using* $c_u = 5\lambda_{\max}(D)$*. Figure 2.8 shows the result for different values of* $\alpha_e$*. Safety is guaranteed regardless of the value of* $\alpha_e$*, but as the value increases, the manipulator is able to move faster and get closer to obstacles, resulting in better performance.*

### 2.2.2 Underactuated systems

The methods developed can also be applied to underactuated systems, i.e., where $m \leq k$ and we have a potentially non-singular actuation matrix $B$. The key idea is to treat $h(q)$ as one of the coordinates. Choose a mapping $\Phi(q) := (w(q), h(q))$,

where $w$ is chosen such that $\Phi$ is a diffeomorphism. This can be easily obtained for non-singular configurations. We obtain the derivative as

$$\begin{bmatrix} \dot{w}(q,\dot{q}) \\ \dot{h}(q,\dot{q}) \end{bmatrix} = J_e(q)\dot{q}, \tag{2.39}$$

where $J_e(q)$ is the Jacobian matrix. $J_e$ is non-singular by property of diffeomorphism. We re-write the equations of motion of the robot as

$$D_e(q)\begin{bmatrix} \ddot{w} \\ \ddot{h} \end{bmatrix} + C_e(q,\dot{q})\begin{bmatrix} \dot{w} \\ \dot{h} \end{bmatrix} + G_e(q) = J_e(q)^{-T}Bu, \tag{2.40}$$

where

$$\begin{aligned} D_e(q) &= J_e(q)^{-T}D(q)J_e(q)^{-1} \\ C_e(q,\dot{q}) &= J_e(q)^{-T}C(q)J_e(q)^{-1} + J_e(q)^{-T}D(q)\dot{J}_e(q)^{-1} \\ G_e(q) &= J_e(q)^{-T}G(q), \end{aligned} \tag{2.41}$$

are the new terms that define the dynamics in the transformed space. It can be verified that the properties of $D_e, C_e$ will be same as that of $D, C$, i.e., $D_e$ is symmetric positive definite, and $\dot{D}_e - 2C_e$ is skew-symmetric. More details are in [106, Chapter 4, Section 5.4]. We can separate (2.40) into two parts:

$$\begin{aligned} D_{11}(q)\ddot{w} + D_{12}(q)\ddot{h} + C_1(q,\dot{q})\dot{q} + G_1(q) &= B_1(q)u \\ D_{21}(q)\ddot{w} + D_{22}(q)\ddot{h} + C_2(q,\dot{q})\dot{q} + G_2(q) &= B_2(q)u, \end{aligned} \tag{2.42}$$

where the terms corresponding to $D, C, G, B$ are apparent from the setup. $\ddot{w}$ can be eliminated from (2.42) to obtain

$$\underbrace{(D_{22} - D_{21}D_{11}^{-1}D_{12})}_{D_h}\ddot{h} + \underbrace{(C_2 - D_{21}D_{11}^{-1}C_1)}_{C_h}\dot{q} + \underbrace{G_2 - D_{21}D_{11}^{-1}G_1}_{G_h} = \underbrace{(B_2 - D_{21}D_{11}^{-1}B_1)}_{B_h}u, \tag{2.43}$$

where $D_h$ is nothing but the Schur complement form, and it is known to be symmetric positive definite [85, Proposition 1]. Note that here $B_h : Q \to \mathbb{R}^{1\times m}$ is the mapping from $u$ to the joints, which is assumed to have full row rank (in other words, $h$ is assumed to be inertially coupled with $u$. This may not be satisfied for all $Q$, in which case a subset $Q_u \subset Q$ is chosen (for example, in the cart-pole, pole-angle is not inertially coupled with $u$ when it is horizontal). With this formulation, we have the following theorem.

**Theorem 5.** *Consider a robotic system* (2.27) *and a kinematic safety constraint:* $h : Q \to \mathbb{R}$. *Consider the dynamically consistent extended CBF for underactuated systems:*

$$\widehat{h}_D(q, \dot{q}) := -\frac{1}{2}\dot{h}(q, \dot{q})^T D_h(q)\dot{h}(q, \dot{q}) + \alpha_e h(q) \tag{2.44}$$

*with the safe set:* $\widehat{S}_D := \{(q, \dot{q}) \in Q \times \mathbb{R}^k \ : \ \widehat{h}_D(q, \dot{q}) \geq 0\}$. *Then* $\widehat{S}_D \subset \mathcal{S}$ *and for all* $(q, \dot{q}) \in \widehat{S}_D$ *the following controller:*

$$\begin{aligned} u^*(q, \dot{q}, t) = &\underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \ \|u - u_{des}(q, \dot{q}, t)\|^2 \\ &\text{s.t. } -\frac{1}{2}\dot{h}\dot{D}_h\dot{h} - \dot{h}(-C_h\dot{q} - G_h) + \alpha_e\dot{h} - \dot{h}B_h u \geq -\alpha(\widehat{h}_D(q, \dot{q})) \end{aligned} \tag{2.45}$$

*guarantees forward invariance of* $\widehat{S}_D$, *i.e., safety of* $\widehat{S}_D$.

*Proof.* Differentiating $\widehat{h}$ yields:

$$\dot{\widehat{h}}_D = -\frac{1}{2}\dot{h}\dot{D}_h\dot{h} - \dot{h}(-C_h\dot{q} - G_h) + \alpha_e\dot{h} - \dot{h}B_h u. \tag{2.46}$$

It can be verified that if $\dot{h} = 0$, then the inequality in (2.45) is satisfied. The safety property follows directly. □

**Remark 2.** *Similar to Corollary 1, we can eliminate some of the model-based terms in* (2.45). *Specifically, we can replace the constraint in the QP with the following:*

$$-\frac{1}{2}c_l\dot{h}^2 - c_u|\dot{h}|(|\dot{q}|^2 + 1) + \alpha_e\dot{h} - \dot{h}B_h u \geq -\alpha(-c_u\dot{h}^2 + \alpha_e h(x)),$$

*where* $c_l, c_u$ *are constants that bound the norms:* $c_l \leq \|D_h\| \leq c_u$, $\|C_h\| \leq c_u|\dot{q}|$, $\|G_h\| \leq c_u$. *We have used the same notations for convenience. Note that these bounds may not exist for all* $(q, \dot{q}) \in Q \times \mathbb{R}^k$, *and they are dependent on the validity of the coordinate transformation* $\Phi$. *This is usually avoided by choosing a smaller configuration set* $Q_u$. *More details on the bounds are in [85].*

**Example 6** (**Cart-Pole System**). *To demonstrate these concepts, we consider the cart-pole system with two states, the cart position* $x$ *and the pole angle* $\theta$. *The system is actuated through a force input* $u$ *applied to the cart, which moves freely in a line. The safety constraint is to ensure that pole remains mostly upright, with* $\theta \in [\frac{5\pi}{6}, \frac{7\pi}{6}]$.

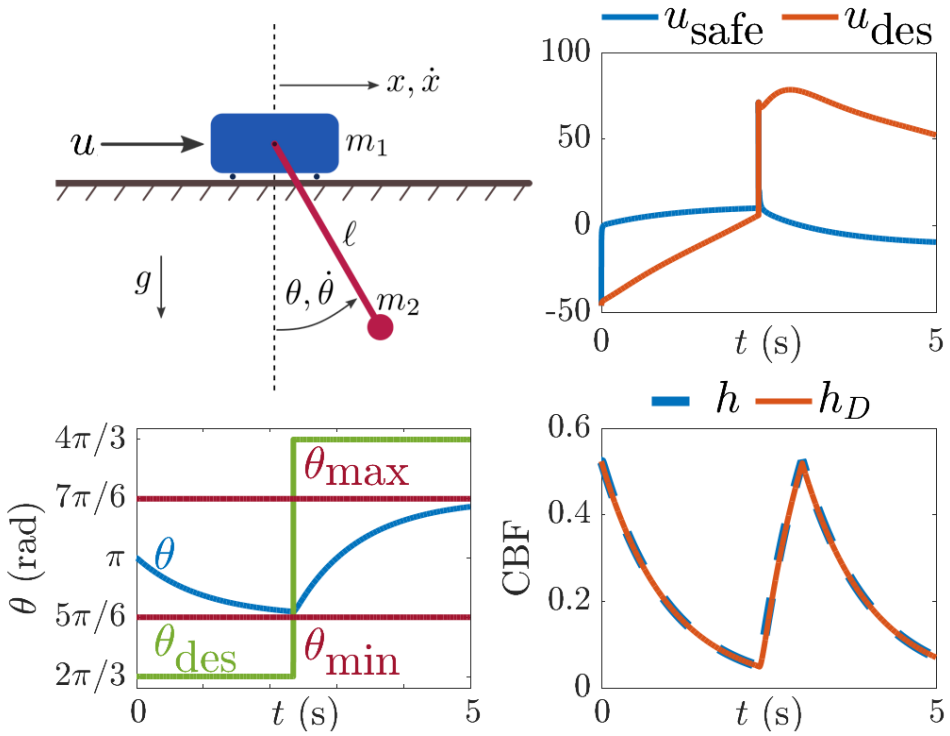



Figure 2.9: Cart-pole system with energy-based CBF.

### 2.3 Model-free guarantees

Similar to the energy-based approach presented in the previous section, the goal of this section is to provide safety guarantees for robots, despite filtering inputs at the velocity level, rather than the low-level control level.

To do this, we leverage the kinematic model of the manipulator and information on the lower-level velocity tracking controller to guarantee safe behavior on the full-order dynamics. Specifically, we establish that tracking the safe velocity obtained from the QP (2.24) results in safety under reasonable conditions on the tracking controller.

To see this, first consider the full-order dynamics associated with a robotic manipulator [106]:

$$D(q)\dot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu, \tag{2.47}$$

with $q, \dot{q} \in \mathbb{R}^n$, $D(q) \in \mathbb{R}^{n\times n}$ the inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ the Coriolis matrix, and $G(q) \in \mathbb{R}^n$ the gravity vector. Here we assume full actuation: the actuation matrix $B \in \mathbb{R}^{n\times n}$ is invertible and $u \in \mathbb{R}^n$. Associated with these dynamics is a control system of the form (1.1) with $x = (q, \dot{q})$ (hence $k = 2n$).

Motivated by the approach in [103], we assume the existence of a "good" low-level

velocity tracking controller on the manipulator (as is common on industrial robots). Concretely, for a velocity command $v^*(q,t)$ consider the corresponding error in tracking this velocity:

$$\dot{e} = \dot{q} - v^*, \tag{2.48}$$

and assume exponentially stable tracking.

**Assumption 1.** *There exist a low-level controller $u = k(x,t)$ for the control system* (1.1) *obtained from* (2.47) *such that*

$$\|\dot{e}(t)\|_2 \leq Me^{-\lambda t}\|\dot{e}_0\|_2 \tag{2.49}$$

*holds for some $M, \lambda > 0$ along the solution $x(t)$ of the closed-loop system with $q(t_0) = q_0$, $\dot{q}(t_0) = \dot{q}_0$ and $\dot{e}(t_0) = \dot{e}_0$.*

Under this assumption, we have the first theoretic result of the section which we state in general terms before applying it to the case of avoiding collisions.

**Theorem 6.** *Consider the full-order dynamics of a robot manipulator* (2.47) *expressed as the control system* (1.1)*, and the safe set $\mathcal{S}$. Assume that $h$ has bounded gradient, i.e., there exists $C_h > 0$ s.t. $\left\|\frac{\partial h}{\partial q}\right\|_2 \leq C_h$ for all $q \in \mathcal{S}$. Let $v^*(q,t)$ be the safe velocity given by the QP* (2.24)*, with corresponding error in* (2.48)*. If Assumption 1 holds with $\lambda > \alpha$, safety is achieved for the full-order dynamics* (2.47) *in that:*

$$(q_0, \dot{e}_0) \in \mathcal{S}_M \implies q(t) \in S, \quad \forall t \geq t_0, \tag{2.50}$$

*where:*

$$\mathcal{S}_M = \left\{(q,\dot{e}) \in \mathbb{R}^{2n} \ : \ h(q) - \frac{C_h M}{\lambda - \alpha}\|\dot{e}\|_2 \geq 0\right\}. \tag{2.51}$$

*Proof.* First, we lower-bound $\dot{h}(q,\dot{q})$ as follows:

$$\begin{aligned}\dot{h}(q,\dot{q}) &= \frac{\partial h}{\partial q}v^* + \frac{\partial h}{\partial q}\dot{e} \\ &\geq -\alpha h(q) - \left\|\frac{\partial h}{\partial q}\right\|_2 \|\dot{e}\|_2 \\ &\geq -\alpha h(q) - C_h M\|\dot{e}_0\|_2 \mathrm{e}^{-\lambda t},\end{aligned} \tag{2.52}$$

where we used (i) the definition (2.48) of the tracking error, (ii) the constraint on the safe velocity in (2.24) and the Cauchy-Schwartz inequality, and (iii) the upper bound

$C_h$ on $\|\frac{\partial h}{\partial q}\|_2$ and the upper bound (2.49) on the tracking error. Then, consider the following continuous function $y : \mathbb{R} \to \mathbb{R}$:

$$y(t) = \left(h(q_0) - \frac{C_h M \|\dot{e}_0\|_2}{\lambda - \alpha}\right) \mathrm{e}^{-\alpha t} + \frac{C_h M \|\dot{e}_0\|_2}{\lambda - \alpha} \mathrm{e}^{-\lambda t}, \tag{2.53}$$

which satisfies:

$$\begin{aligned} \dot{y}(t) &= -\alpha y(t) - C_h M \|\dot{e}_0\|_2 \mathrm{e}^{-\lambda t} \\ y(t_0) &= h(q_0). \end{aligned} \tag{2.54}$$

For $(q_0, \dot{e}_0) \in S_M$, we have $y(t) \geq 0, \forall t \geq t_0$, and by the comparison lemma we get:

$$h(q(t)) \geq y(t) \geq 0, \quad \forall t \geq t_0, \tag{2.55}$$

that implies $q(t) \in S, \forall t \geq t_0$. This completes the proof. □

### 2.3.1 Applications: Automated cooking with robotic arms

In order to prevent collisions with the environment, we must ensure that any point on the robot does not come into contact with any point in the environment. However, unlike the simple example before, we cannot rely on the robot and environment being represented by simple spheres.

Let us denote the set of all points on the robot as $A \subset \mathbb{R}^3$, and the set of all points in the collision environment as $B \subset \mathbb{R}^3$. To guarantee safety, we require that $A \cap B = \emptyset$, thus distance$(A, B) > 0$. More formally, *distance* is defined as:

$$\text{distance}(A, B) = \inf_{\substack{p_A \in A \\ p_B \in B}} \|p_A - p_B\|_2, \tag{2.56}$$

which can be computed in $\mathbb{R}^3$ using the GJK algorithm [57].

This notion gives a nonnegative distance, which could be used as CBF. However, it is advantageous to define a CBF that is negative in the event of collision, since CBFs may also ensure that the boundary of the set $\mathcal{S}$ is re-approached if $h(x) < 0$ [15]. In collision, *penetration* is defined as:

$$\text{penetration}(A, B) = \inf_{\substack{p_A \in A \\ p_B \in \overline{B}}} \|p_A - p_B\|_2, \tag{2.57}$$

where $\overline{B}$ is the complement of $B$, or the set of points outside the collision scene. Penetration is often computed using the EPA algorithm [153].

These two functions can be combined to form the notion of *signed distance*. Signed distance is typically written as

$$\mathrm{sd}(A, B) = \mathrm{distance}(A, B) - \mathrm{penetration}(A, B). \tag{2.58}$$

When the points $p_A$ and $p_B$ of the robot and the environment are given in local coordinates, the following expression from [131] can be utilized to compute the signed distance:

$$\mathrm{sd}_{AB}(q) = \max_{\substack{\tilde{n}\in\mathbb{R}^3 \\ \|\tilde{n}\|_2=1}} \min_{\substack{p_A\in A \\ p_B\in B}} \tilde{n} \cdot \left( F_A^{\mathrm{W}}(q) p_A - F_B^{\mathrm{W}} p_B \right), \tag{2.59}$$

where $F_A^{\mathrm{W}}(q) \in \mathbb{R}^{3\times3}$ gives the pose of the robot in the world frame that depends on the configuration $q$, and $F_B^{\mathrm{W}} \in \mathbb{R}^{3\times3}$ gives the pose of the collision environment, i.e., $F_A^{\mathrm{W}}(q)p_A$ and $F_B^{\mathrm{W}} p_B$ indicate points in the world frame.

Given the signed distance, we propose the CBF candidate:

$$h(q) = \mathrm{sd}_{AB}(q), \tag{2.60}$$

which defines the corresponding safe set of the system:

$$\mathcal{S} = \{q \in \mathbb{R}^n \ : \ h(q) = \mathrm{sd}_{AB}(q) \geq 0\}. \tag{2.61}$$

We remark that based on (2.59) $h$ can be written as:

$$h(q) = \hat{n}(q)^\top \left( F_A^{\mathrm{W}}(q) \hat{p}_A(q) - F_B^{\mathrm{W}} \hat{p}_B(q) \right). \tag{2.62}$$

Here $\hat{n}(q)$ and $\hat{p}_A(q)$, $\hat{p}_B(q)$ denote the direction and points that maximize and minimize the expression in (2.59), respectively, which depend on the configuration $q$.

It is important to note that in Euclidean space, signed distance, $h$, is differentiable almost everywhere, and satisfies $\left\| \frac{\partial h}{\partial p_A} \right\|_2 = 1$ [129]. There exists, however, a set of measure zero where $\frac{\partial h}{\partial q}$ is discontinuous, since functions $\hat{n}$ and $\hat{p}_A$, $\hat{p}_B$ are nonsmooth due to the max and min operators in (2.59). Since the above framework requires continuously differentiable $h$, we take special care in applying the theory, and we handle nonsmoothness under the following construction.

First, we express the gradient of $h$ as follows:

$$\frac{\partial h}{\partial q} = \hat{n}(q)^\top J_A(q) + \delta(q), \tag{2.63}$$

where $J_A(q) = \frac{\partial F_A^W}{\partial q}\hat{p}_A(q)$ and $\delta(q)$ is the remainder term associated with the derivatives of $\hat{n}$, $\hat{p}_A$, and $\hat{p}_B$. Importantly, note that $\hat{n}(q)^\top J_A(q)$ is continuous, while $\delta(q)$ is discontinuous on a set of measure zero. The term $\hat{n}(q)^\top J_A(q)$ can be interpreted as a continuous approximation of $\frac{\partial h}{\partial q}$, while the approximation error $\delta(q)$ acts as disturbance. The size of the disturbance is characterized by its essential supremum[1]:

$$\|\delta\|_\infty := \operatorname*{ess\,sup}_{t \geq t_0} \|\delta(q(t))\|_2.$$

The points where $h$ is not differentiable and $\delta$ is discontinuous occur on a set of measure zero, and therefore do not impact the essential supremum.

Now we incorporate the continuous approximation $\hat{n}(q)^\top J_A(q)$ in (2.63) into the control design. The following result demonstrates that this approximation is sufficient to maintain safety if the disturbance $\delta(q)$ is properly accounted for (in an input-to-state safety (ISSf) context [86, 11]).

**Proposition 3.** *Consider the kinematic model of a robotic manipulator* (2.22)*. Then, the controller expressed as the QP:*

$$\begin{aligned} v^*(q,t) = \underset{v \in \mathbb{R}^n}{\operatorname{argmin}} \;\; & \|v - v_{des}(q,t)\|_2^2 \\ \text{s.t.} \;\; & \hat{n}(q)^\top J_A(q) v \geq -\alpha h(q) + 2 J_{\max} \dot{q}_{\max}, \end{aligned} \tag{2.64}$$

*with* $\dot{q}_{\max} = \|\dot{q}\|_\infty$ *and* $J_{\max} = \max_{q \in \mathbb{R}^n} \|J_A(q)\|_2$*, renders the set* $\mathcal{S}$ *in* (2.61) *forward invariant for the resulting closed-loop system. That is, the controller* (2.64) *keeps system* (2.22) *safe.*

As such, collision-free behavior is enforced for the kinematic model of the manipulator, if the disturbance, i.e., the approximation error in (2.63), is accounted for in the controller. This is achieved by the last term in the constraint of (2.64).

*Proof.* First, we bound the essential supremum $\|\delta\|_\infty$ of the disturbance. Recall that the points where $h$ is not differentiable are on a set of measure zero and do not impact the essential supremum, thus, we construct the bound on $\|\delta\|_\infty$ by picking generic points where the $h$ is differentiable. For an arbitrary point on the robot

[1]The function $\delta$ is *essentially bounded* if $\|\delta(t)\|_2$ is bounded by a finite number for almost all $t \geq t_0$ (i.e., $\|\delta(t)\|_2$ is bounded except on a set of measure zero). The quantity $\|\delta\|_\infty$ is then defined as the least such bound.

$p_A \in A$ where $h$ is differentiable:

$$\begin{aligned}\left\|\frac{\partial h}{\partial q}\right\|_2 &= \left\|\frac{\partial h}{\partial p_A}\frac{\partial p_A}{\partial q}\right\|_2 \\ &\leq \left\|\frac{\partial h}{\partial p_A}\right\|_2 \left\|\frac{\partial p_A}{\partial q}\right\|_2 \\ &\leq 1 \cdot J_{\max}.\end{aligned} \tag{2.65}$$

This leads to the bound:

$$\begin{aligned}\|\delta\|_\infty &= \left\|\frac{\partial h}{\partial q} - \hat{n}(q)^\top J_A(q)\right\|_\infty \\ &\leq \left\|\frac{\partial h}{\partial q} - \hat{n}(q)^\top J_A(q)\right\|_2 \\ &\leq \left\|\frac{\partial h}{\partial q}\right\|_2 + \left\|\hat{n}(q)^\top J_A(q)\right\|_2 \\ &\leq J_{\max} + \|J_A(q)\|_2 \\ &\leq 2J_{\max}.\end{aligned} \tag{2.66}$$

Then, we differentiate the CBF $h$ in (2.60) and use (2.63):

$$\begin{aligned}\dot{h}(q,\dot{q}) = \frac{\partial h}{\partial q}\dot{q} &= \hat{n}(q)^\top J_A(q)\dot{q} + \delta(q)\dot{q} \\ &\geq \hat{n}(q)^\top J_A(q)\dot{q} - \|\delta\|_\infty \dot{q}_{\max}.\end{aligned} \tag{2.67}$$

Substituting $\dot{q}$ with the solution $v^*(q,t)$ to (2.64) and incorporating the bound on $\|\delta\|_\infty$, the result is:

$$\begin{aligned}\dot{h}(q, v^*(q,t)) &\geq \hat{n}(q)^\top J_A(q) v^*(q,t) - \|\delta\|_\infty \dot{q}_{\max} \\ &\geq -\alpha h(q) + 2J_{\max}\dot{q}_{\max} - \|\delta\|_\infty \dot{q}_{\max} \\ &\geq -\alpha h(q).\end{aligned} \tag{2.68}$$

Thus, the set $\mathcal{S}$ is forward invariant based on Theorem 1. □

Self-collisions are defined as collisions between any two links of the robot that are not explicitly allowed to collide. For these types of collisions, we still use the signed distance function, but now $F_B^{\mathrm{W}}$ also depends on the configuration $q$:

$$\mathrm{sd}_{AB}(q) = \max_{\substack{\tilde{n}\in\mathbb{R}^3 \\ \|\tilde{n}\|_2=1}} \min_{\substack{p_A\in A \\ p_B\in B}} \tilde{n} \cdot \left(F_A^{\mathrm{W}}(q)p_A - F_B^{\mathrm{W}}(q)p_B\right). \tag{2.69}$$

Thus, the gradient of $h(q) = \mathrm{sd}_{AB}(q)$ becomes:

$$\frac{\partial h}{\partial q} = \hat{n}(q)^\top \left(J_A(q) - J_B(q)\right) + \delta(q), \tag{2.70}$$

with $J_A(q) = \frac{\partial F_A^{\mathrm{W}}}{\partial q}\hat{p}_A(q)$ and $J_B(q) = \frac{\partial F_B^{\mathrm{W}}}{\partial q}\hat{p}_B(q)$.

Proposition 3 can again be applied to self-collisions, with slight modifications. The analysis results in the QP:

$$\begin{aligned} v^*(q,t) = \underset{v\in\mathbb{R}^n}{\operatorname{argmin}} \ & \|v - v_{\mathrm{des}}(x,t)\|_2^2 \\ \text{s.t. } & \hat{n}(q)^\top \left(J_A(q) - J_B(q)\right) v \geq -\alpha h(q) + 4J_{\max}\dot{q}_{\max}. \end{aligned} \tag{2.71}$$

The safety guarantees of Proposition 3 are valid for the kinematic model (2.22). However, like in Theorem 6, the controllers (2.64) and (2.71) lead to collision-free motion also on the full-order dynamics—assuming good velocity tracking.

**Theorem 7.** *Consider the full-order dynamics of a robot manipulator* (2.47) *expressed as the control system* (1.1)*, and the safe set* $\mathcal{S}$ *in* (2.61) *associated with the signed distance* $\mathrm{sd}_{AB}(q)$ *between the robot and the environment in* (2.59)*. Let* $v^*(q,t)$ *be the safe velocity given by the QP* (2.64)*, with corresponding error in* (2.48)*. If Assumption 1 holds with* $\lambda > \alpha$*, safety is achieved for the full-order dynamics* (2.47) *in that:*

$$(q_0, \dot{e}_0) \in \mathcal{S}_M \implies q(t) \in S, \quad \forall t \geq t_0, \tag{2.72}$$

*where:*

$$\mathcal{S}_M = \left\{ (q,\dot{e}) \in \mathbb{R}^{2n} \ : \ \mathrm{sd}_{AB}(q) - \frac{J_{\max} M}{\lambda - \alpha}\|\dot{e}\|_2 \geq 0 \right\}. \tag{2.73}$$

Note that the same safety guarantees can be stated for self-collision avoidance with the QP (2.71).

*Proof.* The proof follows the same steps as in the Proof of Theorem 6 with the substitution $C_h = J_{\max}$, which is justified by $\left\|\frac{\partial h}{\partial q}\right\|_2 \leq J_{\max}$ based on (2.65). Furthermore, note that $\frac{\partial h}{\partial q} v^* \geq -\alpha h(q)$ still holds due to (2.68). □

#### 2.3.1.1 CBF Implementation on Precomputed Trajectories

Assuming the knowledge of a reference trajectory, we now detail the trajectory *safety filter* algorithm. The most straightforward implementation of the QPs (2.64) and (2.71) is to run them in real-time paired with a desired joint velocity controller, which tracks the waypoints of the reference. This can be achieved with a P controller to the next waypoint $i$:

$$v_{\mathrm{des}}(q,t) = K_P(q^i_{\mathrm{des}} - q). \tag{2.74}$$

For the best results, the error on joint positions should be heavily saturated to avoid large differences in desired velocities at short and long distances. The tracked waypoint is iterated forwards either when the robot is sufficiently close $\left(\left\|q^i_{\text{des}} - q\right\|_2 < \epsilon\right)$, or when the robot gets stuck.

Due to the large time delay that many industrial manipulators have, it is often desired to instead send precomputed time-stamped trajectories, rather than attempting to track a trajectory online with feedback. The basic algorithm for generating these safe trajectories, given a cache of previously computed reference trajectories, is detailed in Algorithm 1.

**Algorithm 1** Trajectory generation in modified collision environments with safety filters.

**Require:** $C$, the cache that contains behaviors $C^i_B$, planning scenes $C^i_P$, and trajectories $C^i_X$
  **Input**
    $B$ Desired behavior
    $P$ Planning Scene
    $q$ Robot State
  **Output**
    $X$ Trajectory
  **for each** $C^i$ s.t. $B == C^i_B$ **do** ▷ Iterate through cache
    $T^i = f(C^i_P, C^i_{X_0}, P, q)$ ▷ Compute suitability metric
    **if** $T^i < T_1$ **then** ▷ Reference is extremely similar
      $X \leftarrow \text{CBF}(C^i_X, P, q)$
      **return**
    **end if**
  **end for**
  $[T_{\min}, \text{idx}] \leftarrow \min(T^i)$ ▷ Find best reference
  **if** $T_{\min} < T_2$ **then** ▷ Close match
    $X \leftarrow \text{CBF}(C^{\text{idx}}_X, P, q)$ ▷ Safety filter
    **return**
  **else if** $T_{\min} < T_3$ **then** ▷ Suitable match
    $X \leftarrow \text{CBF}(C^{\text{idx}}_X, P, q)$
    $C \leftarrow X$
    **return**
  **else** ▷ Best reference is very dissimilar
    $X \leftarrow$ Re-plan from scratch
    $C \leftarrow X$ ▷ $X$ gets added to cache
  **end if**

There are three fields of interest in the cached trajectories: the desired behavior $B$, the manipulator's trajectory $T$, and the collision environment used by the original

planner, referred to as the planning scene $P$. While only the joint trajectory is required to generate the modified, safe trajectory, the inclusion of the original planning scene allows for more information when choosing the closest trajectory to track.

The algorithm first assesses the suitability of previously computed trajectories in the cache. There are two major considerations: the difference in initial conditions and the similarity of the planning scene. The suitability of the $i^{\text{th}}$ member of the cache $C^i$ is evaluated by the function:

$$T^i = f(C^i_P, C^i_{X_0}, P, q) = \delta^i_q + \delta^i_P, \tag{2.75}$$

where

$$\delta^i_q = \left\| C^i_{X_0} - q \right\|_2 \tag{2.76}$$

$$\delta^i_P = \left\| C^i_P - P \right\| = \sum_{o \in O} \left\| C^i_{P_o} - P_o \right\| \tag{2.77}$$

assess the differences in the initial conditions of the robot and the collision objects $o \in O$ making up the planning scene.

There are three threshold values ($T_1$, $T_2$ and $T_3$) for this suitability metric. If $T^i < T_1$, then the search stops, as the trajectory in the cache is so close that it is not worth searching, and the CBF filter is applied. After searching through all cache members, if $T^i < T_2$, then the filter is applied, but the trajectory is not added to the cache to prevent it from growing unnecessarily large. If $T_2 < T^i < T_3$, then the filter is applied and the resulting trajectory is added to the cache. Finally, if $T^i > T_3$, then the original motion planning algorithm is used, and the result is added to the cache.

To obtain the joint trajectory $X$ via the CBF, we simply utilize a trajectory tracking controller like (2.74) along with the CBF-QP, and integrate its solution throughout the behavior.

Figure 2.10 shows the simulated cooking environment. The robot and obstacle representations are a series of meshes described by URDF and SRDF files. The position and orientation of objects are updated before each planning attempt, and collision objects in the environment are assumed to be stationary unless directly interacted with by the manipulator, such as the baskets being grabbed and moved.

To implement the CBF filter, we require three values to be computed: the signed distance to the obstacles and other links $\text{sd}(q)$, the normal vectors corresponding to

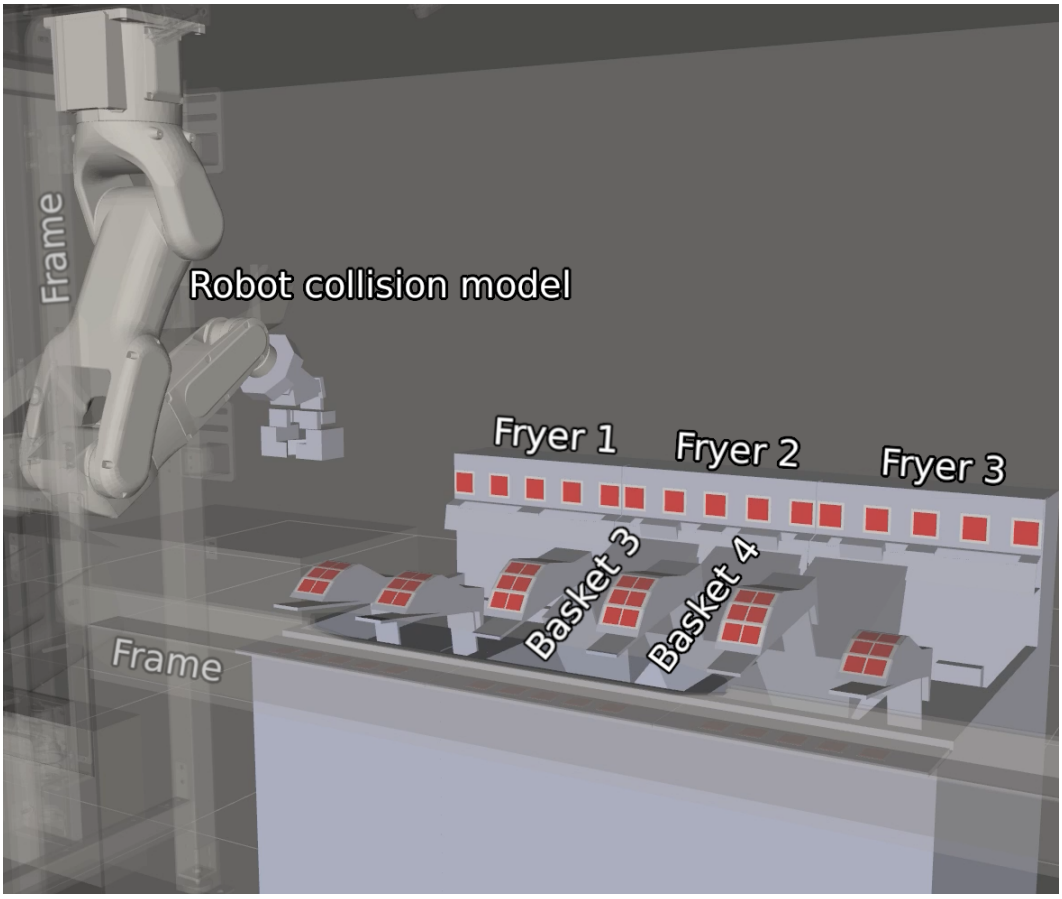



Figure 2.10: The simulation environment, which shows the collision objects and their representations as mesh files. The same mesh representations are used on the hardware system.

these points $\hat{n}(q)$, and the manipulator Jacobian at these points $J(q)$. The MoveIt framework [43], an open-source robotics software package for motion planning, is able to compute all three of these values. Specifically, the `distanceRobot()` and `distanceSelf()` functions of the `CollisionEnv` class provide the signed distances and normal vectors needed for environmental and self-collisions. Moreover, the `getJacobian()` function in the `RobotState` class returns the manipulator Jacobian. Thus, no other external libraries are required to implement this algorithm. Once these three values are computed, the OSQP quadratic program solver [145] is used to calculate the velocity commands subject to the CBF condition, and integration is done manually.

Before hardware implementation, the algorithm was tested in simulation. The resulting behaviors are described in the next section, and the simulation results are shown along with the hardware trajectories in Figure 2.11.

We apply the approach described in this section to one of the Miso Robotics robotic cooking environments. Specifically, we utilize a FANUC LR Mate 200iD/7LC robotic manipulator wrapped in a sleeve, and we send joint trajectories from an Intel i9-9900KF running ROS.

The cooking environment used in the testing is fully modeled using high-quality meshes used for collision checking. There are 36 collision objects in total, each represented by tens to hundreds of mesh triangles. The primary collision objects of concern are the six baskets, three industrial fryers, the hood vent over the fryers, and the glass pane separating the manipulator from the human workers. Of these objects, the baskets and fryers are the most commonly displaced.

As shown in the figures, the configuration space of the manipulator is very densely crowded with obstacles. To complete a behavior, it is common to have less than a few centimeters of clearance between the robot and the surrounding environment. For this reason, planning methods must be minimally conservative, and there is no room for any collision buffer.

For the purpose of the experiments, a minimal cache was utilized to highlight the role of CBFs in re-planning around obstacles. In a commercial setting, with a more populated cache, the CBF would have many more prior trajectories to choose from, meaning that the path modifications would be much smaller in magnitude. In practice, we find that the cache size saturates at around 200 stored behaviors.

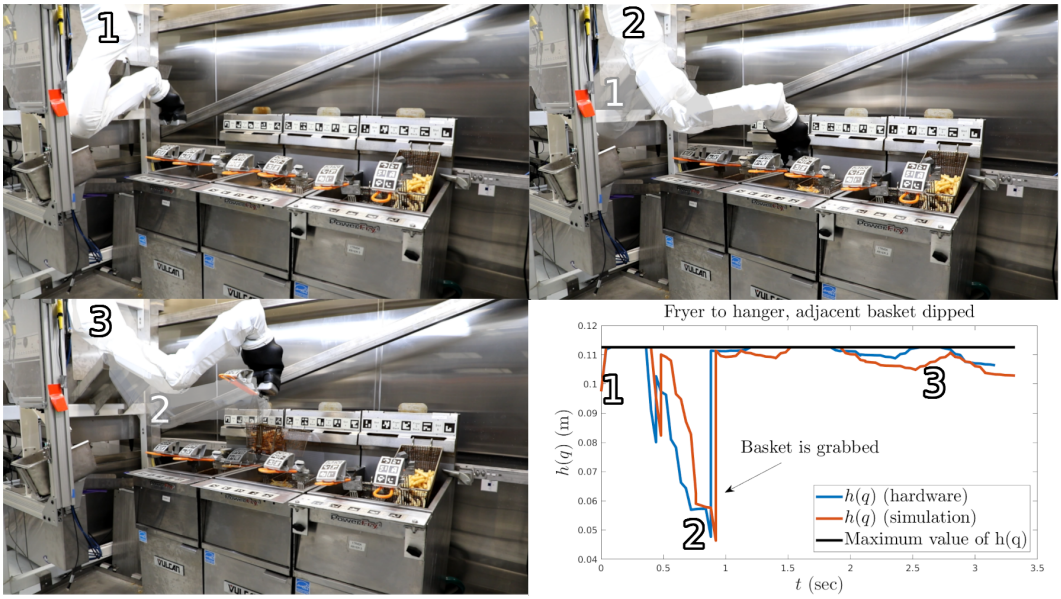
We test our framework's ability to safely re-plan on the two most volatile behaviors: `fryer_to_hanger` and `hanger_to_fryer`, described below.

**Fryer to hanger.** The `fryer_to_hanger` behavior moves a basket from the dipped state to the hanging state. The manipulator picks up a basket that has finished cooking and hangs it, allowing the oil to drip off the basket before serving food to customers.

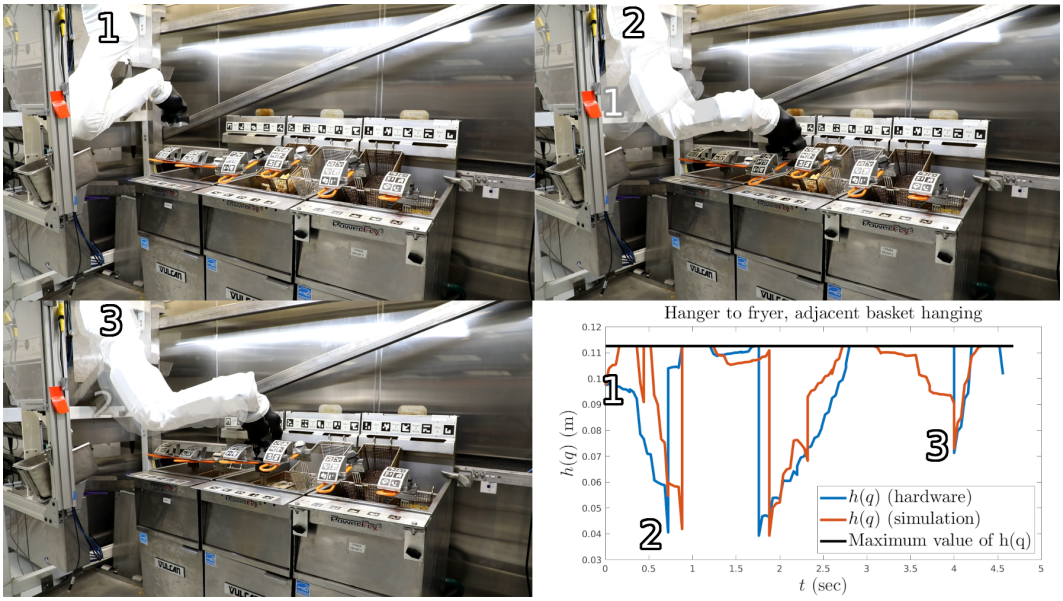
**Hanger to Fryer.** The `hanger_to_fryer` behavior is the reverse of `fryer_to_hanger`, transitioning a basket from the hanging state to the frying state.

Each behavior is tested in two primary configurations: one where the adjacent basket is submerged, and one where it's hanging. For the purpose of this section, each of the four testing configurations were run 25 times, each with different cached trajectories and planning environments, for 100 total executions. The testing methodology was simple: for each setup, we first run the CBF on the best matching reference trajectory in the limited cache, and then we re-plan using TrajOpt for comparison purposes.

*The CBF was able to produce a successful, collision-free trajectory in all 100 cases*, even with the artificially limited cache size. The average computation time per CBF call was 2 ms, and the average computation time for the entire behavior was 223 ms. This is a significant improvement compared to TrajOpt's average computation time of 5923. Note that the CBF's trajectory is updated every 10 ms compared to TrajOpt's 64 ms, meaning no additional local planner needs to be utilized. Two example trajectories from the CBF are visualized in Figure 2.11, and the value of $h(q)$ throughout the motion is included.




(a) `fryer_to_hanger` with adjacent basket in fryer.




(b) `hanger_to_fryer` with adjacent basket hanging.

Figure 2.11: Two examples behaviors implemented on the Flippy2 robot. See `https://youtu.be/nmkbya8XBmw` for video. The large spikes in signed distance $h(q)$ come from enabling and disabling collision objects when required for interaction, like the basket when gripping and the fryer when hanging. At the maximum value of $h(q)$, the robot is only 11 cm away from the frame around it during these behaviors.

*Chapter 3*

# INPUT REGULATION WITH BACKUP CONTROLLERS

To implement a CBF with input bounds, the set $\mathcal{S}$ must be control invariant, as mentioned before. In [59], however, it was shown that an explicit representation of a control invariant set $\mathcal{S}$ is not necessary. A control invariant subset $\mathcal{S}_I$ can be implicitly computed using what is now commonly referred to as the backup set method.

## 3.1 Backup set CBF

The approach for defining such a set is inspired by [69]. The idea is to start with a backup controller and backup set:

**Definition 6** (Backup controller and set)**.** *A **backup controller** is a predefined control law $\pi(x) : \mathcal{D} \to \mathcal{U}$ that attempts to take the system* (1.1) *from anywhere in $\mathcal{D}$ into the **backup set**, a small set that invariant under the backup controller.*

The size of the backup set does not affect the size of the implicitly computed invariant set, meaning that very conservative approaches can be used to compute it. The performance of the backup controller, however, is very critical to the size of the implicit safe set.

To compute the set $\mathcal{S}_I$, we first need to define the flow of the system along the backup control law.

**Definition 7** (Flow)**.** *The **flow** of the system* (1.1) *along a control law $u(x) : \mathcal{D} \to \mathcal{U}$, denoted as $\phi_t^\pi(x_0)$, is the solution to the initial value problem*

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= f(x) + g(x)u(x), \\ x(0) &= x_0. \end{aligned} \tag{3.1}$$

Furthermore, when the dynamics and control law are smooth, the sensitivity of the flow operator with respect to the initial time and state satisfies

$$D\phi_t^u(x_0) = Q(t), \tag{3.2}$$

where $Q(t)$ is a solution to the initial value problem [67]

$$\begin{aligned} \frac{\mathrm{d}Q}{\mathrm{d}t} &= D\left[(f+gu)\circ\phi_t^u(x)\right]Q(t), \\ Q(0) &= I. \end{aligned} \tag{3.3}$$

Consider a safe set $\mathcal{S}$, similar to (1.2), but now defined by $N_s$ continuously differentiable functions

$$\begin{aligned} S &= \{x \in \mathbb{R}^n \mid \forall i \in \{1,\ldots,N_s\},\ h_i(x) \geq 0\} \\ \partial S &= \{x \in S \mid \exists i \in \{1,\ldots,N_s\},\ h_i(x) = 0\}. \end{aligned} \tag{3.4}$$

With this, we can now define $\mathcal{S}_I$ as follows:

$$S^I = \left\{ x \mid \bigwedge_{\tau\in[0,T]} \left(\phi_\tau^\pi(x) \in \mathcal{S}\right) \wedge \left(\phi_T^\pi(x) \in \mathcal{S}_B\right) \right\}. \tag{3.5}$$

In short, $\mathcal{S}_I$ is defined as the intersection of two sets. The first set, described by $\left\{ x \mid \bigwedge_{\tau\in[0,T]} \left(\phi_\tau^\pi(x) \in \mathcal{S}\right) \right\}$, states that the flow of the system along the backup trajectory states in the safe set $\mathcal{S}$ for all time $t \in [0,T]$, where $T$ is the integration time chosen for the method, typically on the order of a few seconds.

The second set, described by $\left\{x \mid \left(\phi_T^\pi(x) \in \mathcal{S}_B\right)\right\}$, is the set of states such that the flow along the backup controller ends in $\mathcal{S}_B$ by time $T$. Note that this condition would be unnecessary if $T = \infty$, but this would make the first constraint intractable.

That is, the implicit set $\mathcal{S}^I$ consists of all initial conditions that are steered to $\mathcal{S}^B$ within time $T$ without exiting $\mathcal{S}$ along the way.

**Theorem 8.** *If $\mathcal{S}^B$ is invariant under $\pi(x)$, then $\mathcal{S}^I$ is a viable set contained in $S$.*

*Proof.* Consider $x \in \mathcal{S}^I$, then $\phi_\tau^\pi(x) \in \mathcal{S}$ for all $\tau \in [0,T]$. Containment of $\mathcal{S}^I$ in $\mathcal{S}$ follows from the special case $\tau = 0$. From the semi-group property of the flow and invariance of $\mathcal{S}^B$ under $\pi$ it follows that for any $\tau, s > 0$

$$\begin{aligned} x \in \mathcal{S}^I \implies \phi_s^\pi \circ \phi_\tau^\pi(x) \in \mathcal{S} \wedge \phi_T^\pi \circ \phi_\tau^\pi(x) \in \mathcal{S}^B \\ \implies \phi_\tau^\pi(x) \in \mathcal{S}^I, \end{aligned}$$

which implies that also $\mathcal{S}^I$ is invariant under $\pi$. Hence $\mathcal{S}^I$ is viable. □

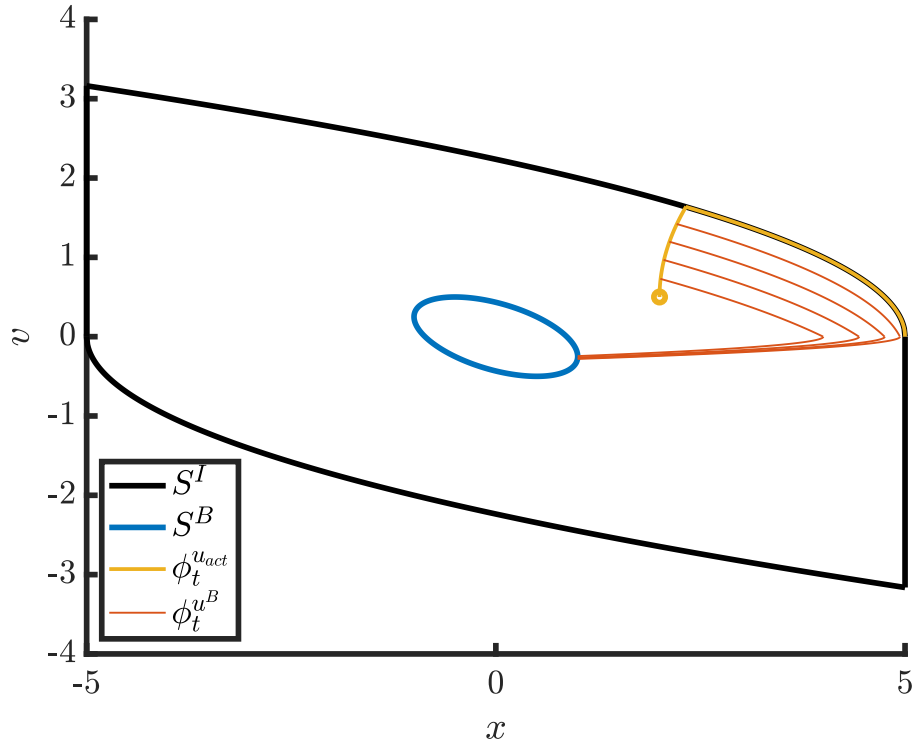



Figure 3.1: Illustration of the backup set and resulting implicit invariant set.

Moreover, there is no need to implement more than one constraint, as demonstrated by the following Lemma.

**Lemma 3.** *$\mathcal{S}_I$ is the 0-level set of the following function*

$$h(x) = \min\{\min_{t\in[0,T]} h(\Phi_t^\pi(x)), h_B(\Phi_T^\pi(x))\}. \tag{3.6}$$

*Proof.* First notice that by the continuity of the flow function $\Phi^\pi$ and the min function, $h$ is continuous. For all $x \in \mathcal{S}$, by definition, under the backup strategy $\pi$, the state evolution $\Phi_t^\pi(x)$ would satisfy the constraint and reach $\mathcal{S}_0$ at time $T$, therefore $h(x) \geq 0$. On the other hand, for all $x \notin \mathcal{S}$, under the backup strategy $\pi$, the state evolution either violates the state constraint at some $t$, i.e., $\exists t \in [0, T]$, $h(\Phi_t^\pi(x)) < 0$, or does not reach $\mathcal{S}_0$ within the horizon $T$, i.e., $h^{\mathcal{S}}(\Phi_t^\pi(x)) < 0$, indicating that $h(x) < 0$. Therefore, $\mathcal{S} = \{x|h(x) \geq 0\}$. □

Figure 3.1 illustrates the implicit viable set for a double integrator under an optimal backup controller with walls at $x = \pm 5$.

With a properly defined $\pi$ the implicit set $\mathcal{S}^I$ can be significantly larger than $\mathcal{S}^B$, and thus be less conservative. However, enforcement of the barrier condition requires knowledge of a collection of level set functions $h^I_j$ that together define $\mathcal{S}^I$, as well as their derivatives. In the following we construct such functions and then discuss how they can be computed on-the-fly via numerical integration.

Two types of constraints are required to define $\mathcal{S}^I$: one for ensuring that $\mathcal{S}^B$ is reached within time $T$, and a family of constraints that ensure that $S$ is kept invariant along the trajectory. Such constraints can be defined in terms of the functions $h^B$ and $h_j$ that define the sets $\mathcal{S}^B$ and $\mathcal{S}$, and the flow of the backup controller:

$$h^I_T(x) = h^B \circ \phi_T^{u^B}(x), \tag{3.7a}$$

$$h^I_{j,\tau}(x) = h_j \circ \phi_\tau^{u^B}(x). \tag{3.7b}$$

Equation (3.7b) represents an infinite collection of functions, which poses an issue that we will address later. The validity of the following proposition is clear from the definition of $S^B$.

**Proposition 4.** *$S^I$ is the super 0 level set of the functions defined in* (3.7)*, i.e.,*

$$S^I = \left\{ x : h^I_T(x) \geq 0 \wedge \bigwedge_{\tau\in[0,T]} \bigwedge_{j=1}^{r} \left( h^I_{j,\tau}(x) \geq 0 \right) \right\}. \tag{3.8}$$

From the chain rule of differentiation the gradients can be written as follows:

$$\nabla h^I_T(x) = D\left[h^B\right]_{\phi_T^{u^B}(x)} D\left[\phi_T^{u^B}\right]_x \left( f(x) + g(x)u^b(x) \right), \tag{3.9a}$$

$$\nabla h^I_{j,\tau}(x) = D\left[h_j\right]_{\phi_\tau^{u^B}(x)} D\left[\phi_\tau^{u^B}\right]_x \left( f(x) + g(x)u^b(x) \right). \tag{3.9b}$$

These expressions can both be evaluated if the flow $\phi_\tau^{u^B}(x)$ and the flow sensitivity $D[\phi_\tau^{u^B}]_x$ are known, which can be found via numerical integration of the closed-loop dynamics under the backup controller.

We now turn to the issue of having an infinite number of functions defining the set. In practice we can only enforce positivity of a finite number of the functions in (3.7) defining $S^I$, and therefore propose a safety filter that enforces positivity of a subset of $\epsilon$-tightened constraints evenly spaced in time:

$$L_f h^I_T(x) + L_g h^I_T(x) u \geq -\alpha_T(h^I_T(x)), \tag{3.10a}$$

$$L_f h^I_{j,k\eta}(x) + L_g h^I_{j,k\eta}(x) u \geq -\alpha_k(h^I_{j,k\eta}(x) - \epsilon), \tag{3.10b}$$

for $k = 0, 1, \ldots, T/\eta$.

Although this just enforces positivity of a finite number of the functions $h^I_{j,\tau}$, under some regularity conditions and appropriate margin $\epsilon$ we expect that this should be sufficient to guarantee positivity of the whole family of functions. We make this more precise below via the following lemma.

**Lemma 4.** *Let $L_h$ be the Lipschitz constant of $h$ with respect to the Euclidean norm and let*

$$L_\phi = \sup_{x \in S} \| f(x) + g(x)u^B(x) \|_2 \tag{3.11}$$

*be the maximal velocity of the closed-loop vector field. Then*

$$\left| h \circ \phi_t^{u^B}(x) - h \circ \phi_s^{u^B}(x) \right| \leq L_h L_\phi |t - s|. \tag{3.12}$$

*Proof.* Assume WLOG that $t \geq s$ and let $y = \phi_s^{u^B}(x)$

$$\begin{aligned} \left| h \circ \phi_t^{u^B}(x) - h \circ \phi_s^{u^B}(x) \right| &\leq L_h \left\| \phi_t^{u^B}(x) - \phi_s^{u^B}(x) \right\|_2 \\ &= L_h \left\| \phi_{t-s}^{u^B}(y) - y \right\|_2 \leq L_h L_\phi |t - s|, \end{aligned}$$

since $L_\phi$ is the maximal velocity of the vector field. □

It follows that enforcing invariance of $S$ via a finite subset of constraints as in

**Theorem 9.** *For $\epsilon \geq L_h L_\phi \frac{\eta}{2}$ the safety filter in* (3.10) *enforces invariance of $S^I$.*

*Proof.* The filter implies that $h^I_{j,\tau}(x) = h_j \circ \phi_\tau^{u^B}(x) \geq \epsilon$ for all $\tau = k\eta$, so by Lemma 4 we can for each $\tau$ find a $k^*$ such that

$$\left| h^I_{j,\tau}(x) - h^I_{j,k^*\eta}(x) \right| \leq L_h L_\phi \frac{\eta}{2}, \tag{3.13}$$

meaning that

$$h^I_{j,\tau}(x) \geq \epsilon - L_h L_\phi \frac{\eta}{2} \geq 0. \tag{3.14}$$

Thus all the functions defining $S^I$ are positive, and hence $S^I$ is invariant. □

### 3.1.1 Applications: Industrial manipulators

We now apply the method to the problem of collision avoidance in an environment with a robotic arm and a human. An advantage of the implicit approach is that the implicit safe set can be time-varying even when the backup set and backup controller

Figure 3.2: The IRB 6640 industrial manipulator.

are not. The dynamics of the robotic arm are described by the usual manipulator equations

$$M(q)\ddot{q} + C\left(q, \dot{q}\right)\dot{q} + G(q) = \tau, \tag{3.15}$$

where $q$ describe the joint angles and $\tau$ is a vector of applied torques.

For manipulators with many degrees of freedom the explicit expressions for $M(q)$, $C(q, \dot{q})$ and $G(q)$ are very complicated. As an alternative, they can be evaluated at given points via the Articulated Body Algorithm (ABA) that steps over links of the manipulator [49]. Only having "black-box" access to the equations of motion would pose a problem for most methods for finding invariant sets, but the implicit method proposed in this section only requires access to the numerical values of the dynamics and its derivatives. We rewrite the dynamics on state-space form and also add a time variable so that we can enforce safety of time-varying sets: $X = [q, \dot{q}, t]^T$.

We now consider the 6-link IRB 6640 manipulator from ABB, depicted in Figure 3.2. This robot has six degrees of freedom, making the overall system in 13-dimensional.

For the ARB 6640, the backup set is considered to be a vertical tube around the robot. In practice, this would be a small closed-off area that is inaccessible to

the human. For this implementation, it is described by the following set of angle constraints:

$$S^B = \{X \in \mathbb{R}^{13} \mid q_2 = \left[ -\frac{\pi}{12}, \frac{\pi}{12} \right] \quad q_3 = \left[ -\frac{7\pi}{12}, -\frac{5\pi}{12} \right]\}.$$

The safe set is then simply the union of the backup set and complement of the reachable set of the human in space-time over the duration of the backup control maneuver.

For the purpose of this demonstration, the human is modeled as a single integrator with a maximum velocity, meaning that the size of its reachable set grows linearly in time. By adding time as a state, we prevent the filter from being overly conservative, which would be the result if we only used the reachable set of the human over the time horizon of the backup controller.

If $(x_0, y_0)$ is the current position of the human, the reachable set of the human, or the complement of $S$, can be simply expressed as an ellipsoid (or an n-cylinder [80]) centered at $(x_0, y_0, H/2)$, where $H$ is the height of the human. We can then write this set as the superlevel set of a time-dependent differentiable function $h : \mathbb{R}^n \to \mathbb{R}$

$$h(t, x, y, z) = (x - x_0)^2 + (y - y_0)^2 + \frac{(z - \sqrt{H})^2}{H/(r_0 + v_{\max} t)} - (r_0 + v_{\max} t)^2.$$

Thus, when $h(t, x, y, z) > 0$, for all points along the robot, the robot is not contacting the human. Similarly to sampling along the backup trajectory, we would theoretically need to check an infinite number of points. Again, however, we can pick a finite number of samples along the robot to enforce this condition. This sampling does not affect the guarantee on safety, as one can simply increase the radius of the human $r_0$ and the height $H_h$ by the spacing between the points. It does, however, add conservativeness to the problem, so the choice is sampling becomes a tradeoff between computational performance and system performance.

As the dynamics of the robot are defined in joint space, and the safety set is defined in Cartesian space, one must be careful when implementing the barrier condition. Let us define our forward kinematics function that takes us from joint space to Cartesian space as $K(q, t) : \mathbb{R}^{n+1} \to \mathbb{R}^4$, augmenting it with the identity map for time.

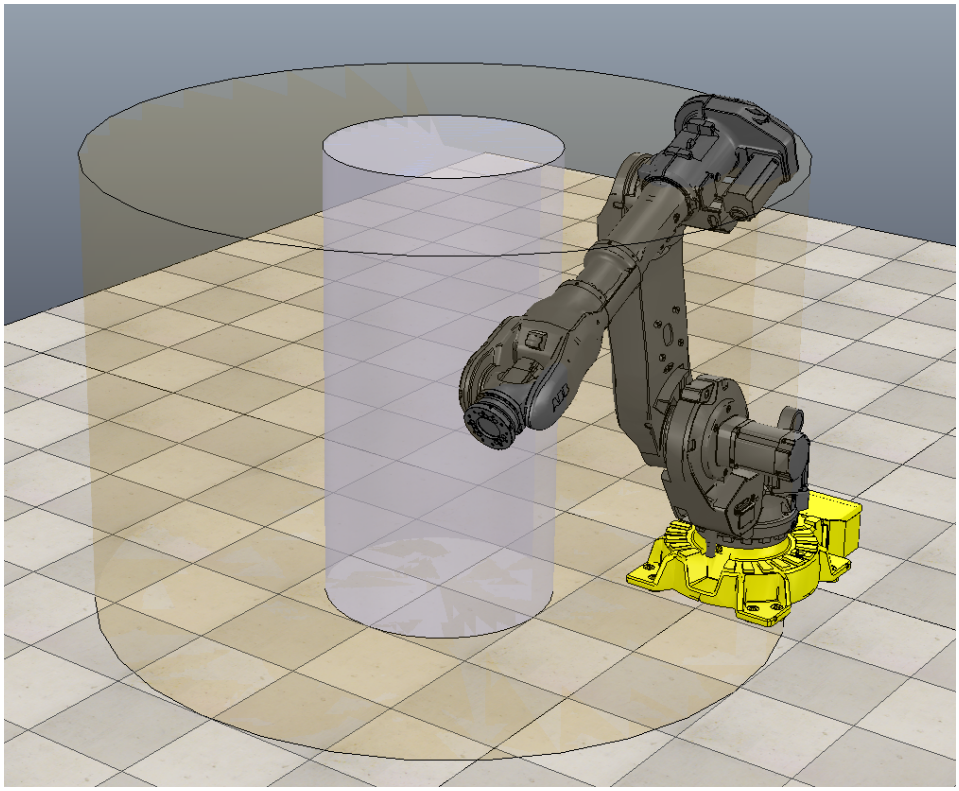


Figure 3.3: The set describing the human at $t_0$ and reachable set after one second.

Consider $E = [x, y, z, t]^T$ and $X = [q, \dot{q}, t]^T$. The gradient of $h$ with respect to the states is

$$\frac{\partial h(E)}{\partial X} = \frac{\partial h(K(q))}{\partial X} = \left(\frac{\partial h(E)}{\partial E} \circ K(q)\right) \frac{\partial K}{\partial X},$$

where

$$\frac{\partial K}{\partial X} = \begin{bmatrix} \frac{\partial K}{\partial q} & \frac{\partial K}{\partial \dot{q}} & \frac{\partial K}{\partial t} \end{bmatrix} = \begin{bmatrix} J & \vec{0} & J\dot{q} \\ \vec{0} & \vec{0} & 1 \end{bmatrix},$$

where the Jacobian $J$ is calculated numerically.

For the backup controller, we will leverage the power of the recursive Newton-Euler algorithm (RNEA) [79], which provides the necessary joint torques to generate desired joint accelerations. The flexibility of this method is again showcased by the fact that we do not need an analytic expression for the backup controller, as long as we know its gradient.

There are only two joints that require actuation to reach the backup set. A simple PD controller is used to obtain desired joint accelerations for these joints, which is

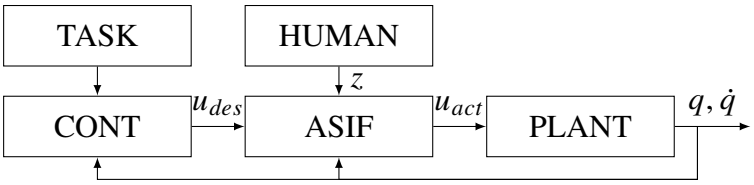



Figure 3.4: Block diagram of the ROS nodes used in the simulations.

fed into the RNEA that generates the control inputs, as well as their gradient. The controller is of the form,

$$a_{\text{des}}(q, \dot{q}) = -k_p(q - q_d) - k_d(\dot{q})$$
$$u_{\text{b}}(q, \dot{q}) = \text{RNEA}(q, \dot{q}, a_{\text{des}}(q, \dot{q})).$$

The gradient of this backup controller, which is required to evaluate (3.9) online, is described by

$$\frac{\partial u_b}{\partial q} = \frac{\partial \text{RNEA}}{\partial q} + \frac{\partial \text{RNEA}}{\partial a_{\text{des}}}\frac{\partial a_{\text{des}}}{\partial q} = \frac{\partial \text{RNEA}}{\partial q} - k_p\frac{\partial \text{RNEA}}{\partial a_{\text{des}}},$$
$$\frac{\partial u_b}{\partial \dot{q}} = \frac{\partial \text{RNEA}}{\partial \dot{q}} + \frac{\partial \text{RNEA}}{\partial a_{\text{des}}}\frac{\partial a_{\text{des}}}{\partial \dot{q}} = \frac{\partial \text{RNEA}}{\partial \dot{q}} - k_d\frac{\partial \text{RNEA}}{\partial a_{\text{des}}},$$
$$\frac{\partial u_b}{dt} = 0.$$

Since the RNEA provides the exact torques needed to achieve desired joint accelerations, the forward invariance of the backup controller is almost trivially guaranteed under the proper choice of desired joint accelerations.

The rigid body algorithm library used for this simulation is Pinocchio [31]. This C++ library has been shown to be the fastest of its kind, with the Table 3.1 illustrating the average computation times of each necessary expression for our robot.

Table 3.1: Computation time of IRB 6640 in Pinocchio

| Expression | Time ($\mu$s) |
|---|---|
| Affine forward dynamics ($f(x)$ and $g(x)$) | 4 |
| Gradient of closed-loop forward dynamics | 42 |
| Backup controller | 5 |
| Gradient of backup controller | 31 |

A ROS environment was created to simulate the system, with V-REP used as a visualizer. The ROS package consisted of five nodes: the robotic arm (PLANT), the task giver (TASK), a nominal controller (CONT), the human (HUMAN), and the safety filter (ASIF), connected as shown in Figure 3.4. Each component of the system ran at 200 Hz on a desktop PC with an Intel 8700k processor. The dynamics were integrated in the plant node via the Boost C++ library, with the `runge_kutta_dopri5` scheme over the timestep of 5 ms.

The controller node tracked a sequence of desired end-effector positions, given to it by the task giver node. Once the system reached the desired position, the task giver would send a new desired location to the system. The RNEA is also used for this tracking controller.

The human node allowed the user to joystick a human, modeled as a single integrator, around the factory floor.

Lastly, the safety filter node handled safety for the system. It takes in the state from the plant and the desired inputs from the controller, and outputs the actual inputs that are used for integration by the plant.

The ASIF uses an adaptive-step RK4 scheme for integration under the backup controller, and the resulting quadratic program is solved by the OSQP library [145].

Figure 3.5 shows the value of the ASIF when a human attempts to pass through the arm. This image well illustrates the minimally invasive property of the ASIF, as the filter keeps the value of $h(x)$ just barely above zero.

### 3.1.2 Handling sampled-data systems and input delay

Control theory in practice is almost always implemented in the form of a digital controller on a physical system that evolves continuously. However, these systems are rarely treated as such due to the difficulties that arise in the formulation of controllers that act optimally for these types of systems. Generally, the system is controlled rapidly enough that the time discretization can be ignored, and the entire system can be treated as continuous, allowing for a much larger class of control techniques. This is especially true for robotic systems, where continuous controllers are often implemented at loop rates faster that 1 kHz. In the context of safety-critical control, however, it is important to model the system as accurately as possible, in order to extend the guarantees from theory to practice. Moreover, optimization-based controllers tend to be less robust than simple control laws such as PID, and therefore this calls for a more accurate model.

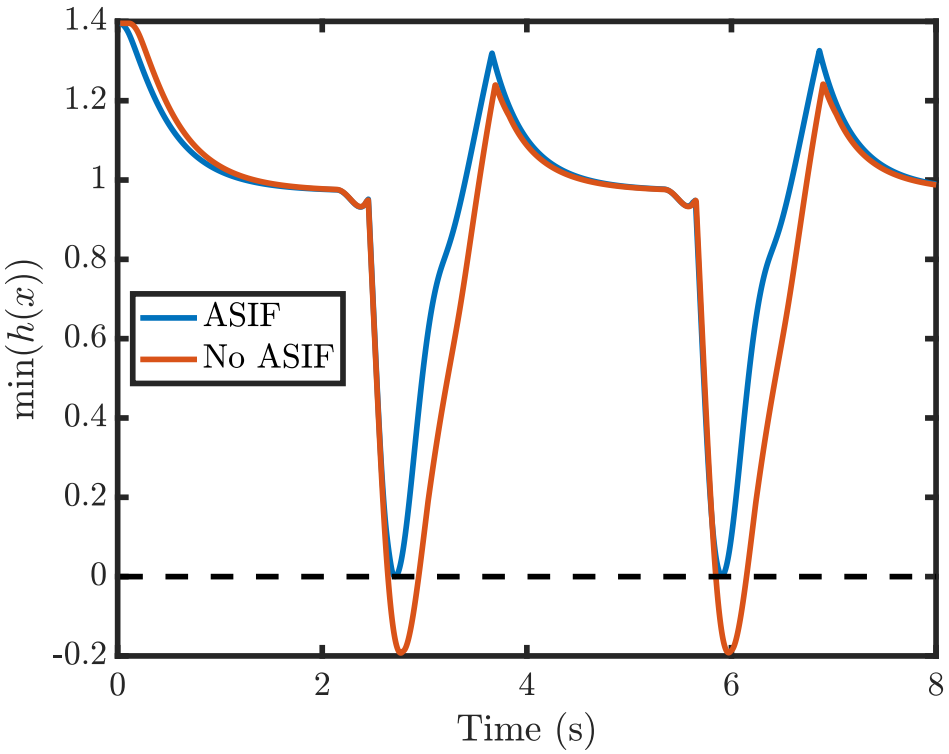



Figure 3.5: Value of the Barrier Function with and without ASIF engaged.

Another reality of control theory in practice is the presence of input delay. Input delay for myopic, optimization-based controllers (such as Control Lyapunov Functions [111] and control barrier functions), is often handled by making the problem formulation robust to any value of the input delay in some bounded set. However, this treatment degrades performance due to conservatism and complicates the (already difficult) computation of the Lyapunov or barrier functions. In practice, the input delay for a system is relatively easy to identify, so it would be beneficial to formulate the problem with the knowledge of the input delay of the system. While many solutions to handling specific time delays have been proposed, in general they either require either linear systems [70, 52], are applicable only to autonomous systems [115], require difficult construction [71], or rely on frequency-domain analysis that is not applicable to these optimization-based controllers [170].

The contributions of this subsection are:

- We propose a formulation of the implicitly defined control barrier function that is applicable to sampled-data systems, while retaining scalability.

- We use the concept of incremental stability [17] to prove robustness of the proposed backup controller-based CBF controller under state uncertainty.
- We are able to guarantee safety under a known input delay with much less conservatism under the proposed framework.

While condition (3.5) can be used to guarantee safety for continuous-time systems, it relies on the input being computed and applied continuously. In the presence of a zero-order hold controller, the description of the invariant set need to be modified. The zero-order hold backup controller, denoted as $\overline{u_B}(x(\cdot), t)$, simply takes on the value of $u_B(x)$ every $\Delta_t$ seconds, and holds that value until its next update:

$$\overline{u_B}(x(\cdot), t) = u_B(x(\lfloor t/\Delta_t \rfloor \Delta_t)), \tag{3.16}$$

where $\lfloor \cdot \rfloor$ is the largest integer not greater than the argument.

**Remark 3.** *The flow of the system under the zero-order backup controller $\phi_t^{\overline{u_B}}$ is well-defined, as unique solutions to sampled-data systems exist so long as the underlying controller is piecewise continuous (c.f. [92, p. 16]).*

The updated control invariant set under the zero-order hold backup controller can be written as

$$\begin{aligned}\overline{\mathcal{S}_I} = \Bigg\{ x \in \mathbb{R}^n \mid \Big(\forall \tau \in [0, T],\, h\left(\phi_\tau^{\overline{u_B}}\right) \geq 0\Big) \text{ and } \\ \left(h_B\left(\phi_T^{\overline{u_B}}\right) \geq 0\right)\Bigg\}.\end{aligned} \tag{3.17}$$

This is a control invariant set for the system under a zero-order controller with the same sampling time $\Delta_t$ as $\overline{u_B}$. The proof follows from [59, Theorem 1].

**Remark 4.** *While the flow of the system under the zero-order hold backup controller is Lipschitz [2], it is nonsmooth. Because of this, the barrier function itself is nonsmooth, and thus $\dot{h}$ cannot be expressed at finitely many points, which correspond to when the controller is updated. Despite this, a nonsmooth barrier function is valid if $\dot{h} \geq -\alpha(h)$ almost everywhere. For proof, see [58, Lemma 2.2].*

To enforce the CBF condition, $\dot{h}$ needs to be computed, which requires $\frac{\partial \phi_t^{\overline{u_B}}}{\partial x}$ to be evaluated. This expression is continuous over each controller sampling time, and can be computed using finite-differences [95].

To do this, simply integrate forward $n+1$ initial conditions under $\overline{u_B}$ to evaluate $\frac{\partial \phi^{u_B}_{(i+1)\Delta_t}(x)}{\partial \phi^{u_B}_{i\Delta_t}(x)}$ at each time-step. Then, use the chain rule to get

$$\frac{\partial \phi^{u_B}_{i\Delta_t}(x)}{\partial x} = \prod_{n=0}^{i-1} \frac{\partial \phi^{u_B}_{(n+1)\Delta_t}(x)}{\partial \phi^{u_B}_{n\Delta_t}(x)}. \tag{3.18}$$

The last caveat to consider in the implementation of the CBF condition is the fact that the condition must be met over the entire time horizon of the zero-order hold controller. Note that this consideration must be taken regardless of the method used for expressing the robust control barrier function, and was a subject of prior research of the authors [60].

Consider verifying the barrier function over the horizon of a single time-step of the zero-order hold controller with sample time $\Delta_t$,

$$h(\phi^{\overline{u_B}}_{\tau}(x_0)) \geq 0 \qquad \forall\, \tau \in [0, \Delta_t]. \tag{3.19}$$

The robust satisfaction of the above condition can be verified by checking the stronger condition shown in [60],

$$h(\mathcal{R}(x_0, \Delta_t)) \geq 0, \tag{3.20}$$

where $\mathcal{R}(x_0, \Delta_t)$ is the set of states reachable from $x_0$ in time $\Delta_t$ with any input $u \in \mathcal{U}$.

**Remark 5.** *This condition adds conservatism to the barrier formulation. While checking only the points $\phi^{u}_{\tau}(x_0)$ $\forall \tau \in [0, \Delta_t]$ would result in a more performant condition, this would make the constraint no longer affine, due to its dependence on the decision variable $u$.*

Let $\overline{u_B}(\cdot)$ be the input signal w.r.t. the nominal state flow, the robust CBF condition defined from the set $\overline{\mathcal{S}_I}$,

$$\begin{aligned}
&\left.\frac{dh}{dx}\right|_{\phi^{\overline{u_B}(\cdot)}_{\tau}(\mathbf{x_0})} \frac{\partial \phi^{\overline{u_B}(\cdot)}_{\tau}(\mathbf{x_0})}{\partial x} \left(f(\mathbf{x_0}) + g(\mathbf{x_0})u\right) + \alpha(h(\phi^{\overline{u_B}(\cdot)}_{\tau}(\mathbf{x_0}))) \geq 0 \\
&\left.\frac{dh_B}{dx}\right|_{\phi^{\overline{u_B}(\cdot)}_{T}(\mathbf{x_0})} \frac{\partial \phi^{\overline{u_B}(\cdot)}_{T}(\mathbf{x_0})}{\partial x} \left(f(\mathbf{x_0}) + g(\mathbf{x_0})u\right) + \alpha(h_B(\phi^{\overline{u_B}(\cdot)}_{T}(\mathbf{x_0}))) \geq 0,
\end{aligned} \tag{3.21}$$

where $\mathbf{x_0} = \mathcal{R}(x_0, \Delta_t)$, and the first inequality must hold for all $\tau \in [0, T]$.

**Proposition 5.** *Let $(\overline{u}_i)_{i=0}^{\infty}$ be a sequence of inputs that satisfies* (3.21) *at the beginning of each time-step and is applied with zero-order hold to the system* (1.1)*. If $x_0 \in \overline{\mathcal{S}_I}$, then $\phi_t^{\overline{u}}(x_0) \in \overline{\mathcal{S}_I}$ for all $t \geq 0$.*

*Proof.* Consider any time interval of a single time-step $\mathcal{T} = [t_0, t_0 + \Delta_t]$, and assume $x_{t_0} \in \overline{\mathcal{S}_I}$.

Denote $\Phi := \{\bigcup_{t\in\mathcal{T}} \phi_t^{\overline{u}_i}(x_{t_0})\}$. Since $\mathbf{x_{t_0}} = x_{t_0} + \mathcal{R}(x_{t_0}, \Delta_t)$ covers all states reachable from time $t_0$, $\Phi \subset \phi_t^{\overline{u}_i}(\mathbf{x_{t_0}})$, Therefore, if $\overline{u}_i$ satisfies (3.21) for $\mathbf{x_{t_0}}$, then the CBF condition holds for $\Phi$ as well. Thus, by the invariance of $\overline{\mathcal{S}_I}$, $\phi_t^{\overline{u}_i}(x_{t_0}) \in \overline{\mathcal{S}_I}$ for all $t \in \mathcal{T}$.

Since this condition is met over the entire sequence of time-steps, $\phi_t^{\overline{u}}(x_0) \in \overline{\mathcal{S}_I}$ for all $t \geq 0$. □

**Remark 6.** *It is possible that, for some $x_0 \in \mathcal{S}_I$, that $x_0 \in \mathcal{S}_I$ but $\mathbf{x_0} \notin \mathcal{S}_I$. In this case, the system is inside of its control invariant set, but the CBF condition* (3.21) *cannot be satisfied. This is due to conservatism mentioned in Remark 5. However, when this occurs, the backup control action can be taken. Thus, the system will stay safe for all time. Furthermore, this occurs on a very small set at the boundary of $\mathcal{S}_I$, which the strengthening term $\alpha(\cdot)$ makes difficult to reach.*

Note that the condition is evaluated here over a set, rather than a single point. The evaluation of $\phi_\tau^{\overline{u_B}(\cdot)}(\mathbf{x_0})$ poses the most difficulty, as it involves robustly integrating over a set. This makes techniques like interval arithmetic [72, 61] difficult to implement, due to numerical issues. To show that safety can be guaranteed for a small neighborhood of initial conditions, we adopt the concept of incremental stability (c.f. [17]).

**Definition 8.** *Given the dynamic system in* (1.1)*, the system is incrementally stable inside a set $\mathcal{X} \subseteq \mathbb{R}^n$ if $\forall T \geq 0$, $\forall\ x_1, x_2 \in \mathcal{X}$ and $u(\cdot) : [0, T] \to \mathbb{R}^m$ such that $\phi_t^{u(\cdot)}(x_1)$ and $\phi_t^{u(\cdot)}(x_2)$ stay inside $\mathcal{X}$, the evolution of the state satisfies $\left\|\phi_t^{u(\cdot)}(x_1) - \phi_t^{u(\cdot)}(x_2)\right\| \leq \beta(\|x_1 - x_2\|, t)$, where $\beta : \mathbb{R} \times [0, T] \to \mathbb{R}$ is nonincreasing in $t$ and $\forall\ t \in [0, T]$, $\beta(\cdot, t)$ is a class-$\mathcal{K}$ function.*

**Proposition 6.** *Suppose there exists a Lyapunov function $V : \mathcal{X} \to \mathbb{R}$ that satisfies $c_1||x|| \leq V(x) \leq c_2||x||$ for some $c_2 \geq c_1 > 0$. For two initial conditions $x_1, x_2 \in \mathcal{X}$ and an input signal $u(\cdot)$ such that $\phi^{u(\cdot)}(x_1)$, $\phi^{u(\cdot)}(x_2)$, and $\phi^{u(\cdot)}(x_1) - \phi^{u(\cdot)}(x_2) \in \mathcal{X}$, let $V(t) = V(\phi_t^{u(\cdot)}(x_1) - \phi_t^{u(\cdot)}(x_2))$. If $\dot{V} \leq 0$, then the system is locally incrementally stable in $\mathcal{X}$.*

*Proof.* The proof follows from the fact that $V(\cdot)$ and $||\cdot||$ are equivalent norms. □

In the context of control barrier functions with a backup strategy, if the system is incrementally stable, then given a nominal initial condition $x_0$ and an uncertainty set characterized as a level-set of the Lyapunov function, $\forall x \in \{x|V(x-x_0) \leq \epsilon\}$, for any input signal $u(\cdot)$, $\phi_t^{u(\cdot)}(x) \in \{x|V(x-\phi_t^{u(\cdot)}(x_0)) \leq \epsilon\}$. We shall show how this result can simplify the robust CBF condition in (3.21), which requires that the CBF condition hold for a small set $\mathcal{R}(x_0, \Delta_t)$ around the nominal initial condition.

The system (1.1) may not incrementally stable, but pre-feedback can be used to make it so. Since the error dynamics are being considered, the nonlinear dynamics are linearized to simplify the analysis. Given a set $\mathcal{X} \subseteq \mathbb{R}^n$ of states, multiple linear dynamics models $\dot{x} = A_i x + B_i u,\ i = 1, ..., N$ can be obtained by considering the extreme points of $\mathcal{X}$. Given a quadratic Lyapunov function $V = x^\intercal P x$ where $P$ is symmetric and positive definite, and an input set hyperbox defined as $\mathcal{U} = \{-u^{\max} \leq u \leq u^{\max}\} \subseteq \mathbb{R}^m$, we develop the following Linear Matrix Inequality (LMI) to search for a pre-feedback gain that guarantees incremental stability for the system:

$$\begin{aligned} \min_{K \in \mathbb{R}^{n\times m}} \ & ||\Lambda P^{-\frac{1}{2}} K^\intercal||_\infty \\ \text{s.t. } & \forall i = 1, ..., N, P(A_i + B_i K) + (A_i + B_i K)^\intercal P \leq 0, \end{aligned} \tag{3.22}$$

where $\Lambda = \text{diag}(\frac{1}{u_1^{max}},\ \dots,\ \frac{1}{u_m^{max}})$.

The cost function is chosen due to the fact that

$$\{\max_x\ |K_i x| \text{ s.t. } x^\intercal P x \leq 1\} = \sqrt{K_i P^{-1} K_i^\intercal}, \tag{3.23}$$

which means that the pre-feedback is available within the level set $\{x|x^\intercal P x \leq \min_{i=1,...,N} \frac{u_i^{\max}}{\sqrt{K_i P^{-1} K_i^\intercal}}\}$. Therefore, minimizing the cost function in (3.22) is maximizing the size of the level-set of the Lyapunov function in which the pre-feedback is available.

**Proposition 7.** *Given a dynamic system as described in* (1.1) *with* $\mathcal{U} = \{-u^{\max} \leq u \leq u^{\max}\}$*, a set* $\mathcal{X} \subseteq \mathbb{R}^n$*, and a Lyapunov function* $V(x) = x^\intercal P x$*,* $P \geq 0$*, assume that* $\forall\, x_1, x_2 \in \mathcal{X}, \forall\, u \in \mathcal{U}$*,* $f(x_1)+g(x_1)u-f(x_2)-g(x_2)u \in Conv(A_i)(x_1 - x_2)$*. Then, with a* $K$ *solved with* (3.23)*, the system with pre-feedback* $\dot{x} = f(x) + g(x)(u + Kx)$ *is incrementally stable within* $\mathcal{X} \cap \{x|x^\intercal P x \leq \min_{i=1,...,N} \frac{u_i^{\max}}{\sqrt{K_i P^{-1} K_i^\intercal}}\}$.

*Proof.* Since the $A$ and $B$ matrix enters linearly into the Lyapunov condition in (3.22), by the assumption that $f(x_1)+g(x_1)u-f(x_2)-g(x_2)u \in Conv(A_i)(x_1-x_2)$, convexity shows that $\dot{V}(x_1-x_2) \leq 0$, which shows incremental stability. □

The CBF condition shown in Equation (3.21) is shown for a specific uncertainty set $\mathbf{x_0} = R(x_0, \Delta_t)$ that arises from the sampled-data nature of the system. For an incrementally stable dynamic system, the CBF condition is rewritten as

$$\begin{aligned} &\frac{dh}{dx}\bigg|_{\phi_\tau^{\overline{u_B}(\cdot)}(x_0)\big|_{\mathbf{x_0}}} \frac{\partial \phi_\tau^{\overline{u_B}(\cdot)}}{\partial x}\bigg|_{\mathbf{x_0}} (f(\mathbf{x_0}) + g(\mathbf{x_0})u) + \alpha(h(\phi_\tau^{\overline{u_B}(\cdot)}(x)\big|_{\mathbf{x_0}})) \\ &\frac{dh_B}{dx}\bigg|_{\phi_T^{\overline{u_B}(\cdot)}(x_0)\big|_{\mathbf{x_0}}} \frac{\partial \phi_\tau^{\overline{u_B}(\cdot)}}{\partial x}\bigg|_{\mathbf{x_0}} (f(\mathbf{x_0}) + g(\mathbf{x_0})u) + \alpha(h_B(\phi_T^{\overline{u_B}(\cdot)}(x)\big|_{\mathbf{x_0}})). \end{aligned} \tag{3.24}$$

The new set in which the constraint is being evaluated is $\mathbf{x_0} := \mathcal{R}(x_0, \Delta_t) + \mathbf{\Delta}_x$, where $\mathbf{\Delta}_x \subset \mathbb{R}^n$ is the state uncertainty set such that the estimated value of the state $\tilde{x} \in x + \mathbf{\Delta}_x$, with $x$ being the true state. The other major difference from Equation (3.21) is that the flow over the backup trajectory is now being computed for the nominal value of $x_0$, and it is simply being evaluated over the set $\phi_t^u(x_0) + \mathbf{\Delta}_x$. This greatly simplifies the computation, and makes the constraint tractable in real-time.

**Theorem 10.** *Let $\overline{u}(\cdot)$ be a input signal with zero-order hold that satisfies* (3.24)*. If the system is incrementally stable in $\overline{\mathcal{S}_I}$, then $\phi_t^{\overline{u}(\cdot)}(x_0) \in \overline{\mathcal{S}_I}$ for all $t \geq 0$.*

*Proof.* From incremental stability, we have $\forall\, x_1, x_2 \in \overline{\mathcal{S}_I}$, and for $\beta : \mathbb{R}\times[0,T] \to \mathbb{R}$ nonincreasing in $t$, and $\forall\, t_1, t_2 \in \mathbb{R}_+$ with $t_2 > t_1$,

$$\begin{aligned} &\left\|\phi_t^{u(\cdot)}(x_1) - \phi_t^{u(\cdot)}(x_2)\right\| \leq \beta(\|x_1 - x_2\|, t) \\ \Downarrow \\ &\left\|\phi_{t_2}^{u(\cdot)}(x_1) - \phi_{t_2}^{u(\cdot)}(x_2)\right\| \leq \left\|\phi_{t_1}^{u(\cdot)}(x_1) - \phi_{t_1}^{u(\cdot)}(x_2)\right\|. \end{aligned} \tag{3.25}$$

Therefore,

$$\begin{aligned} &\left\|\phi_0^{u(\cdot)}(x_1) - \phi_0^{u(\cdot)}(x_2)\right\| = \|x_1 - x_2\| \\ \Downarrow \\ &\phi_t^{\overline{u_B}(\cdot)}(\mathbf{x_0}) \subset \phi_t^{\overline{u_B}(\cdot)}(x_0)\big|_{\mathbf{x_0}}. \end{aligned} \tag{3.26}$$

Fix any $\mathbf{x_0}$, $\bar{u}$, $t$. For brevity, let $\Phi_1 := \phi_t^{\overline{u_B}(\cdot)}(\mathbf{x_0})$ and $\Phi_2 := \phi_t^{\overline{u_B}(\cdot)}(x_0)\big|_{\mathbf{x_0}}$, and let $\mathbf{x} := \frac{\partial \phi_t^{\overline{u_B}(\cdot)}}{\partial x}\Big|_{\mathbf{x_0}} (f(\mathbf{x_0}) + g(\mathbf{x_0})u)$.

$$\Phi_1 \subset \Phi_2 \quad \Rightarrow \quad \left.\frac{dh}{dx}\right|_{\Phi_1} \subset \left.\frac{dh}{dx}\right|_{\Phi_2} \quad \Rightarrow \quad \left.\frac{dh}{dx}\right|_{\Phi_1} \mathbf{x} \subset \left.\frac{dh}{dx}\right|_{\Phi_2} \mathbf{x}.$$

Following the same logic, we have

$$\Phi_1 \subset \Phi_2 \quad \Rightarrow \quad \alpha(h(\Phi_1)) \subset \alpha(h(\Phi_2)).$$

Thus,

$$\left.\frac{dh}{dx}\right|_{\Phi_1} \mathbf{x} + \alpha(h(\Phi_1)) \subset \left.\frac{dh}{dx}\right|_{\Phi_2} \mathbf{x} + \alpha(h(\Phi_2)).$$

Therefore, any $\mathbf{x_0}, \bar{u}(\cdot)$ that meets condition (3.24) will also meet condition (3.21), and by Proposition 1, $\phi_t^{\bar{u}(\cdot)}(x_0) \in \overline{\mathcal{S}_I}$ for all $t \geq 0$ □

Now, we will extend the safety guarantees to systems with known time-delay, without simply making the barrier robust to a set of possible input delays, which would degrade system performance. First, we rely on two assumptions.

**Assumption 2.** *Suppose that the system has a time delay equal to some integer n multiple of the controller period $\Delta_t$. Therefore, the system evolves with dynamics*

$$\dot{x} = f(x) + g(x)\bar{u}(x, t - n\Delta_t) \tag{3.27}$$

*for zero-order hold controller $\bar{u}$.*

This is a reasonable assumption, especially for the time delay caused by the numerical computation of the digital controller. Moreover, rounding of the time-delay can always be made robust with an addition to the state uncertainty.

Since it is not possible to provide any input to the system before time $t = n\Delta_t$, one more assumption is required.

**Assumption 3.** *From any initial set of states $\mathbf{x_0} = x_0 + \mathbf{\Delta}_x \subset \overline{\mathcal{S}_I}$ , we require*

$$\phi^0_{n\Delta_t}(\mathbf{x_0}) \subset \overline{\mathcal{S}_I}. \tag{3.28}$$

*Here, the 0 in $\phi^0_{n\Delta_t}$ denotes the fact that a control input of zero is applied to the system during this time.*

In practice, this is not a restrictive assumption since the initial condition can be set well within the safe set. Moreover, if this is not met, there is no hope to keep the system safe whatsoever.

In order to obtain the state at which the control input will be applied, the most recent $n\Delta_t$ inputs must be stored in a vector $\bar{u}_H$. Since no input can be applied during time $t \in [0, n\Delta_t]$, the input vector is initialized to all zeros. With this, the state at which the $i$th computed control action will be applied can be expressed as

$$x_{(i+n)\Delta_t} = \phi_{n\Delta_t}^{\bar{u}_H}(x_{i\Delta_t}) \tag{3.29}$$

, The input history vector $\bar{u}_H$ is executed under zero-order hold, just as the inputs are applied to the system. Starting with the initial state, the algorithm for handling input delay is now described.

At initial time-step $t_0$, the control action to be implemented at time $t = n\Delta_t$ is computed. To keep the system safe, the barrier conditions must be evaluated at state $x_{n\Delta_t}$, which is computed using Equation (3.29). Note that the constraint itself does not need to be altered, and is still affine. The only extra step is the integration from $x_0$ to $x_{n\Delta_t}$.

The input chosen by the quadratic program at time $t_0$ is then placed at the head of the $\bar{u}_H$ buffer, after each previous value is shifted backwards. Thus, the oldest value in the input buffer is lost, as it has already taken effect on the system.

The computation for all future time-steps is outlined in Algorithm 2.

**Algorithm 2** CBF with Input Delay of $n\Delta_t$

```
u_H[n] = {0}
i ← 0
while (true) do
    x_{(i+n)Δ_t} = φ^{ū_H}_{nΔ_t}(x_{iΔ_t})
    Compute safe action with u_des, x_{(i+n)Δ_t} using (3.24)
    update u_H
    i = i + 1
end while
```

While this algorithm may seem trivial, it is only made possible by treating the system as a sampled-data system. The continuous case of this algorithm would be much more complex, as there is no finite time-history of inputs to integrate over. Safety under this algorithm is summarized with the following theorem.

**Theorem 11.** *Given a control invariant set $\overline{S_I}$, if inputs are chosen with Algorithm 2, and the system model is accurate, then the system* (3.27) *remains safe, i.e. $\phi_t^{\bar{u}}(x_0) \in \overline{S_I}$ for all $t \geq 0$.*

*Proof.* Assume by contradiction that for some time $t = m\Delta_t$, $x_{m\Delta_t} \notin \overline{\mathcal{S}_I}$. Let this be the first time in which $x \notin \overline{\mathcal{S}_I}$, thus $x_{k\Delta_t} \in \overline{\mathcal{S}_I}\ \forall k < m$.

Because system integration is accurate,

$$x_{m\Delta_t} = \phi_{n\Delta_t}^{\overline{u}_H^m}(x_{(m-n)\Delta_t}) \quad \overline{u}_H^m = [u_{m-2n+1}, ..., u_{m-n}]$$
$$x_{(m-1)\Delta_t} = \phi_{n\Delta_t}^{\overline{u}_H^{m-1}}(x_{(m-n-1)\Delta_t}) \quad \overline{u}_H^{m-1} = [u_{m-2n}, ..., u_{m-n-1}].$$

Since $x_{(m-1)\Delta_t} \in \overline{\mathcal{S}_I}$, the control inputs chosen up until time $t_{m-n-1}$ keep the system in $\overline{\mathcal{S}_I}$. Therefore, the control action $u_{m-n}$ must cause the system to exit $\overline{\mathcal{S}_I}$. The CBF condition at $t = (m-n-1)\Delta_t$ is based on $\phi_{n\Delta_t}^{\overline{u}_H^{m-1}}(x_{(m-n-1)\Delta_t})$, the estimated $x_{(m-1)\Delta_t}$, but since the model is assumed to be correct, it is equal to the actual state. However, if $u_{m-n}$ was computed via CBF condition, then $\phi_{n\Delta_t}^{\overline{u}_H^m}(x_{(m-n)\Delta_t}) \in \overline{\mathcal{S}_I}$ by Theorem 10. This implies that $x_{m\Delta_t} \neq \phi_{n\Delta_t}^{\overline{u}_H^m}(x_{(m-n)\Delta_t})$, which contradicts the assumption that the model is accurate.

□

The set $\overline{\mathcal{S}_I}$ is an example of one such control invariant set robust to zero-order hold, but this theorem holds for any other such set.

It is important to recognize the fact that, under this algorithm, one is effectively performing open-loop control over the time-horizon of the input delay. However, due to the state uncertainty result, it is possible to guarantee safety for a range of possible initial conditions that the system is expected to lay within at the time of the control input being enacted. Thus, we have the following extension:

**Corollary 2.** *Given an invariant set $\overline{\mathcal{S}_I}$, robust to state uncertainty $\mathbf{\Delta}_x$, if inputs are chosen with Algorithm 2, and $\phi_{n\Delta_t}^{\bar{u}_H}(x_0) \in x_{n\Delta t} + \mathbf{\Delta}_x$ for any $x_0 \in \overline{\mathcal{S}_I}$ (i.e., the system integration is accurate up to the set uncertainty set $\mathbf{\Delta}_x$), then the system* (3.27) *remains safe, i.e., $x(t) \in \overline{\mathcal{S}_I}$ for all $t \geq 0$.*

*Proof.* The proof follows directly from the Theorem 1, and simply utilizes the guarantees over state uncertainty from before. □

#### 3.1.2.1 Applications: Segway simulation

The simulation is done in a ROS-based simulation environment. The full, nonlinear dynamics are integrated under zero-order hold at a variable sampling time $\Delta_t$. The true state of the system is not known to the controller, only the state estimate from

an extended Kalman filter. This state observer receives noisy sensor data based on the true state of the system. A pre-feedback gain was computed following (3.22).

The Segway has 4 states $x = [p, \dot{p}, \theta, \dot{\theta}]^\intercal$. The safe set is described by $\mathcal{S} = \{x | 1 - 4p^2 \geq 0\}$, which enforces the robot position $p$ to stay within a 0.5 m range from the origin. The robust barrier condition (3.24) is evaluated over sets using the interval arithmetic library libaffa [53]. The constraint is imposed at the 10 closest points to the boundary of the safe set along the backup trajectory.

Figure 3.6 shows the result of three simulations with the nominal CBF conditions (3.5), and the robust condition (3.24). The robust condition is set to handle a state uncertainty set based on the uncertainty caused by the zero-order hold and the Kalman filter. At 40 Hz, the Segway is able to stay within the set with the nominal controller, but it is unable to maintain invariance at 20 Hz, or in the presence of an input delay of 30 ms. The robust barrier is able to maintain safety for not just the nominal trajectory, but over the entire robustness margin.

### 3.1.3 Multi-agent backup CBFs

In this subsection, we propose a backup CBF approach for multi-agent obstacle avoidance that can be implemented completely decentrally for multiple agents and scales to an arbitrary number of agents.

The core idea is to equip each agent with one or multiple backup strategies that bring the agent to an equilibrium point and check whether the corresponding backup trajectory satisfies the safety constraint. In fact, we will show later in the paper that all initial conditions whose corresponding backup trajectories satisfy the safety constraint constitute a control invariant set. Then by enforcing the CBF supervisory controller, if the backup trajectory associated with the initial condition satisfies the safety constraint, the state can be kept within the safe set indefinitely. Furthermore, we show that the CBF condition can be implemented decentrally with no communication between agents and can scale to an arbitrary number of agents. Nonetheless, communication between agents is helpful, especially in the case with multiple backup strategies, and we propose a simple broadcast scheme that to guarantee compatibility between agents.

The proposed approach bears some similarity to motion primitives [50] as the backup trajectory can be viewed as a simple motion primitive. However, the key difference is that the agent almost never execute the backup strategy. Instead, the backup strategy is used as a feasibility check to make sure that an equilibrium point

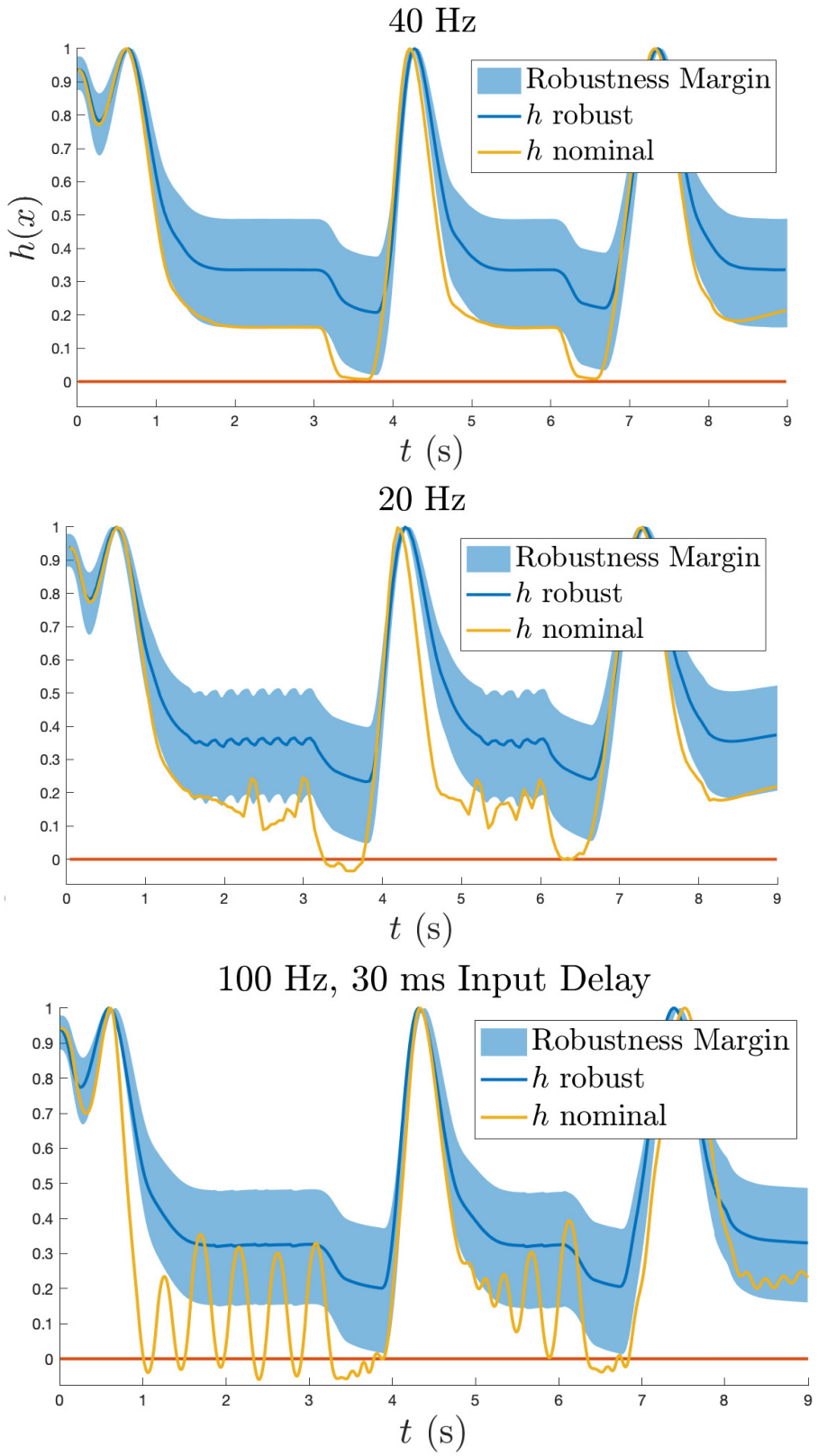



Figure 3.6: Results from simulations with three different controller frequencies, and one with input delay.

can be always be reached safely.

Here, we show how the control barrier function based on backup strategies can be applied to multi-agent obstacle avoidance. We consider a multi-agent system consisting of $N$ agents with state $x_1, ..., x_N$, respectively. The $N$ states evolve with potentially heterogeneous dynamics:

$$\dot{x}_i = f_i(x_i, u_i), x_i \in \mathbb{R}^{n_i}, u_i \in \mathcal{U}_i. \tag{3.30}$$

For the whole system, let $x = [x_1^\intercal, x_2^\intercal, ..., x_N^\intercal]^\intercal$ denote the aggregated state and the dynamics for $x$ is the following:

$$\dot{x} = f(x, u) = \begin{bmatrix} f_1(x_1, u_1)^\intercal & \dots & f_N(x_N, u_N)^\intercal \end{bmatrix}^\intercal.$$

For agent $i$, let $\pi_i : \mathbb{R}^n \to \mathcal{U}_i$ be its backup strategy and let $f_{\pi_i}(x) \doteq f_i(x_i, \pi_i(x))$ be the closed loop dynamics under $\pi_i$. The overall backup strategy given $\pi_1, \pi_2, ...\pi_N$ is then denoted as $\pi$, where $u = \pi(x) = [\pi_1(x)^\intercal, ..., \pi_N(x)^\intercal]^\intercal$.

The control barrier function constructed from a backup strategy requires an invariant set $\mathcal{S}_B$ to begin with. Although control invariant set can be difficult to compute, one control invariant set is almost free to obtain for most of the commonly seen robotic systems.

**Definition 9.** *A point $x_\pi^e \in \mathbb{R}^n$ is a stable equilibrium point for a given backup strategy $\pi$ if $f(x_\pi^e, \pi(x_\pi^e)) = 0$ and $x_\pi^e$ is stable in the sense of Lyapunov under $f_\pi$.*

We let $\mathcal{X}_\pi^e$ denote the set of all equilibrium points under $\pi$. For example, if the backup strategy $\pi$ stabilizes the steady hovering maneuver of a drone, then any steady hovering state is an equilibrium point under $\pi$. Obviously, any subset of $\mathcal{X}_\pi^e$ is a control invariant set, and $\mathcal{S}_0$ is taken as $\mathcal{X}_\pi^e \cap C$.

Given a multi-agent system as described previously, if all agents are controlled by a centralized controller, the CBF scheme should work with any backup strategy $\pi$ that result in a nonempty $\mathcal{S}_0$. However, as mentioned previously, centralized control is usually not realizable due to the communication and scalability limitations. Therefore, the focus of this paper is on a decentralized implementation of a CBF-based supervisory controller. We assume that each agent can measure the states of other agents but independently select the control input. For the proposed decentralized scheme to work, the following assumption is needed.

**Assumption 4.** *The state constraint $C$ is pairwise decomposable, i.e.,*

$$C = \{h^C(x) \geq 0\} = \{(\bigwedge_i h_i^C(x_i) \geq 0) \wedge (\bigwedge_{i \neq j} h_{ij}^C(x_i, x_j) \geq 0)\},$$

*where $h_i^C : \mathbb{R}^{n_i} \to \mathbb{R}$ is a constraint of only $x_i$, $h_{ij}^C : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \to \mathbb{R}$ is a constraint of $x_i$ and $x_j$.*

This is a reasonable assumption since for static obstacles, the obstacle avoidance constraint is on each agent; for agent-to-agent collision avoidance, the constraint is defined pairwise. Under Assumption 4, since $\mathcal{S}_0$ is taken as $\mathcal{X}_\pi^e \cap C$, and the equilibrium point is defined on the state of each agent, $\mathcal{S}_0$ is also pairwise decomposable:

$$\mathcal{S}_0 = \{h^S(x) \geq 0\} = \{(\bigwedge_i h_i^S(x_i) \geq 0) \wedge (\bigwedge_{i \neq j} h_{ij}^S(x_i, x_j) \geq 0)\}. \tag{3.31}$$

Next, we show how the CBF QP can be implemented in a decentralized structure. In a decentralized setting, the backup strategy for each agent is restricted to be a function that only depends on the agent itself, i.e., $\pi_i$ only depends on $x_i$. Under Assumption 4, the original CBF QP becomes

$$\begin{aligned}
u^\star = &\arg\min_{u \in \mathcal{U}} \left\|u - u^0\right\|^2 \\
\text{s.t. } &\forall t \in [0, T],\ \forall i \neq j \in \{1, 2, ..., N\}, \\
&\nabla h_i^C(\nabla_{x_i} \Phi_{f_{\pi_i}}^t f_i(x_i, u_i) - \partial \Phi_{f_{\pi_i}}^t / \partial t) + \alpha(h_i^C \Phi_{f_{\pi_i}}^t(x_i))) \geq 0, \\
&\nabla_{x_i} h_{ij}^C(\nabla_{x_i} \Phi_{f_{\pi_i}}^t f_i(x_i, u_i) - \partial \Phi_{f_{\pi_i}}^t / \partial t) + \\
&\nabla_{x_j} h_{ij}^C(\nabla_{x_j} \Phi_{f_{\pi_j}}^t f_i(x_j, u_j) - \partial \Phi_{f_{\pi_j}}^t / \partial t) \\
&+ \alpha(h_{ij}^C(\Phi_{f_{\pi_i}}^t(x_i), \Phi_{f_{\pi_j}}^t(x_j)) \geq 0, \\
&\nabla h_i^S(\nabla_{x_i} \Phi_{f_{\pi_i}}^T f_i(x_i, u_i) - (\partial \Phi_{f_{\pi_i}}^t / \partial t)|_{t=T}) + \alpha(h_i^S \Phi_{f_{\pi_i}}^T(x_i))) \geq 0, \\
&\nabla_{x_i} h_{ij}^S(\nabla_{x_i} \Phi_{f_{\pi_i}}^T f_i(x_i, u_i) - (\partial \Phi_{f_{\pi_i}}^T / \partial t)|_{t=T}) + \\
&\nabla_{x_j} h_{ij}^S(\nabla_{x_j} \Phi_{f_{\pi_j}}^T f_i(x_j, u_j) - (\partial \Phi_{f_{\pi_j}}^t / \partial t)|_{t=T}) \\
&+ \alpha(h_{ij}^S(\Phi_{f_{\pi_i}}^T(x_i), \Phi_{f_{\pi_j}}^T(x_j))) \geq 0.
\end{aligned} \tag{3.32}$$

If all agents follow their backup strategies, (3.32) is feasible whenever $h(x) \geq 0$. However, due to the coupling constraints between agents, this CBF QP cannot be solved decentrally. In particular, for each $\dot{h}_{ij}^C$ and $\dot{h}_{ij}^S$, the derivatives contain two parts, one determined by $\dot{x}_i$ and one by $\dot{x}_j$. To resolve this problem, notice that the

terms containing $u_i$ and $u_j$ are summed together. Therefore, with a decomposition of the CBF derivative, the CBF QP with a sufficient condition of (3.32) can be solved decentrally. For the agent $i$, the following CBF QP is solved:

$$
\begin{aligned}
u^\star = &\arg\min_{u_i \in \mathcal{U}_i} \left\| u_i - u_i^0 \right\|^2 \\
\text{s.t. } &\forall t \in [0,T],\ \nabla h_i^C(\nabla_{x_i}\Phi^t_{f_{\pi_i}} f_i(x_i,u_i) - \partial\Phi^t_{f_{\pi_i}}/\partial t) + \alpha(h_i^C\Phi^t_{f_{\pi_i}}(x_i))) \geq 0, \\
&\forall j \neq i, \nabla_{x_i} h_{ij}^C(\nabla_{x_i}\Phi^t_{f_{\pi_i}} f_i(x_i,u_i) - \partial\Phi^t_{f_{\pi_i}}/\partial t) \\
&+ 0.5\alpha(h_{ij}^C(\Phi^t_{f_{\pi_i}}(x_i), \Phi^t_{f_{\pi_j}}(x_j)) \geq 0, \\
&\nabla h_i^S(\nabla_{x_i}\Phi^T_{f_{\pi_i}} f_i(x_i,u_i) - (\partial\Phi^t_{f_{\pi_i}}/\partial t)|_{t=T}) + \alpha(h_i^S\Phi^T_{f_{\pi_i}}(x_i))) \geq 0, \\
&\nabla_{x_i} h_{ij}^S(\nabla_{x_i}\Phi^T_{f_{\pi_i}} f_i(x_i,u_i) - (\partial\Phi^T_{f_{\pi_i}}/\partial t)|_{t=T}) + \\
&0.5\alpha(h_{ij}^S(\Phi^T_{f_{\pi_i}}(x_i), \Phi^T_{f_{\pi_j}}(x_j))) \geq 0,
\end{aligned}
\tag{3.33}
$$

where $u_i^0$ is the desired input for agent $i$ from the legacy controller. Note that this optimization only depends on information of agent $i$ and $\Phi^t_{\pi_j}(x_j)$, i.e., the backup trajectories of other agents. Since we assume that each agent can measure the state of other agents, if $\pi_j$ is known a priori, $\Phi^t_{\pi_j}(x_j)$ can be solved by agent $i$ by a simple integration scheme.

**Theorem 12.** *For all $x \in \{x|h(x) \geq 0\}$,* (3.33) *is always feasible for every agent; and when each agent implement the supervisory controller in* (3.33)*, the solution $[u_1^\star; ...u_N^\star]$ is a feasible solution to* (3.32) *(not necessarily the optimal solution).*

*Proof.* The feasibility comes from the fact that $u_i = \pi_i(x_i)$ is a feasible solution. Given $[u_1^\star; ...u_N^\star]$ as the solutions to (3.33) for each agent, for each $i \neq j, \forall t \in [0,T]$, we have

$$
\begin{aligned}
&\frac{dh_{ij}^C(\Phi^t_{f_{\pi_i}}(x_i), \Phi^t_{f_{\pi_j}}(x_j))}{dt} \\
=&\nabla_{x_i} h_{ij}^C(\nabla_{x_i}\Phi^t_{f_{\pi_i}} f_i(x_i,u_i) - \partial\Phi^t_{f_{\pi_i}}/\partial t) \\
+&\nabla_{x_j} h_{ij}^C(\nabla_{x_j}\Phi^t_{f_{\pi_j}} f_i(x_j,u_j) - \partial\Phi^t_{f_{\pi_j}}/\partial t) \\
\geq& -0.5\alpha(h_{ij}^C(\Phi^t_{f_{\pi_i}}(x_i), \Phi^t_{f_{\pi_j}}(x_j)) \times 2 \\
=& -\alpha(h_{ij}^C(\Phi^t_{f_{\pi_i}}(x_i), \Phi^t_{f_{\pi_j}}(x_j)) \geq 0.
\end{aligned}
$$

The same is true for the constraints on $h_{ij}^S$, therefore $[u_1^\star; ...u_N^\star]$ is a feasible solution to (3.32). □

To conclude, the proposed decentralized CBF supervisory controller begins by selecting a backup strategy for each agent in the system that brings the agent to a stable equilibrium point under the backup strategy. The backup strategies for all agents are known a priori to every agent as part of the centralized design. Then each agent measures the state of the adjacent agents (agents that are far away do not pose any danger of collision) and makes sure that if other agents execute the backup strategy, its own backup strategy would avoid collision with both the static and other agents. This is achieved by every agent solving (3.33) decentrally. We show that the decentralized CBF QP is always feasible when the CBF $h(x) \geq 0$. Furthermore, since the computation only depends on the states of adjacent agents, whose number is bounded (due to the clearance requirement), the algorithm can scale to an arbitrary number of agents.

#### 3.1.3.1 CBF with multiple backup strategies

The previous strategy guarantees obstacle avoidance for a multi-agent system, but the mobility of the system may be compromised for safety. Since the CBF intervention is based on the backup strategy, one natural way to increase mobility is to equip the agents with multiple backup strategies and the CBF condition only need to hold for one of the backup strategies. However, we show that this is not always implementable, especially in the cases without communication. We present a simple broadcast scheme that enables the implementation of CBF controllers with multiple backup strategies for each agent.

Let $m_i$ denote the number of backup strategies for agent $i$ and let $\pi_i^k$ denote the $k$-th backup strategy for agent $i$. Given $x_i$ and $x_j$, we say that $\pi_i^k$ and $\pi_j^l$ are two compatible backup strategies for agent $i$ and $j$ if $\forall t \in [0, T]$,

$$h_i^C(\Phi^t_{\pi_i^k}(x_i)) \geq 0,\ h_j^C(\Phi^t_{\pi_j^l}(x_j)) \geq 0,\ h_{ij}^C(\Phi^t_{\pi_i^k}(x_i), \Phi^t_{\pi_j^l}(x_j)) \geq 0,$$
$$h_i^S(\Phi^T_{\pi_i^k}(x_i)) \geq 0,\ h_j^S(\Phi^T_{\pi_j^l}(x_j)) \geq 0,\ h_{ij}^S(\Phi^T_{\pi_i^k}(x_i), \Phi^T_{\pi_j^l}(x_j)) \geq 0,$$

that is, if the backup trajectories of agent $i$ and $j$ under $\pi_i^k$ and $\pi_j^l$ satisfy the state constraint and terminal constraint.

With multiple backup strategies for each agent, a choice of backup strategies for the whole multi-agent system $\{\pi_i^{k_i}\}_{i=1}^N$ is a feasible backup strategy if for each agent pair $i$ and $j$, $\pi_i^{k_i}$ and $\pi_j^{k_j}$ are compatible.

Unfortunately, decentralized CBF with multiple backup strategies without communication between agents is in general not implementable. Consider the situation

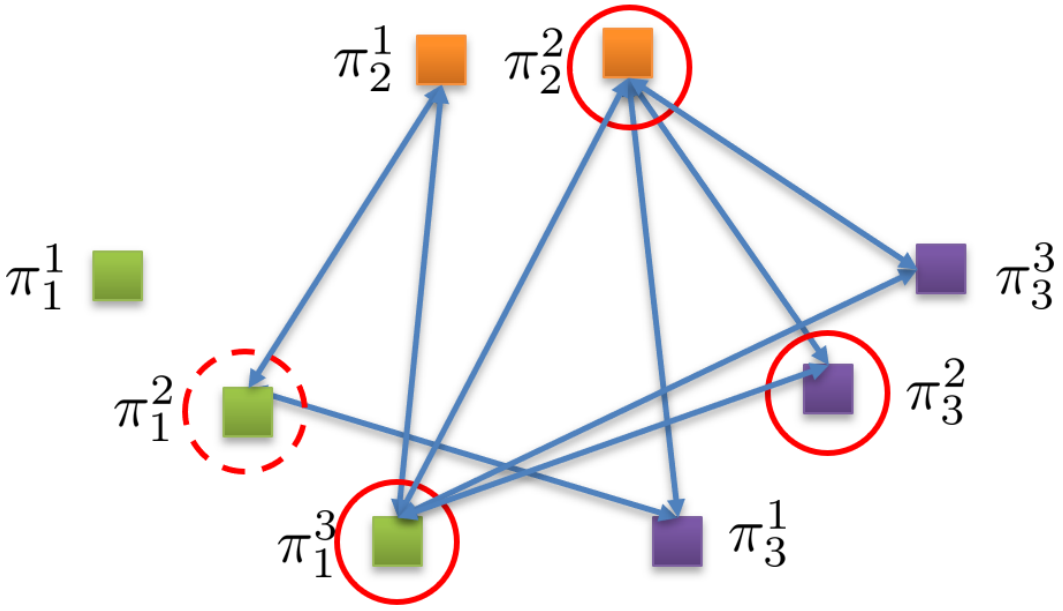



Figure 3.7: Compatibility of multiple backup strategies

depicted in Fig. 3.7 consisting of 3 agents with 3, 2, and 3 backup strategies. Each line indicates that the two backup strategies it links are compatible. For the whole system, $(\pi_1^3, \pi_2^2, \pi_3^2)$ and $(\pi_1^3, \pi_2^2, \pi_3^3)$ are the two feasible backup strategies. However, agent 1 would not be able to tell that $\pi_1^2$ is not a valid choice without the information about the compatibility between the backup strategies of agent 2 and 3.

To resolve this problem, we propose a broadcast scheme in which each agent broadcast its currently selected backup strategy and use the information about other agents' selected backup strategies to determine whether it can change its current backup strategy. For example, in the situation depicted in Fig. 3.7, if the circled backup strategies are selected and broadcasted by the agents, agent 3 can switch from $\pi_3^2$ to $\pi_3^3$ since it is able to determine that $\pi_3^3$ is also compatible with the backup strategies selected by other agents. Let $\pi_i^s$ denote the selected backup strategy for agent $i$, and based on the selected backup strategies of other agents, agent $i$ is able to determine the set of all backup strategies that is compatible with other agents in the system, we denote this set as $\Pi_i^a$, the active backup strategy set for agent $i$.

The CBF QP is solved for every backup strategy in $\Pi_i^a$ and the one with the smallest intervention is selected and broadcasted to other agents in the system. The input corresponding to the selected backup strategy is then taken as the input $u_i$ of agent $i$, shown in Algorithm 3.

#### 3.1.3.2 Applications: Dubin's car

In this section, we present the application of the proposed algorithm on a Dubin's car example and a quadrotor example.

**Dubin's car** We consider a simple Dubin's car example with the following dynamics:

$$\begin{bmatrix} \dot{X} & \dot{Y} & \dot{v} & \dot{\theta} \end{bmatrix}^\intercal = \begin{bmatrix} v\cos(\theta) & v\sin(\theta) & a & r \end{bmatrix}^\intercal, \tag{3.34}$$

where $X$, $Y$, $v$, $\theta$ are the longitudinal and lateral coordinates, the velocity, and the heading angle. The inputs are acceleration $a$ and yaw rate $r$, and are bounded by $a^{\max}$ and $r^{\max}$. This is a valid model for differentially driven ground robots such as the ones in the Robotarium of Georgia Institute of Technology [116]. We first consider the simple case with only one backup strategy for each robot. In this case, the backup strategy is simply to brake until full stop. In practice, the expression of the CBF in (3.33) is not implementable since there are uncountably many $t$ in $[0, T]$. We replace the continuous spectrum $[0, T]$ with a finite time sequence $0 = t_0 < t_1 <, ..., < t_M = T$ and enforce the CBF condition on these time instances instead. This finite sampling and the finite update rate of the CBF controller call for additional robustness of the control strategy. We proved robust safety under time discretization and the finite sampling of the backup trajectory in [140], check the result therein for detail. The backup trajectory and the sensitivity matrices are computed by solving the corresponding ODEs.

We conduct experiment in the Robotarium environment with 3 and 6 robots and the goal for each robot is to patrol between two way points. The legacy controller $u^0$ is a simple greedy linear controller that tries to bring the robot to the destination without

**Algorithm 3** CBF QP with multiple backup strategies

**procedure** CBF-QP( $\pi^s_{1:N}$, $u^0_i$, $\mathcal{U}_i$)
  Compute $\Pi^a_i$, set of all compatible backup strategies with $\pi^s_{1:N}$
  **for** $\pi^k_i$ in $\Pi^a_i$ **do**
    Solve (3.33) with $\pi^k_i$ and $u^0_i$, and obtain $u^\star_{i,\pi^k_i}$
  **end for**
  $u_i = \min_{\pi^k_i \in \Pi^a_i} u^\star_{i,\pi^k_i}$
  $\pi^s_i = \arg\min_{\pi^k_i \in \Pi^a_i} u^\star_{i,\pi^k_i}$
**end procedure**

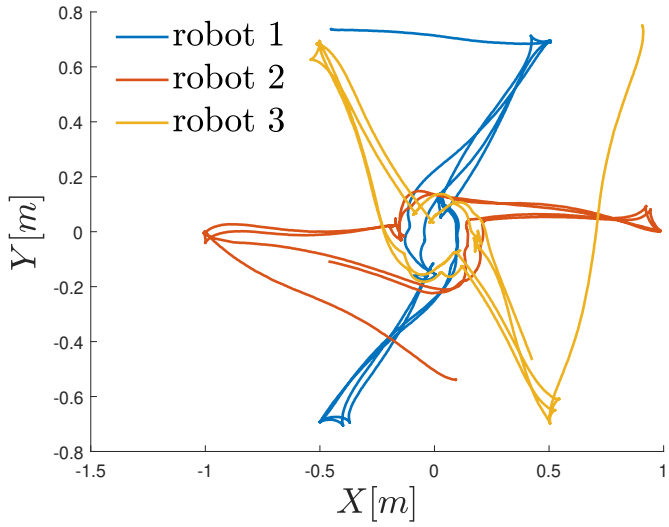



Figure 3.8: Robot traces of the Robotarium experiment with 3 robots. Each robot is asked to patrol between 2 positions. The CBF controller guarantees zero collision.

any knowledge of other robots. The CBF keeps the robot within the boundary (static state constraints) and avoids any collision with other robots in the system.

Fig. 3.8 shows the traces of the 3 robots where they meet at the center of the state space and rotate to make ways for each other until they can move towards their destinations. Note that the seemingly coordinated behavior is actually the result of a decentralized control structure. With the same setup, we conducted experiment with 6 robots in the Robotarium, and the result shows that the CBF is able to guarantee no collision between the robots. The video of the experiments and the simulations can be found at https://youtu.be/RqsCvHBjf88. The differences between this experiment and the control barrier function approach in [158] are (1) acceleration rather than speed is used as control input and the CBF QP is guaranteed to be feasible under torque limit (2) the proposed CBF approach is implemented decentrally where each robot solves for the safe input without communication with other robots.

We also tested the case in which each robot is equipped with 3 backup strategies, where $\pi^1$ is to simply break, $\pi^2$ is to break and turn right, and $\pi^3$ is to break and turn left. Fig. 3.9 shows the state trajectories of 2 robots performing a similar surveillance task controlled under the CBF with 3 backup strategies and the broadcasting scheme from before. Different colors were used to mark the segments of the trajectory during which different backup strategies were chosen. When the two robots swerve and avoid each other, $\pi^2$ and $\pi^3$ are selected so that the intervention needed is

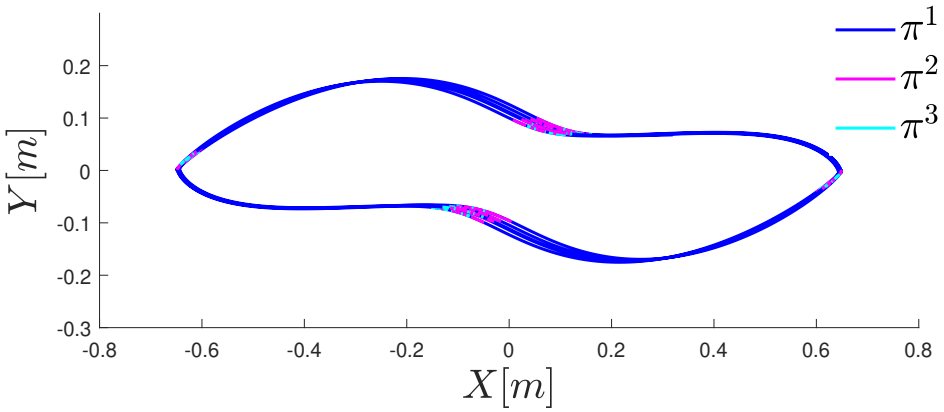



Figure 3.9: Traces of 2 robots equipped with 3 backup strategies and broadcasting their current backup strategy. Colors show the selected backup strategy in the CBF QP.

minimized.

#### 3.1.3.3 Applications: Two quadrotors in simulation

To showcase the scalability of this method, we now consider a 17-dimensional quadrotor model. The state vector $x = [\mathbf{r}, \mathbf{v}, \mathbf{q}, \mathbf{w}, \mathbf{\Omega}]^\intercal$ where $\mathbf{r}$ is the position $[x, y, z]^\intercal$ in $\mathbb{R}^3$, $\mathbf{v}$ is the velocity $[v_x, v_y, v_z]^\intercal$ in the world frame, $\mathbf{q}$ is the quaternion $[q_w, q_x, q_y, q_z]^\intercal$, $\mathbf{w}$ is the angular velocity vector $[w_x, w_y, w_z]^\intercal$ in the body frame, and $\mathbf{\Omega}$ is the vector of angular velocities of the propellers, $[\Omega_1, \Omega_2, \Omega_3, \Omega_4]^\intercal$. The control input is the voltages applied at the motors $u = [V_1, V_2, V_3, V_4]^\intercal$.

The dynamics are derived from force-balance equations in a rotating frame, as well as a first order motor model. The gradients of the dynamics w.r.t. the state are computed symbolically. The resulting symbolic expressions for $f(x)$ and $\frac{\partial f(x,u)}{\partial x}$, and $\frac{\partial f(x,u)}{\partial u}$ are then exported to C++ using code generation from Matlab for the simulation environment.

The backup policy aims to simply stop the quadrotor and set the pitch and roll angles to zero, which is achieved through simple PD controllers around the linear velocities, angular rates, and angles. The gradients of these dynamics with respect to the state are also computed symbolically and exported to C++. The corresponding backup set is a small ball around linear velocity, pitch, and roll angles equal to 0.

For each quadrotor, the barrier function $h(x)$ seeks to avoid a ball of radius $r$ around

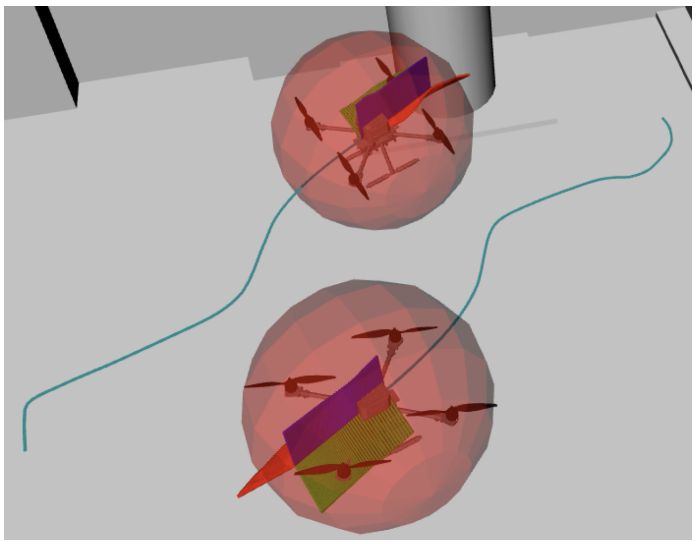


Figure 3.10: Swerving maneuvers of the drones under the CBF controller when commanded to fly at each other.

the closest point on the other quadrotor's backup trajectory ($[x_c, y_c, z_c]^\intercal$), giving

$$h(x) = (x - x_c)^2 + (x - x_c)^2 + (x - x_c)^2 - r^2.$$

The final expression needed is the gradient of this function $\nabla h$, which can be derived trivially.

The simulation environment is based on ROS and is written in C++. The code can be found at https://github.com/DrewSingletary/uav_sim_ros, including the Matlab dynamics and C++ code generation. The solver used was the OSQP solver [145], and the nominal controller tracked linear velocity and yaw rate commands using the same PD control strategy as the backup controller. Fig. 3.10 shows a snapshot of the simulation, when the two drones are sent directly at each other.

## 3.2 Gradient-free backup CBFs

While the backup controller CBF formulation is applicable for a wide variety of systems, it is too computationally expensive to run on microcontrollers, such as those used in flight hardware. This is due primarily to the expensive computations of $\frac{\partial \Phi}{\partial x}$ needed for the CBF condition. In this section, we introduce an alternative approach that avoids such gradient computations.

### 3.2.1 Safety regulator formulation

As in the previous section, we consider the implicit safe set:

$$\mathcal{S}_I \triangleq \left\{x \in X \mid \left(\phi_T^\pi(x) \in \mathcal{S}_B\right) \wedge \left(\forall\, t \in [0,T]\,,\ \phi_t^\pi\,(x) \in \mathcal{S}\right)\right\} \tag{3.35}$$

for backup policy $\pi$, safe set $\mathcal{S}$ described by $h(x)$, and backup set $\mathcal{S}_B$ described by $h_B(x)$.

Assuming $h$ and $h_B$ are continuously differentiable, we can express the implicit safe set as

$$\mathcal{S}_I = \left\{x \in \mathcal{S} \mid \min_{t\in[0,T]} h \circ \phi_t^{u_b}\,(x) \geq 0 \wedge h_B \circ \phi_T^{u_b}(x) \geq 0\right\}. \tag{3.36}$$

Therefore:

$$h_I(x) \triangleq \min_{t\in[0,T]} \left\{h \circ \phi_t^{u_b}\,(x)\,,\ h_B \circ \phi_T^{u_b}(x)\right\} \tag{3.37}$$

is a control barrier function and one can define a filtering policy that guarantees set invariance of $\mathcal{S}_I$, and therefore $\mathcal{S}$ as well.

To enforce safety without the use of the standard CBF condition on the implicit safe set, we propose the following method.

**Theorem 13.** *Given a smooth function $k : \mathcal{D} \times \mathbb{R} \to U$, a control law defined by*

$$u(x) = k\left(x, h_T^{\mathcal{Q}}(x)\right), \tag{3.38}$$

*regulates solutions of* (1.1) *to stay in $S_I \subseteq \mathcal{S}$ for all time if for all $x \in \mathcal{S}_I$:*

$$k\,(x, 0) = \pi(x). \tag{3.39}$$

*Proof.* By definition, $S_I$ is invariant under the control law $\pi(x)$. Therefore, $\forall x_0 \in S_I, \forall t \geq t_0, \Phi_\pi(x_0, t) \in S_I$. By continuity of the flow operator, we know that $\Phi_k$ is continuous, and $h_I$ is continuous since it is the composition of continuous functions $h$ and $h_B$ and the flow $\Phi_k$. Therefore, the state must pass through $h_I(x) = 0$ before exiting $S_I$. Without loss of generality, label any such point $x_{h_0}$. Since $f_k = f_\pi$ at such a point, the system will remain in $S_I$ for all time, as $x_{h_0} \in \mathcal{S}_I \implies \Phi_\pi(x_0, t) \in S_I, \forall t \geq t_{h_0}$. □

More than just safety, however, we also have a smoothness result on the control law.

**Theorem 14.** *If $k(\cdot,\cdot)$ is locally Lipschitz in its arguments, and the dynamics under the backup controller $f_\pi$ are continuous and bounded on $\mathcal{S}$, then the resulting filter $k\,(x, h_I(x))$ is locally Lipchitz continuous.*

*Proof.* By the definition of a CBF, $h(x)$ and $h_B(x)$ are differentiable. Moreover, the flow of the system $\Phi_\pi(x,t)$ is differentiable for continuous dynamics and backup controller by the second fundamental theorem of calculus, and therefore locally Lipschitz (since it is also bounded). Since Lipschitz continuity is preserved under the min operator, we have that $h_I(x)$ is locally Lipschitz continuous. Lastly, since Lipschitz continuity is preserved under compositions, we have that $k\left(x, h_I(x)\right)$ is locally Lipschitz continuous. □

The two requirements for our filtering function is that (i) the backup controller $\pi(x)$ is applied when $h_I(x) = 0$, (ii) it is Lipschitz continuous in its arguments. When $h_I(x) > 0$, we want the filter to mimic $u_{\text{des}}(x)$ as much as possible. To achieve this, we first choose the mixing function

$$k\left(x, h_I(x)\right) = \lambda\left(x, h_I(x)\right) u_{\text{des}}(x) + (1 - \lambda(x, h_I(x)))\pi(x), \tag{3.40}$$

for $\lambda(x, h_I(x)) : \mathbb{R}^n \times \mathbb{R} \to [0,1]$. Figure 3.11 illustrates how this mixing works. When $\lambda(x, h_I(x)) = 1$, the human operator is in complete control. As the drone approaches the wall, the value of $\lambda(x, h_I(x))$ decreases, and it begins to slow down. Finally, near the boundary, a steady-state is reached between the operator's control action and the backup control action, and the drone stops. It is important that the backup controller attempts to move away from the boundary, so that the system does not get "stuck" near the boundary, i.e., $\lambda > 0$ always.

To allow for maximum freedom of the operator, the function should exactly match $u_{\text{des}}(x)$ when $h_I(x) \gg 0$. One way to achieve this is by exploiting the exponential function:

$$\lambda(x, h_I(x)) = 1 - \exp\left(-\beta h_I^+(x)\right), \tag{3.41}$$

where constant $\beta$ is used to tune how quickly the function $\lambda(x, h_I(x))$ decays, and $h_I^+(x) = \max(h_I(x), 0)$ ensures that $\lambda(x, h_I(x)) \in [0,1]$. With this filtering controller, we get the desired behavior of $\lambda(x, h_I) \approx 1$ when $h_I(x) \gg 0$, while providing a smooth decay to 0 when as $h_I(x) \to 0$. The constant $\beta$ is used to tune how quickly the function $\lambda$ decays.

### 3.2.2 Comparison to backup controller CBF

To demonstrate the effectiveness of this method, we compare its performance to the backup set CBF approach [59]. The two relevant metrics for this comparison are the computational times and the conservatism, as both methods provide guarantees of safety.

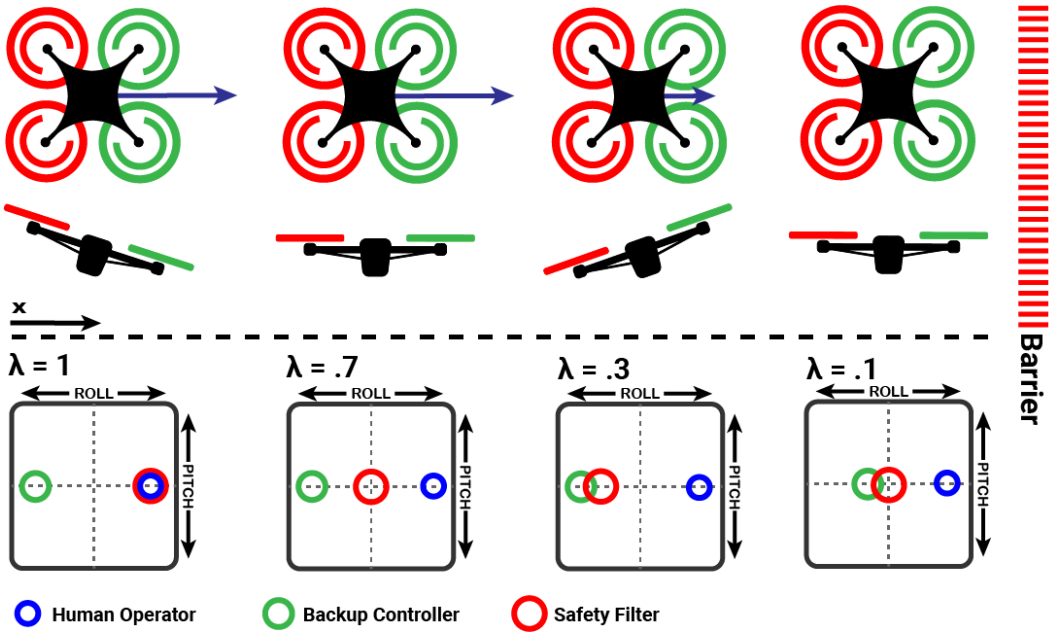



Figure 3.11: As the drone approaches the barrier, $\lambda$ decreases, resulting in the backup controller being utilized more.

While no perfect comparison can be made, due to the freedom in the selection of both $\lambda(x)$ and $k(x)$, these functions are chosen independently to result in smooth transitions when approaching the boundary of the set, while minimizing conservatism.

**Example 7.** *Consider an inverted pendulum with state x dynamics* $\dot{x}$

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \quad \dot{x} = \begin{bmatrix} \dot{\theta} \\ \sin(\theta) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \tag{3.42}$$

*and backup control law*

$$\pi(x) = -Fx, \tag{3.43}$$

*which attempts to stabilize the system to the backup set*

$$h_B(x) = \min\left\{\left(\frac{\pi}{12}\right)^2 - x_1^2, \delta^2 - x_2^2\right\}, \tag{3.44}$$

*for a small velocity value $\delta$ chosen to be 0.1 rad/s, while staying in the set*

$$h(x) = \min\left\{1 - x_1^2, 2 - x_2^2\right\}. \tag{3.45}$$

For the exact formulation of the backup-set CBF, see [36]. The filtering function used here is the same as that used on the drone (3.40), to be detailed in the following

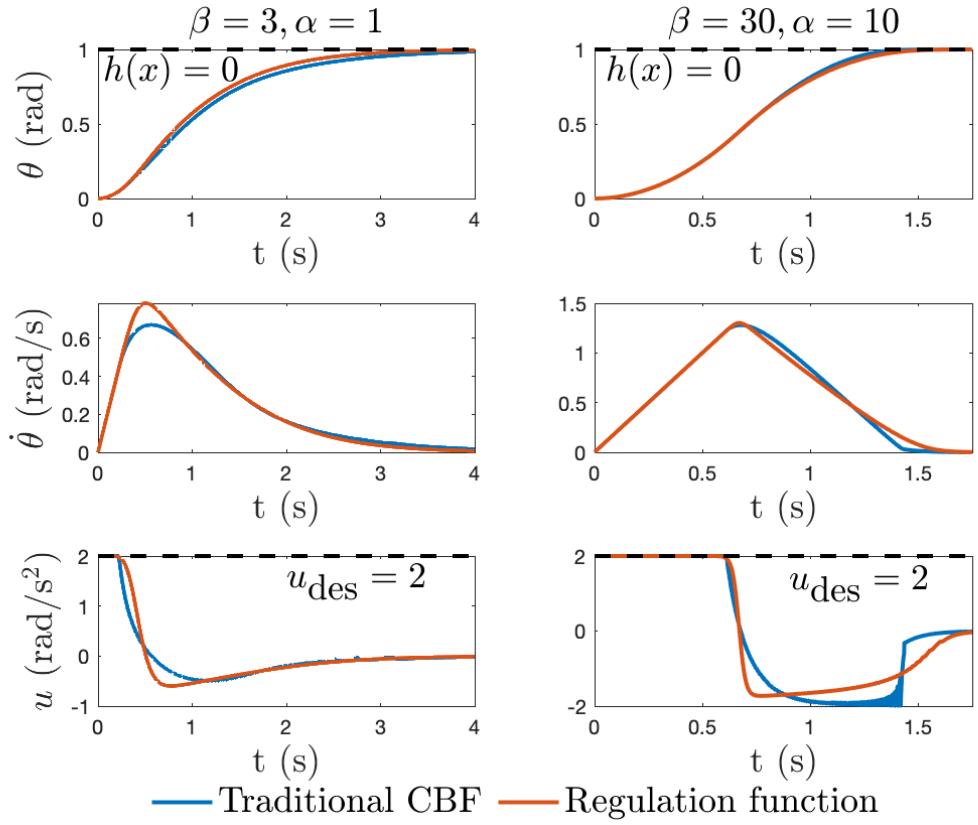



Figure 3.12: The filtering performance of the traditional CBF compared to the proposed regulation function, for two parameters $\beta$ in (3.41) and the scalar $\alpha$ for the CBF (2.9).

section. The inverted pendulum was commanded a constant angular acceleration of 2 rad/s$^2$, and the resulting positions, velocities, and filtered inputs are displayed in Figure 3.12.

Two benefits of the smooth filter can be seen in this comparison. While filtering performance is similar, the regulation function is an order of magnitude faster to evaluate. Moreover, when the gains are increased to allow a rapid approach of the boundary of the safe set, the CBF begins to oscillate near the boundary, whereas the smooth filter does not suffer from such behavior. These oscillations occur due to numerical instability of the optimization problem as the system pushes against boundary of the safe set.

It is important to note that the optimization-based CBF still has a distinct advantage in some situations. Utilizing gradient information allows quick motion along the boundary of the set, whereas with this switching approach, the value of $\lambda(x, h_I(x))$ will be low, limiting performance near the boundary. However, the following

section will partially remedy this issue through the use of various time-varying backup control policies.

### 3.2.3 Applications: Collision avoidance

The most prevalent uses for drones, including deliveries, exploration, environmental monitoring, and more, involve navigating through unknown or uncertain environments. Due to the altitude of the vehicles and their often exposed propellers, collisions are catastrophic for the drone and might also be dangerous for its surroundings. For this reason, collision avoidance techniques are crucial to further the use of these systems in everyday life.

In typical drone flight, collision avoidance is the process of creating and tracking trajectories that take the drone through the surrounding free space and avoid occupied or uncertain space. While this approach to collision avoidance can be effective in practice, as evidenced in [98, 97, 68], it is typically quite conservative and leads to slow mobility, or lacks guarantees of collision-free paths. The conservative aspect of these planners stems from two major hurdles: the computational complexity of the planners that necessitate simplified abstractions of the model and obstacles, and uncertainty in the mapped environment.

Planning in uncertain environments requires frequent updates to the planned trajectories as new information is gained. It is intractable to plan feasible trajectories for the true dynamics of these aerial vehicles in such short time, so traditionally, a global planner with no regard for the system dynamics creates a rough path to follow, and the local planner uses this as a guide to create more realistic, shorter trajectories that can be tracked. Even the trajectories generated by the local planner do not account for the full nonlinear dynamics, as this would require solving a large nonlinear constrained optimization problem, but are instead generated with other assumptions that approximate dynamically feasible paths to various degrees, such as triple integrator models with jerk-limited trajectories [151, 90] or linearizations [47].

This subsection showcases the application of the gradient-free CBF to collision avoidance for aerial vehicles that enables high-speed flight in uncertain environments. This offers an alternative to the planning-based approaches mentioned previously, as it directly modifies the desired velocities at the rate of the flight controller, using real-time sensor data.

The simulation environment is shown in Figure 3.13. We utilize this environment

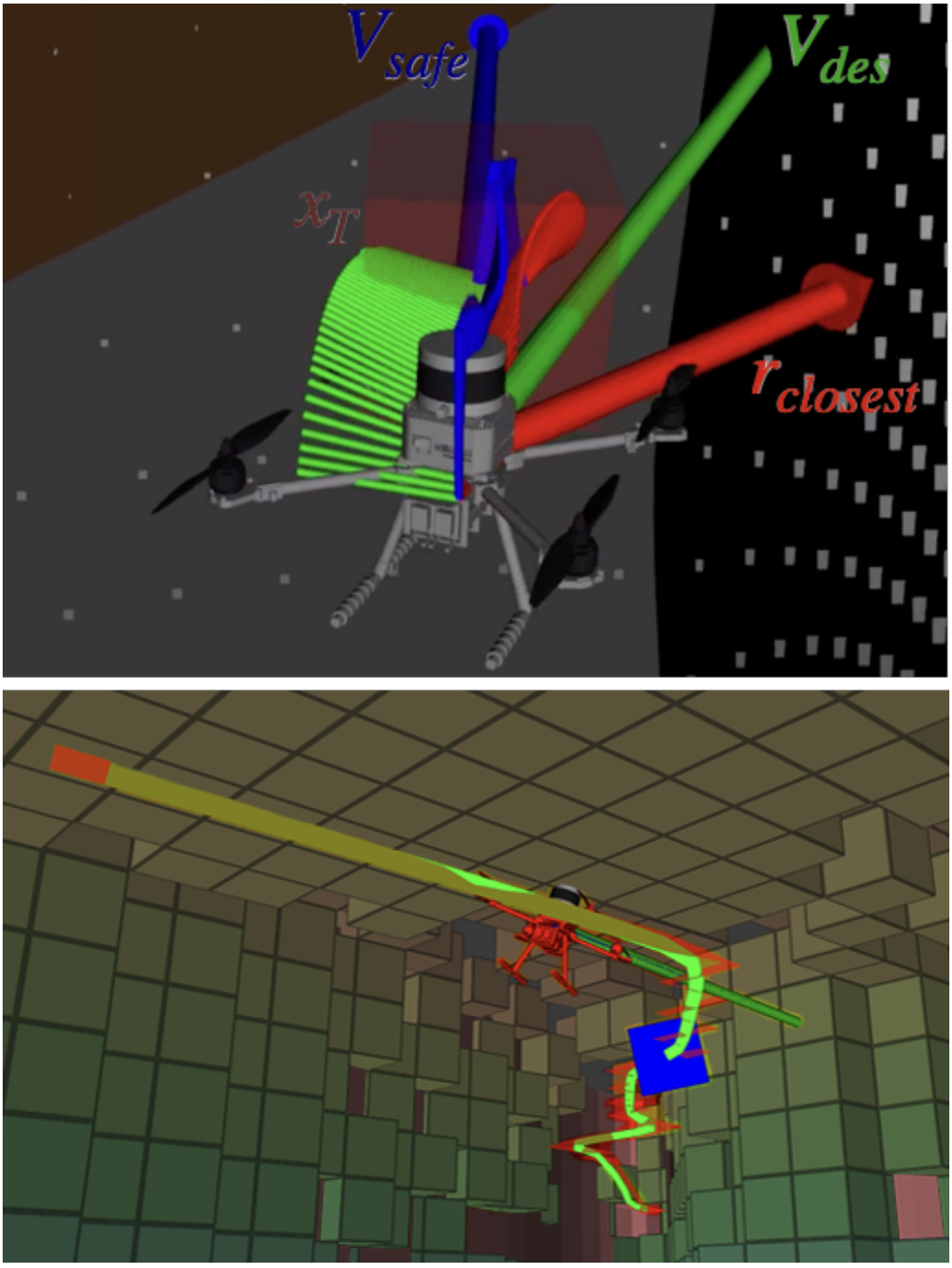



Figure 3.13: Simulation environment. The top shows the desired and filtered velocity commands based on the closest point in the point cloud. The bottom shows the drone navigating through the cave.

to showcase the method working with a simple planner to explore a large 240m by 460m cave system. The model used for the demonstration is a 16-dimensional, nonlinear model of a quadrotor with voltage inputs, to be as realistic to the actual system as possible.

A standard 12-dimensional model for a quadrotor is first obtained from force-balance equations in a rotating reference frame (e.g. [100, 171]). Let the 12-dimensional state be $\mathbf{x} = [\mathbf{r}, \mathbf{v}, \xi, \omega]^\intercal$, where $\mathbf{r}$ and $\mathbf{v}$ are position and velocity in $\mathbb{R}^3$, $\xi = [\phi, \theta, \psi]^\intercal$ are roll, pitch, and yaw angles, and $\omega \in \mathbb{R}^3$ are angular velocities in the quadrotor body frame. Then

$$M\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ -Mg \end{bmatrix} + F_z R(\xi) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{3.46a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\xi = T(\xi)\omega, \tag{3.46b}$$

$$J\frac{\mathrm{d}}{\mathrm{d}t}\omega = \tau - (\omega \times (J\omega)). \tag{3.46c}$$

Here $R(\xi)$ is the x-y-z rotation matrix from a body-fixed frame to the world frame, and $T$ the resulting mapping between angular velocities:

$$\begin{aligned} R(\xi) &= R_z(\psi)R_y(\theta)R_x(\phi), \\ T(\xi) &= \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sec(\theta)\sin(\phi) & \cos(\phi)\sec(\theta) \end{bmatrix}. \end{aligned} \tag{3.47}$$

The vertical force and angular torques acting on the body are obtained from motor angular velocities

$$F_z = k_f\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right), \tag{3.48a}$$

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} -lk_f & -lk_f & lk_f & lk_f \\ -lk_f & lk_f & lk_f & -lk_f \\ -k_t & k_t & -k_t & k_t \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \tag{3.48b}$$

with $l = \frac{D}{2\sqrt{2}}$ and $D$ the frame diameter. Since the angular velocities of the propellers cannot be controlled directly, their response to a voltage input $V_i$ must be modeled. The equation of motion for the angular velocity $\Omega_i$ of each motor is

$$(J_{rot} + J_{prop})\dot{\Omega}_i = \frac{1}{K_v R}\left(V_i - \frac{\Omega_i}{K_v}\right) - k_t\Omega_i^2. \tag{3.49}$$

Disturbances enter the system through external forces acting on the center of mass of the vehicle.

The attitude controller is simple cascade PD controller on the angles and angular rates of the quadrotor. The motor voltages at the output of the velocity controller are given by

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} u_z - u_\phi - u_\theta - u_\psi \\ u_z - u_\phi + u_\theta + u_\psi \\ u_z - u_\phi + u_\theta - u_\psi \\ u_z - u_\phi - u_\theta + u_\psi \end{bmatrix}, \tag{3.50}$$

where

$$u_z = K_{vz}(v_{z,\text{des}} - v_z) + u_{\text{hover}}(\phi, \theta) \tag{3.51}$$

$$u_\phi = K_\phi(K_v(v_{y,\text{des}} - v_y) - \phi) - k_\phi \omega_x \tag{3.52}$$

$$u_\theta = K_\theta(K_v(v_{x,\text{des}} - v_x) - \theta) - k_\theta \omega_y \tag{3.53}$$

$$u_\psi = -k_\psi(\omega_{z,\text{des}} - \omega_z). \tag{3.54}$$

Note that the upper-case $K$ gains are the proportional gains, while the lower-case $k$ gains are the derivative gains. Also, $u_{\text{hover}}(\phi, \theta)$ is the input required to maintain a constant height.

The backup policy $\pi$ is simply the velocity controller fed with a desired velocity of zero.

The filtering law is chosen to be

$$v_{\text{safe}} = \lambda(x) v_{\text{des}} + (1 - \lambda(x))\, \pi, \tag{3.55}$$

with

$$\lambda(x) = 1 - e^{-h_I(x)/\Delta_m}, \tag{3.56}$$

where $\Delta_m$ is a tuning parameter.

The simulation environment is a ROS-based C++ environment. The point cloud data is obtained from a Velodyne LIDAR sensor inside of the Gazebo simulator at a frequency of 10 hz. The simulation, including visualization in Gazebo and RVIZ, was able to run at a frequency of 500 hz on a modern laptop computer with an Intel i9 CPU.

The cave environment was a large 240m by 460m structure with one entrance and one exit. The cave height is constant at roughly 3m, but the width is constantly

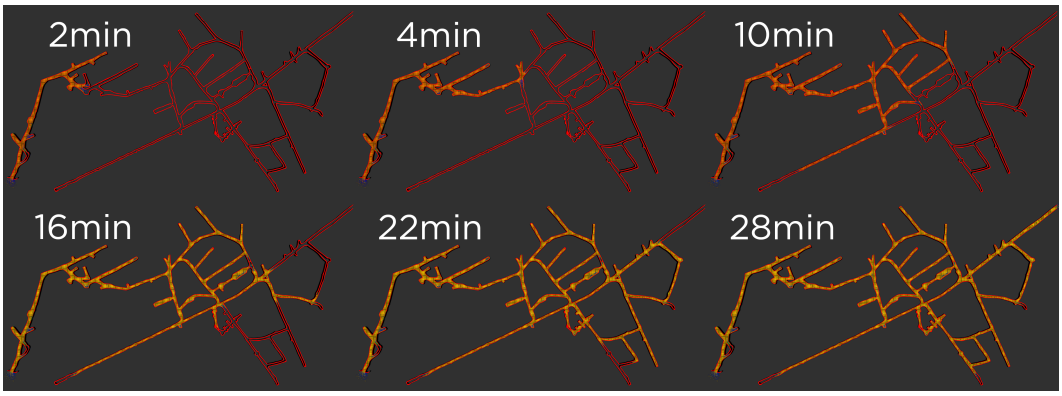



Figure 3.14: Pictures of the cave (in red) and the octomap (in yellow) being built throughout the 28 minutes it takes for the drone to completely explore the cave.

changing, and gets as small as 0.75m with several protruding areas. For reference, the size of the quadrotor is 0.5m in diameter.

The quadrotor was able to explore the entire 240m by 460m cave in just under 28 minutes. The maximum allowable speed from the planner was 5 m/s, which the drone reached during open areas of the cave. The average desired speed sent from the planner was 4.09 m/s, and the average speed of the drone after the safety filter was 3.28 m/s.

A positive value of the barrier function was maintained throughout, meaning the quadrotor never went closer than the minimum allowed distance to a point in the point cloud, which was set at 0.2 meters. The results can be shown in Figure 3.14

### 3.2.4 Applications: High-speed geofencing

Safety of small aerial vehicles is a heavily researched area. These works generally focus on safely planning trajectories rather than intervening along a desired trajectory. In this setting, [152] accounts for the low computational ability of drones, as well as the slow updates of mapping software, in their design of a planner for quick flight in unknown environments using motion primitives. While this and similar planners [150] have demonstrated results in unknown environments, they have not been demonstrated at the high speeds seen in drone racing. This is true for nearly all vision planners [78, 77], as the localization and mapping algorithms simply cannot keep up with speeds that human operators are capable of. Moreover, for known environments, reinforcement learning has been utilized to plan highly dynamic trajectories at speeds exceeding 60 km/h [142], but attempting to track these trajectories on hardware results in large tracking errors. This points to the

difficulty of adapting existing strategies to ensure safety with human operators in the loop.

While most drone racing research focuses on autonomy [104], this work departs from this paradigm with the goal of giving the human operators as much freedom and control authority as possible subject to safety constraints. In the context of geofencing, [168] presents an MPC-based approach, but it lacks the guaranteed feasibility of solutions.

Flight controllers on modern racing drones are able to track desired angular rates extremely well. With the availability of low-cost, high-speed electronic speed controllers (ESCs) and rate gyros, state-of-the-art controllers can track angular rates at control frequencies of 8 kHz. We utilize this by wrapping our controller around the closed-loop system of the drone with the onboard flight controller. Therefore, rather than the control inputs being the torques of the four motors, we command throttle and angular rates. This choice of architecture greatly simplifies the task of modeling the drone dynamics, and allows us to better filter the system in a way that minimizes the impact on the pilot. Moreover, the same filter can be applied to different drones with different dynamics, including those with six or eight rotors.

By modeling the response of the system to desired angular rate commands, the proposed method does not rely on perfect angular rate tracking. Through this, we also account for any delay in the filter stemming from communication between the sensors and the onboard flight controller.

The drone and flight controller system is modelled as rigid body motion in the Special Euclidean Group in 3 dimensions, SE(3). The state-space model $x \in \mathbb{R}^{13}$ is chosen to be $x = [p_w, q, v_w, \omega_b]^T$ where $p_w = [x, y, z]^T$ is the position in the world frame, $v_w = [v_x, v_y, v_z]^T$ are the world-frame velocities, $q$ is the quaternion representation of the orientation with respect to the world frame, and $\omega_b = [\omega_x, \omega_y, \omega_z]^T$ are the body-frame angular velocities.

To model the system's response to an angular rate command, we set the derivative of the angular rates to

$$\dot{\omega} = C(x)(\omega_{\text{des}} - \omega), \tag{3.57}$$

where $C(x) : \mathbb{R}^n \to \mathbb{R}_+$ is a (potentially state-dependent) function that determines how quickly the desired rates are tracked. For a well-tuned racing drone with minimal filter delay, its value should be on the order of 50, and can be treated as state-independent.

Lastly, to map the throttle command to thrust, we fit a second-order polynomial with data from the accelerometer and GPS. While this mapping will be dependent on the voltage, the inclusion of an integral term in the altitude controller is generally sufficient to eliminate any drift.

The primary goal of this work is to constrain the position of the drone inside a large polytope in the 3D space, inside of which the pilot has almost complete control, but is unable to leave. To this end, we define the safe set

$$h(x) = \min \left\{ r_x^2 - (x - x_c)^2, r_y^2 - (y - y_c)^2, r_z^2 - (z - z_c)^2 \right\}, \tag{3.58}$$

which is positive inside of a box with side lengths $(r_x, r_y, r_z)$ centered at $(x_c, y_c, z_c)$, and negative outside.

The backup controller $\pi(x)$ is a velocity controller on SE(3), inspired by [94]. The backup controller attempts to bring the drone to zero velocity in the $x, y, z$, but has one other goal which is very important: to bring the drone away from the boundary if it is too close. This is critical, as if the drone were to simply stop at the boundary, the pilot would be stuck at the edge of the safe set due to $\lambda(x)$ approaching 0. To achieve this, we set the desired velocity to

$$v_x = \begin{cases} 0 & r_x^2 - (x - x_c)^2 \geq \delta, \\ -(\delta - r_x^2 + (x - x_c)^2) & \text{otherwise.} \end{cases} \tag{3.59}$$

Under this backup controller, the drone will move a distance $\delta$ from the boundary before stopping. The desired velocities are identical for $y$ and $z$ directions.

Finally, the backup set $S_B$ is defined by the function

$$h_B = -\sqrt{v_x^2 + v_y^2 + v_z^2} + \epsilon. \tag{3.60}$$

This backup set ensures that the drone is able to slow itself to a speed of $\epsilon$, chosen to be 0.1 m/s, thus guaranteeing that the drone is able to stop before hitting the boundary.

Modifications must be made to the function $\lambda(x, h_I(x))$ to work well at very high-speeds. This is because $\beta$ must be made large to have smooth breaking at high speeds, which would make the filter overly conservative near the boundary at low speeds. This can be fixed simply by scaling the value of $h_I(x)$ by the inverse of the velocity towards the barrier. The safety filtering function used by the drone is

$$\lambda(x, h_I(x)) = 1 - \exp\left(-\frac{\beta h_I(x)^+}{v_\perp^+}\right), \tag{3.61}$$

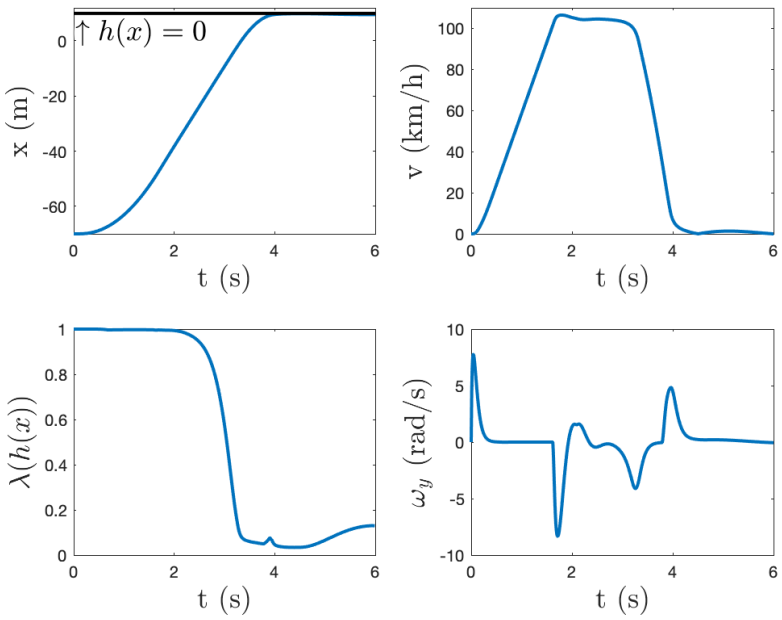
where $v_\perp$ is the velocity in the direction of the barrier.

Before testing the barrier functions on hardware, we first devise the test cases in simulation. While the safety filters are the same in simulation and on hardware, the hardware is operated by a human pilot, while the sim has its own desired controllers, so some discrepancies will arise because of this.

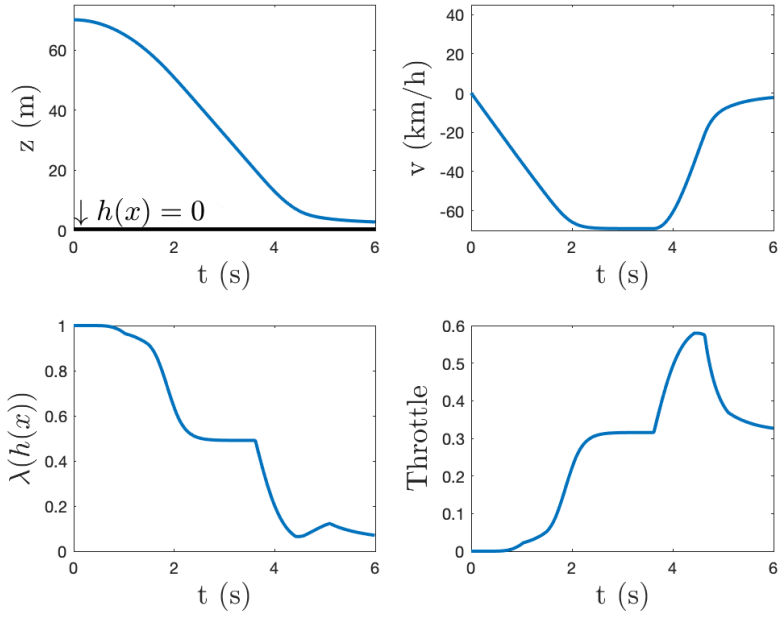
Two primary test cases were run in simulation, a high-speed horizontal test, with the goal of successful filtering at 100 km/h, and a free fall from 70 m. The results of the horizontal simulation are shown in Fig. 5a. The drone accelerates to a maximum speed of 107 km/h before being forced to stop. The minimum distance to barrier was 0.12 m. The free fall simulation was also successful: the drone accelerated to a top speed of 70 km/h downwards, before reaching a hover at a distance of 1.2 m above the barrier.

Our quadrotor is built on a Chimera 7" frame with four iFlight XING X2806.5 1300 KV brushless motors, a T-Motor F55A Pro II 4-in1 ESC, a MAMBA BASIC F722 Flight Controller (FC), a Teensy 4.1 microcontroller, a Vectornav VN-200 IMU+GPS, a FrSky R-XSR receiver, a DJI FPV air unit, and a Cadex FPV camera. We use a FrSky QX7 radio to send desired angular rates commands to Teensy microcontroller through the receiver. The VN-200 fuses GPS and IMU data with a built in extended Kalman filter. This data is sent to the Teensy as navigation data at 400 Hz. Using this data, the microcontroller then modifies these angular rates commands with the regulation function and then forwards them on to the FC. The FC runs betaflight, an open source software, to track the commanded angular rates. The PID loop runs at the gyro update rate at 8 kHz. The FC sends digital commands to the ESC using DSHOT600 at the same 8 kHz. FPV video is digitally streamed with an end to end latency of 25 ms from the DJI FPV air unit to DJI FPV goggles, which are worn by the operator.

To simplify the transition from simulation to hardware, the barrier functions are first generated in MATLAB for simulation, and then codegen is used to create C++ code that will run on hardware. All code used to generate and run the barrier functions on a Teensy 4.1 can be found at `https://github.com/DrewSingletary/racing_drone_geofencing`. This codebase also includes the interface for our specific




(a) Horizontal barrier active at 107 km/h.




(b) Vertical barrier active at -70 km/h.

Figure 3.15: Simulation results of the two primary hardware test cases. On the left, the drone accelerates towards the barrier at $x_w$ = 10 m. On the right, the drone free-falls from 70 m towards the barrier at $z_w$ = 0.5 m.


Figure 3.16: The 7" racing drone used for experiments.

receiver and flight controller, but this could be modified to fit other drone and radio configurations.

The execution time of the filter on the Teensy is approximately 350 $\mu$s. The algorithm runs at the update rate of the navigation data, which is 400 Hz.

A large number of flight tests were done to verify the safety guarantees provided by our method. We emphasize two specific examples, but several more flight tests are displayed in Figure 3.19.

**Test 1: Horizontal barrier at 104 km/h**. For this test, the pilot commanded the drone to head north at high speeds. The active component of the barrier was 40 m north of the initial position. The drone was able to reach a top speed of 104 km/h, before beginning to break at a distance of 15 m from the barrier. The pilot attempted to push the drone past the barrier, but was stopped at a minimum distance of 1.7 m from the barrier. Due to this, the pilot was then able to safely move away from the restricted airspace.

Figure 7a showcases the results of the experiment. The results agree strongly with the simulation data in Fig. 5a, despite the human piloting commanding different desired inputs than the simulation. This is, in part, due to the very accurate angular rate tracking showcased at the bottom right of Fig. 7a. While there is a slight delay in this tracking, this is properly modeled in (3.57), and thus it does not affect our

ability to guarantee safety.

Figure 7c shows the drone throughout this maneuver, highlighted in blue. Above this, the orientation of the drone at different snapshots is visualized. As shown, the drone reaches an angle of nearly 90° while breaking.

**Test 2: Free fall from 70 m**. At the beginning of this test, the pilot was flying at an altitude of 70 m and then sends no commands, mimicking a loss of radio connection. The barrier was chosen to be a distance of 0.5 m from the ground. After a short free-fall, reaching a vertical velocity of -60 km/h, the safety filter stabilizes the drone before coming to a stop at a distance of 1.8 m above the ground.

The data from this flight is visualized in Fig. 7b. Rather than plotting desired angular rates, we now showcase the desired throttle of the drone sent from the user compared to the throttle produced by the safety filter. Despite the pilot commanding no throttle for the entire duration of the descent, the drone is able to smoothly recover before crashing.

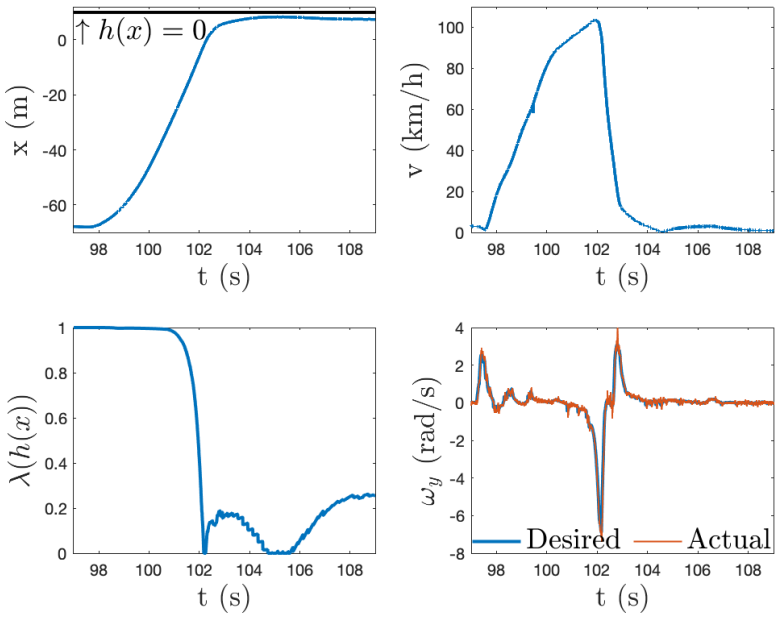
Again, when comparing this data to the simulation in Fig. 5b, notice the extremely similar results. In fact, the only major discrepancy, which is the fact that the simulation reached a speed of 10 km/h faster downwards than the drone, can be easily explained by a lack of drag in the simulation model. This did not occur during the horizontal tests, as the velocity controller is able to correct for this drag in flight.

**Testing for reliability and consistency.** Figure 8 highlights the reliability and consistency of this method in the application of geofencing. Four separate flights are plotted, two of which engage the horizontal barrier whereas two engage the vertical barrier. In each flight, the barrier is engaged two to four times, and every time, safety is maintained, and $\lambda$ never reaches zero, meaning the pilot never lost complete control of the drone for any period of time.

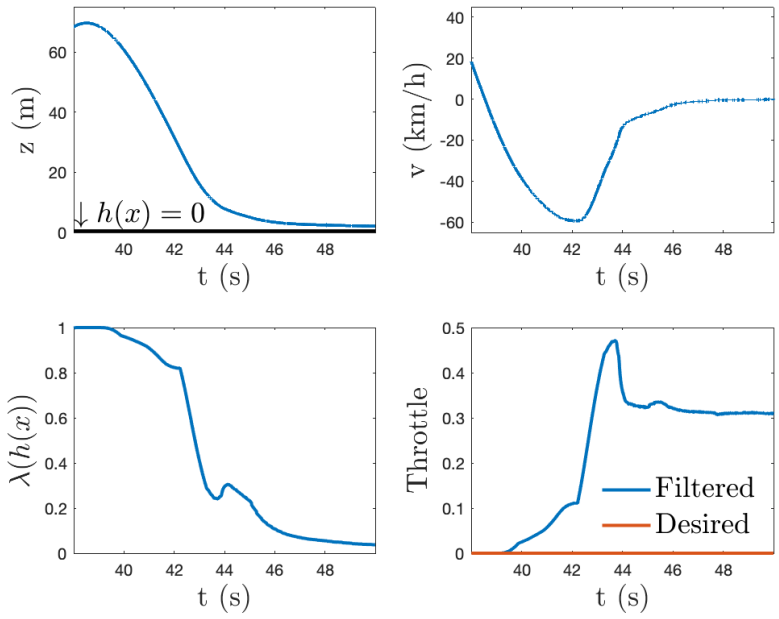
### 3.2.5 Time-varying backup controllers

The intuition behind a time-varying backup controller stems from the desire to execute a maneuver before engaging the backup controller that makes it easier for the backup controller to return to the backup set $\mathcal{S}_B$ while staying in $\mathcal{S}$. In this section, we will show that even a bad maneuver will not reduce the size of $S_I$ when compared to the original backup controller when properly formulated.

**Definition 10** (Time-Varying Backup Controller)**.** *Let* $u_M : \mathbb{R}^n \to U$, $u_B : \mathbb{R}^n \to U$

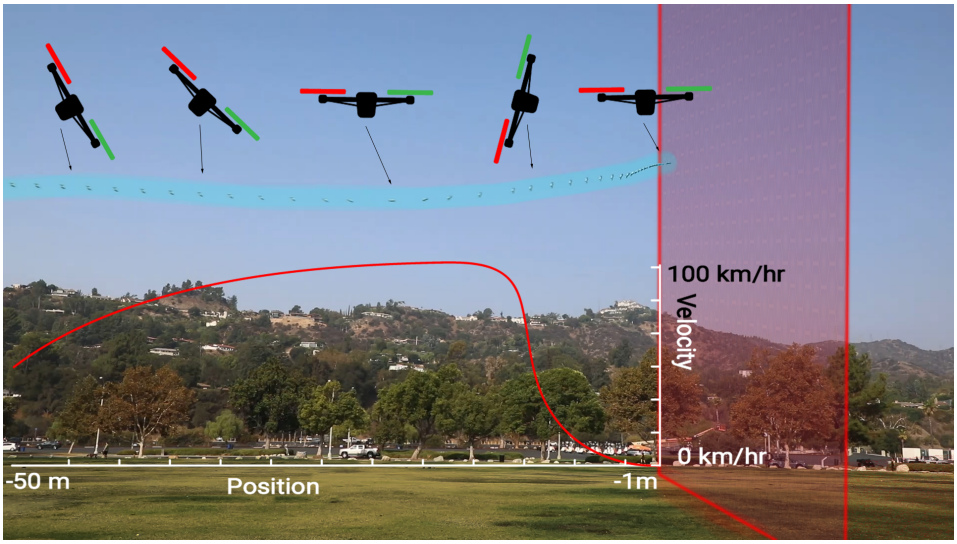



(a) Horizontal barrier active at 105 km/h.

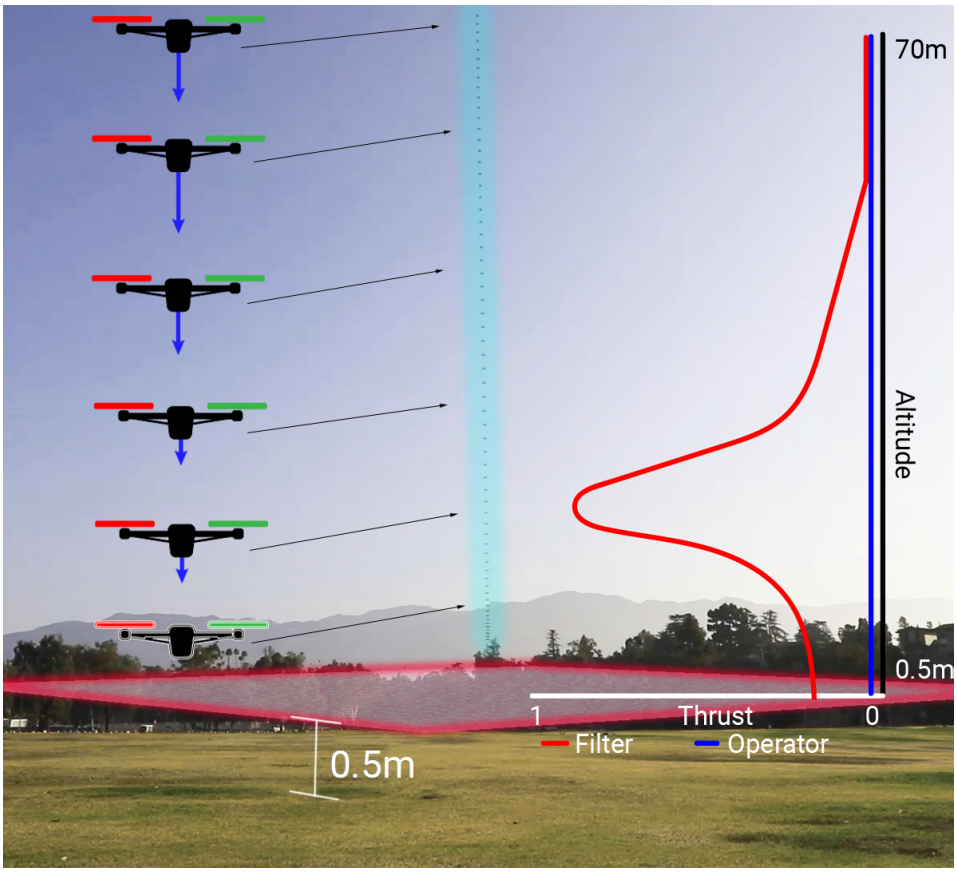



(b) Vertical barrier active at -60 km/h.

Figure 3.17: Two highlighted examples of geofencing with the high-speed racing drone.

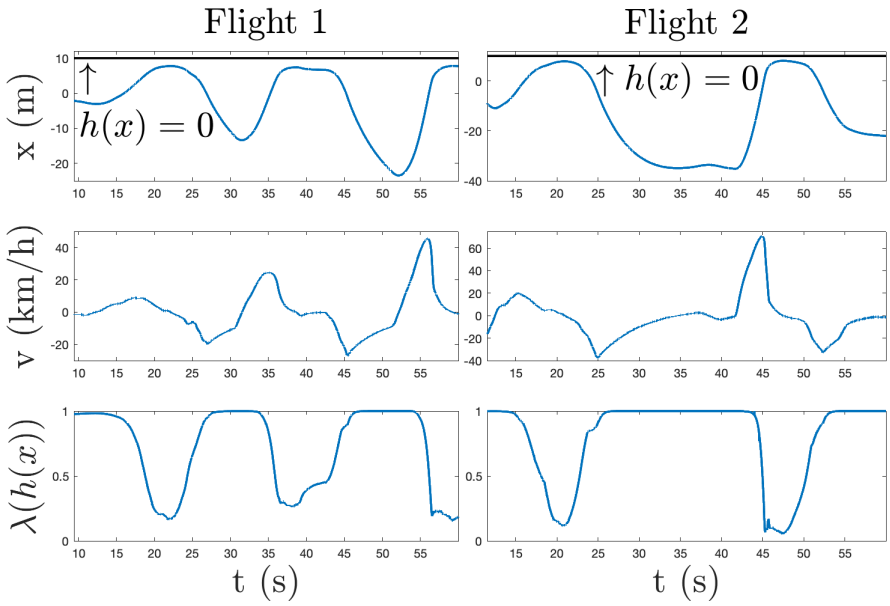



(a) Horizontal barrier active at 105 km/h.

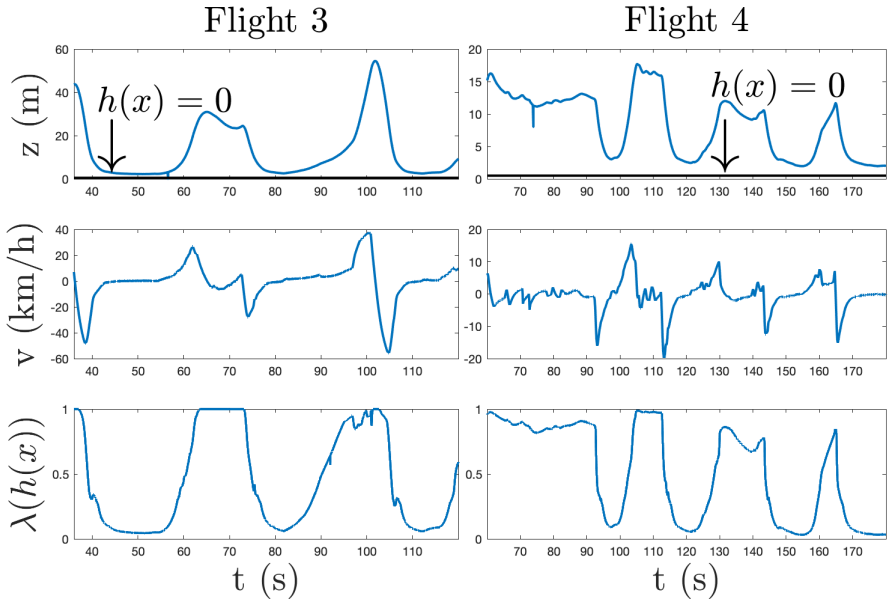



(b) Vertical barrier active at -60 km/h.

Figure 3.18: Actual drone flight during the two showcased example is highlighted in blue.




(a) Horizontal barrier active




(b) Vertical barrier active.

Figure 3.19: Four separate experimental runs where the drone is commanded to approach the barrier several times. $\lambda$ never reaches zero, meaning the pilot always has some amount of control, and the drone never leaves the defined safe set.

*and* $u_{M\to B}\mathbb{R}^n \times [T_M, T_M + \delta] \to U$ *for some* $\delta > 0$*. We call policies of the following form time-varying backup controllers:*

$$\pi(x,\tau) = \begin{cases} u_M(x) & \tau \le T_M \\ u_{M\to B}(x,\tau) & T_M \le \tau \le T_M + \delta \\ u_B(x) & \tau > T_M + \delta \end{cases}. \tag{3.62}$$

In (3.62), $\pi$ is executing two controllers ($u_M$ and $u_B$) in sequence and continuously transitioning between them with $u_{M\to B}$. The first controller, $u_M$ describes a maneuver to be performed before the backup controller is engaged. The controller $u_B$ is the backup controller that brings you to $S_B$. Finally, $u_{M\to B}$ is a time-varying controller that (locally Lipschitz) continuously transitions the input from $u_M(x)$ to $u_B(x)$ over time $\delta$. The potential benefit from introducing a maneuver is illustrated in Figure 3.20.

For a policy $\pi(x,t)$ and initial condition $x(t_0) = x_0 \in \mathbb{R}^n$, the solution to this closed-loop system is given by the flow:

$$x(t) = \Phi_\pi(x_0, t) = x_0 + \int_{t_0}^{t} \left(f(x) + g(x)\pi(x,\tau)\right) d\tau. \tag{3.63}$$

In this case, for any initial time $t_0 \ge T_M$, the maneuver will not be performed. To remedy this, we introduce an alternative notion for the flow of the system with a time-varying backup controller:

$$\begin{aligned} x(t) &= \Phi_\pi(x_0, t, \tau_0) \\ &= x_0 + \int_{t_0}^{t} \left(f(x) + g(x)^\top \pi(x, \tau - \tau_0)\right) d\tau. \end{aligned} \tag{3.64}$$

This flow will implicitly maintain forward-invariance of the following set:

$$\begin{aligned} &S_I(\tau_0) = \\ &\left\{ x \in \mathbb{R}^n \,\middle|\, \min_{t\in[0,T]} \left\{ h\left(\Phi_\pi(x,t,\tau_0)\right), h_B\left(\Phi_\pi(x,T,\tau_0)\right)\right\} \ge 0 \right\} \end{aligned} \tag{3.65}$$

which is equivalent to the formulation in (3.6) where the policy $\pi$ corresponds to $\pi$ with the system time offset by $\tau_0$. The choice of time-offset parameter $\tau_0$ ultimately determines the shape $S_I$.

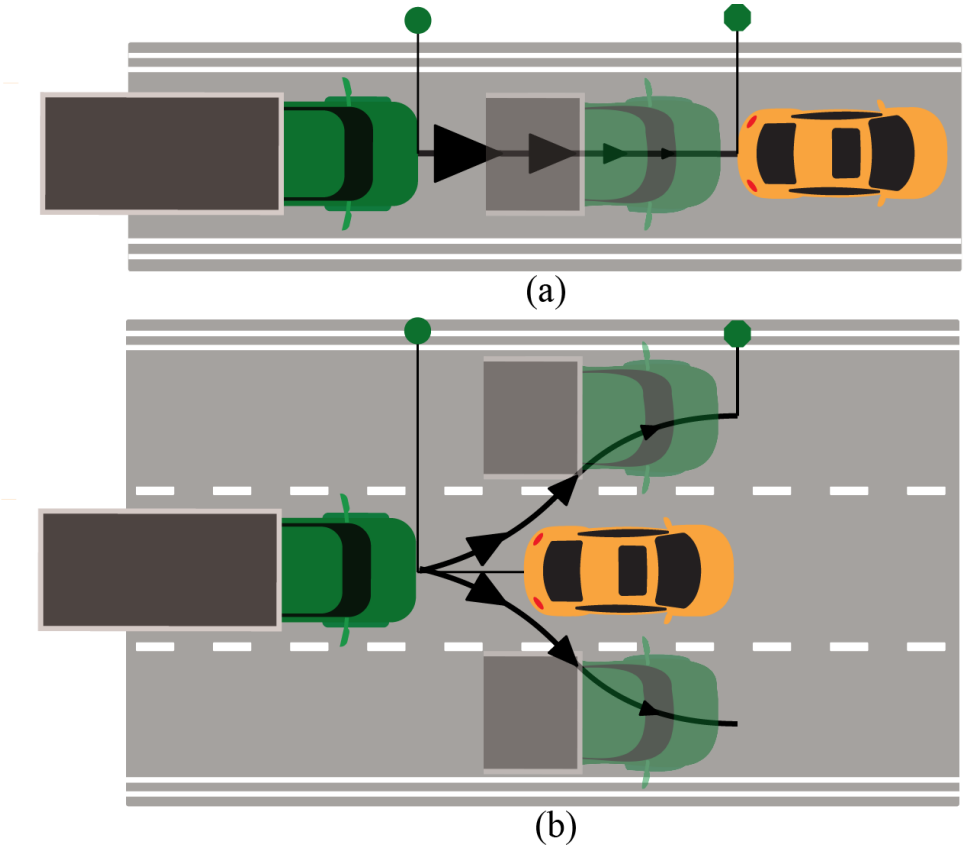



Figure 3.20: Illustration of the benefits of time-varying backup controllers. (a) A high-inertia semi truck driving along the highway must keep a large distance behind lighter cars in order to stop before reaching them. (b) By adding a maneuver to switch lanes before stopping, the truck can follow much more closely, under the condition that no one is in the lane.

**Example 8.** *Let us consider two examples that demonstrate how the parameter* $\tau_0$ *transforms the final invariant set. First, suppose* $\tau_0 = t_0 - T_M - \delta$*. In this case,*

$$\begin{aligned}\Phi_\pi(x_0, t, t_0 - T_M - \delta) &= x_0 + \int_{t_0}^{t} \left(f(x) + g(x)^\top u_B(x)\right) d\tau \\ &= \Phi_{u_B}(x_0, t),\end{aligned} \tag{3.66}$$

*which implies that* $S_I(t_0 - T_M - \delta)$ *recovers the safe-set of the nominal backup controller* $u_B$*.*

*Second, suppose* $\tau_0 = t_0 + T - T_M$*, which corresponds to the case of only the maneuver being performed over time* $T$*:* $\Phi_\pi(x_0, T, t + T - T_M) = \Phi_{u_M}(x_0, T)$*. This would likely result in* $S_I = \emptyset$*, as the backup maneuver* $u_M(x)$ *would not bring the system back into the backup set* $S^B$*.*

Clearly, the choice of $\tau_0$ is critical to the performance of the time-varying backup controller and, as demonstrated below, the right choice of $\tau_0$ can yield a control invariant set that is larger than the one guaranteed by the nominal controller.

#### 3.2.5.1 Time-Offset for Time-Varying Backup Control Filters

Rather than pick a single $\tau_0$ for the entire evolution of our dynamical system we will allow it to change as a function of the system time and the current state:

$$\tau_0^*(x,t) \coloneqq \min_{\tau_0 \in [-t, T-t]} \tau_0 \tag{3.67}$$

$$\text{s.t.} \quad \Phi_\pi(x, t+\tau, \tau_0) \in S \quad \forall \tau \in [0,T] \tag{3.68}$$

$$\Phi_\pi(x, t+T, \tau_0) \in S_B. \tag{3.69}$$

Recall that in (3.6), we give the policy a maximum time horizon of $T$ to bring the system into the backup set. We operate under the assumption that $T > T_M + \delta$ so that the horizon can in principle cover both the maneuver and leave enough time for the backup set controller to take the system back to $S_B$. (3.68) ensures that the chosen $\tau_0^*$ will be safe for states in a horizon of length $T$. The constraint in (3.69) ensures that the final state of the horizon belongs to the backup set $S_B$. Notice that these two constraints, by definition, imply that $\pi(x, \tau, \tau_0)$ is a valid backup controller with time-offset $\tau_0^*$. The linear objective ensures we include as much of our maneuver behavior as possible, otherwise we could simply set $\tau_0 = t_0 - T_M - \delta$ and only execute the backup controller.

We now show that a time-offset chosen by this optimization problem results in a strict improvement over the nominal case of a single backup controller.

**Theorem 15.** *The safe set for a nominal backup controller $u_B(x)$ is strictly smaller than the safe set provided by a time-varying backup controller $\pi(x, \tau - \tau_0)$ with time-offset $\tau_0^*$ as defined in* (3.67) *so that the following condition holds true:*

$$S_I(u_B) \subseteq S_I(\tau_0^*) \tag{3.70}$$

*Proof.* Suppose for the sake of contradiction that there exists an $x \in S_I(u_B)$ so that $x \notin S_I(\tau_0^*)$. Now $x \in S_I(u_B)$ implies that $\Phi_{u_B}(x,t) \in S \; \forall t \in [t_0, t_0 + T]$ and $\Phi_{u_B}(x, t_0 + T) \in S_B$ by definition. Recall that $\tau_0 = t_0 - T_M - \delta$ implies that

$$\Phi_\pi(x, t, t_0 - T_M - \delta) = \Phi_{u_B}(x,t).$$

Therefore $\tau_0 = t_0 - T_m - \delta$ is feasible for the optimization problem $\tau_0^*$. This is a contradiction since $x \notin S(\tau_0^*)$ implies $t_0^*$ has no feasible solution but $\tau_0 = t_0 - T_m - \delta$ is a feasible solution. □

Notice that because $\tau_0^*$ is an optimization problem over time rather than state or control input, it is amenable to the time discretization required to numerically integrate ODE's and can be efficiently approximated in practice. We introduce Algorithm 4 to approximate $\tau_0^*$ online within the mixing framework.

**Algorithm 4** Online Approximation of $\tau_0^*$

**Inputs:**
$x, t$ ▷ Current State and System Time
$\Delta$ ▷ Discrete-Time Increment
$\tau_0^*(x(t-\Delta), t-\Delta)$ ▷ Previous $\tau_0^*$ Solution
$\pi$ ▷ Time-Varying Backup Controller
**procedure** ResetTimeOffset
    $\mathcal{P} \leftarrow \{\Phi_\pi(x, t+\tau, -t) \,|0 \le \tau \le T\}$
    **if** $(\mathcal{P} \subseteq S) \wedge (\Phi_\pi(x, t+T, -t) \in S_B)$ **then**
        **return** $\tau_0^*(x(t), t) \leftarrow -t$
    **else**
        **return** $\tau_0^*(x(t), t) \leftarrow \tau_0^*(x(t-\Delta), t-\Delta) + \Delta$
    **end if**
**end procedure**

The end result of Algorithm 4 is that the time-offset $\tau_0$ gets set to the the negation of system time whenever the system is able to perform the backup maneuver $u_M$ for the entirety of $T_M$ and remain safe. Otherwise $\tau_0^*$ is allowed to increase with the system time.

**Remark 7.** *The maneuver $u_M(x)$ is chosen not to be time-varying is so that $\tau_0$ may be reset freely before $t - \tau_0 < T_M$ without discontinuities. While there is likely to be a discontinuity in the input when reset after $t - \tau_0 > T_M$, it can only occur every $T_M$ seconds, and the resulting backup maneuver corresponding to the new $\tau_0$ is Lipschitz continuous. Therefore, the reset-induced discontinuities do not affect the safety guarantees, nor do they lead to high-frequency oscillations in the input.*

#### 3.2.5.2 Multiple Backup Policies

Now that the theory is established for the time-varying backup maneuvers, we can introduce the utilization of multiple maneuvers. For a maneuver $i$, the time-varying

backup controller takes the following form:

$$\pi^i(x,\tau) = \begin{cases} u_{M^i}(x) & \tau \le T_M \\ u_{M^i \to B}(x,\tau) & T_M \le \tau \le T_M + \delta \\ u_B(x) & \tau > T_M + \delta \end{cases}. \tag{3.71}$$

For ease of notation, we assume that $T_M$ is constant among the different maneuvers, but in practice, there is no need for this to be the case.

Much like the introduction of a single maneuver $u_M(x)$ resulted in a safe set $S^I$ at least as large as, and generally larger, than the original set, the inclusion of multiple maneuvers can further increase the size of $S^I$. For example, in the case of Figure 3.20, having multiple maneuver options could correspond switching to different lanes, increasing the likelihood that one of them is empty.

As shown in Figure 3.20, naive mixing of backup maneuvers may not yield a safe control input and could cause discontinuities in the resulting control action taken. In the case of the truck, going left or going right are mutually exclusive. As a secondary consideration we also want to avoid the expensive computation of rolling out trajectories for all possible maneuvers every time we wish to generate a $u_{\text{act}}$.

One possible condition for switching is once it is no longer possible to perform the current maneuver and has engaged only the backup controller portion, i.e. $\tau_0 \ge t_0 - T_M - \delta$. In this case, the backup controller is already engaged, switching to a different policy $\pi^i(x, t, \tau_0)$ will not result in a discontinuity. This process can be continued for all possible backup maneuvers, until one of the maneuvers allows a reset of $\tau_0$. If no choice of backup maneuver allows a reset, the backup controller continues executing maintaining the safety of the system. This is formalized in Algorithm 5.

**Remark 8.** *Similar to Algorithm 4, this algorithm results in a discontinuity no more frequently than every $T_M$ seconds. Moreover, only one maneuver is evaluated at each time-step, resulting in minimal computational burden.*

For backup CBFs, the decentralized multi-agent case was handled in [35]. In this work, we follow a similar approach with our regulation functions along backup maneuvers.

Suppose we have a set of agents $a \in \mathcal{A}$, each with its own independent state $x^a \in \mathbb{R}^n$ and nominal safe set defined by continuously differentiable $h^a : \mathbb{R}^n \to \mathbb{R}$. We

**Algorithm 5** Maneuver Switching Algorithm

**Inputs:**
$x, t$ ▷ Current State and System Time
$j$ ▷ Current Maneuver
$\Delta$ ▷ Discrete-Time Increment
$\tau_0^*(x(t-\Delta), t-\Delta)$ ▷ Previous $\tau_0^*$ Solution
$\{\pi^i\}_{i=1}^J$ ▷ Time-Varying Backup Controllers
**procedure** SwitchManeuver
  **if** $\tau_0^*(x(t-\Delta), t-\Delta) \geq t - T_M - \Delta$ **then**
    $j \leftarrow j + 1$
    **if** $j > J$ **then**
      $j = 1$
    **end if**
    $\mathcal{P}^j \leftarrow \{\Phi_{\pi^j}(x, t+\tau, -t) \,|0 \leq \tau \leq T\}$
    **if** $\left(\mathcal{P}^j \subseteq S\right) \wedge \left(\Phi_{\pi^j}(x, t+T, -t) \in S_B\right)$ **then**
      **return** $\tau_0^*(x(t), t) \leftarrow -t$
    **else**
      **return** $\tau_0^*(x(t), t) \leftarrow \tau_0^*(x(t-\Delta), t-\Delta) + \Delta$
    **end if**
  **end if**
**end procedure**

are also concerned with avoiding collisions between agents using pair-wise barrier $h^{ab} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Combining these two conditions we define a safe set for each agent:

$$S^a = \left\{ x_0^a \in \mathbb{R}^n \,\middle|\, \left( \bigwedge_a h^a(x^a) \geq 0 \right) \wedge \left( \bigwedge_{a \neq b} h^{(a,b)}(x^a, x^b) \geq 0 \right) \right\} \tag{3.72}$$

This leads to the following result on safety in the decentralized case.

**Theorem 16** (Distributed Safety)**.** *Consider a set of agents $\mathcal{A}$, with each agent $a \in \mathcal{A}$ described by* (1.1)*, i.e., $\dot{x}^a = f^a(x^a) + g^a(x^a)\pi^a(x)$ with each agent following a TBC $\pi^a$ using independent time-offsets $\tau_0^{*a}$. Then*

- *For all $a \in \mathcal{A}$ if the initial state $x_0^a \in S^a$ then $S^a$ in* (3.72) *is safe (forward invariant) so long as each agent has state information, $x^b$ for $b \in \mathcal{A} \setminus \{a\}$.*
- *The global safe set*

$$S = \left\{ x_0 \in \mathbb{R}^{n|\mathcal{A}|} \,\middle|\, \left( \bigwedge_a h^a(x^a) \geq 0 \right) \wedge \left( \bigwedge_{a \neq b} h^{ab}(x^a, x^b) \geq 0 \right) \right\} \tag{3.73}$$

*is forward invariant where $x_0 \in \mathbb{R}^{n|\mathcal{A}|}$ is the concatenation of all agent states: the global state.*

*Proof.* Recall that by Theorem 15, a TBC using $\tau_0^{*a}$ as offset results in a backup controller that renders $S^a$ safe (forward invariant) for each agent $a \in \mathcal{A}$. Clearly, if for all $a \in \mathcal{A}$ $S^a$ is forward invariant then $\cap_{a\in\mathcal{A}}$ is forward invariant. This allows us to perform the following derivation:

$$\cup_{a\in A} S^a = \left\{ x \in \mathbb{R}^{n|\mathcal{A}|} \,\middle|\, \bigwedge_{a\in A} x^a \in S^a \right\} \tag{3.74}$$

$$= \left\{ x \in \mathbb{R}^{n|\mathcal{A}|} \,\middle|\, \bigwedge_{a\in\mathcal{A}} h^a(x^a) \geq 0 \wedge \forall_{b\in\mathcal{A}\setminus\{a\}} h^{ab}(x^a, x^b) \geq 0 \right\} \tag{3.75}$$

$$= \left\{ x_0 \in \mathbb{R}^{n|\mathcal{A}|} \,\middle|\, \left( \bigwedge_{a} h^{(a)}(x^{(a)}) \geq 0 \right) \wedge \left( \bigwedge_{a\neq b} h^{(ab)}(x^{(a)}, x^{(b)}) \geq 0 \right) \right\} \tag{3.76}$$

$$= S. \tag{3.77}$$

Here, (3.75) follows from the application of the definition of $S^a$ and (3.76) follows from the application of associativity and commutativity the logic and operator ($\wedge$). □

If multiple agents have multiple maneuvers, the current maneuver must be known by all agents. As a consequence of $u_B^i(x)$ being constant in time throughout all maneuvers for an agent $i$, the drones are able to independently update their maneuvers at any time $\tau_0 \geq t - T_M - \delta$. However, all drones must have knowledge of the new maneuver being utilized, in order to know whether or not they can reset $\tau_0$.

Figure 3.21 demonstrates the idea behind requiring knowledge of the other agents' maneuvers to guarantee safety of the individual.

#### 3.2.5.3 Application: Multi-drone collision avoidance

For this work, the nominal safe set of each agent $i$ is chosen as a box of free space centered at $(x_c, y_c, z_c) \in \mathbb{R}^3$ with side lengths $r_x, r_y, r_z$,

$$h^i(x) = \min\left\{ r_x^2 - (p_x^i - x_c)^2, r_y^2 - (p_y^i - y_c)^2, r_z^2 - (p_z^i - z_c)^2 \right\}. \tag{3.78}$$

The safety constraint between agents $i$ and $j$ is given as

$$h^{ij}(x) = (p_x^i - p_x^j)^2 + (p_y^i - p_y^j)^2 + (p_z^i - p_z^j)^2 - 4r^2, \tag{3.79}$$

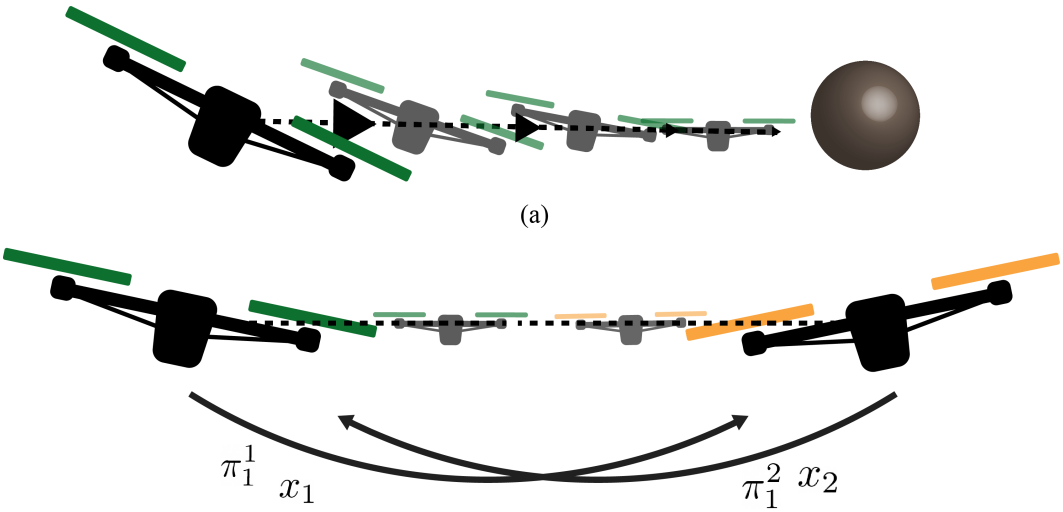



Figure 3.21: Illustration contrasting single agent and multi-agent implementation. (a) shows the single agent case, while (b) shows the multi-agent case where state and backup policy information is required to guarantee safety.

which ensures that no collision occurs between the drones, which are modeled as spheres of radius $r$.

The backup controller $u_B(x)$ is the same velocity controller on SE(3), inspired by [94], that was used in in the previous example. The backup controller attempts to stop the drone, and repel it from the boundary of the safe set if it gets too close.

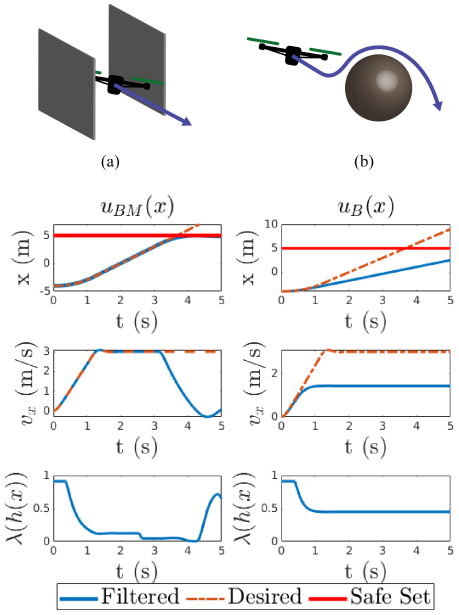
The backup set $S_B$ is defined by the function

$$h_B = -\sqrt{v_x^2 + v_y^2 + v_z^2} + 0.1, \tag{3.80}$$

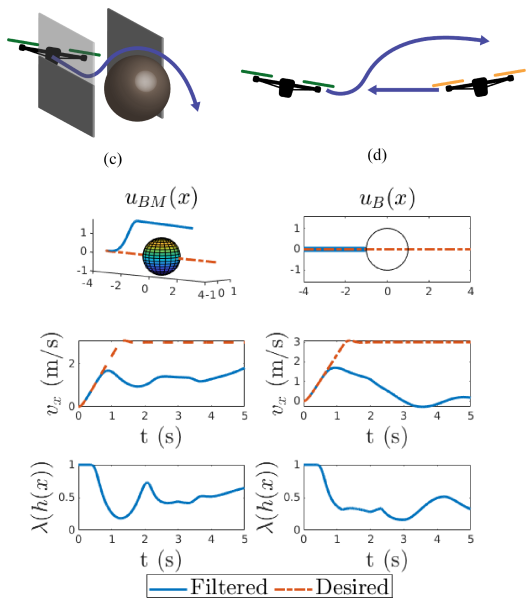
which guarantees that the drone is able to slow itself to a speed of 0.1 m/s, which is forward invariant under the backup controller.

In this section, we will outline two simple maneuvers, and demonstrate how they improve performance on hardware. We call these maneuvers the "carry on" and "evade" maneuvers.

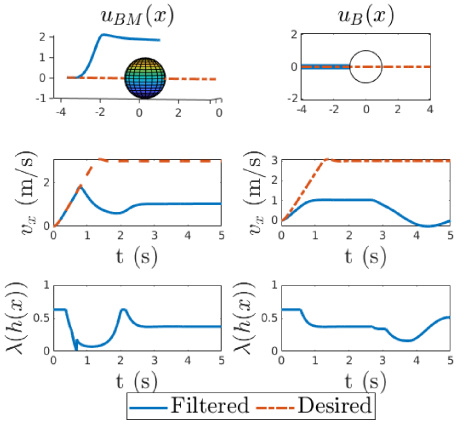
**Carry on maneuver.** The carry on maneuver is exactly how it sounds: the maneuver attempts to propagate the current desired input forward. This input is held constant throughout $T_M$, and is updated every time $\tau_0$ is allowed to reset. With the carry on



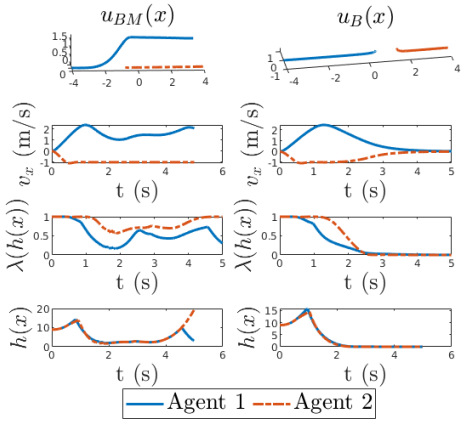

(a) Carry on maneuver compared to only $u_B(x)$ in tightly constrained space.




(b) Evade maneuver compared to only $u_B(x)$ when encountering a spherical obstacle.




(c) Carry on maneuver switching to evade maneuver when encountering a spherical obstacle in tightly constrained space




(d) Multi-agent sim

Figure 3.22: Simulation results capturing the performance benefits of allowing backup maneuvers. While the value of $\lambda$ is often lower, due to being closer or faster near the barrier, there is significantly better alignment with the desired velocities.

maneuver, the overall policy is defined as

$$\rho^i(x,\tau) = \begin{cases} u_{\text{des}} & \tau \leq T_M \\ u_{M^i \to B}(x,\tau) & T_M \leq \tau \leq T_M + \delta \\ u_B(x) & \tau > T_M + \delta \end{cases}. \tag{3.81}$$

This policy shines when a skilled pilot wants to move along the edge of the safe set, when one might have $\lambda << 1$. This is illustrated in Figure 3.22a.

**Evade maneuver.** The evade maneuver is a maneuver that attempts to reposition the drone before coming to a stop. This is similar to a lane change, as in Chapter 3.20. For this specific maneuver, we attempt to bring the drone upwards, but similar results could be achieved with evading in different directions. The policy is written as

$$\rho^i(x,\tau) = \begin{cases} u_z(x) & \tau \leq T_M \\ u_{M^i \to B}(x,\tau) & T_M \leq \tau \leq T_M + \delta \\ u_B(x) & \tau > T_M + \delta \end{cases}, \tag{3.82}$$

with $u_z(x)$ being the velocity controller attempting to track to the desired height. This policy shines when the pilot does not react to avoid an obstacle fast enough, and relies on the safety filter to do it for them. This is illustrated in Figure 3.22b.

Finally, the decentralized, multi-agent approach was tested for two agents moving towards each other. Agent 1 is utilizing with the evade maneuver, while Agent 2 attempts the carry-on maneuver. The desired velocity $v_x$ of Agent 1 is 3 m/s, and for Agent 2, it is -1 m/s.

Figure 3.22d demonstrates the advantage over simply utilizing the backup controller $u_B(x)$ in preventing the drones from reaching a deadlock.

#### 3.2.5.4 Hardware results

The quadrotor used in our experiments is a consumer "Cinewhoop" drone shown in Fig. 3.23. We add several components to this off the shelf drone for the computation of the barrier functions: a Teensy 4.1 microcontroller, a Xbee 93b radio and an Adafruit BNO055 IMU. Optitrack position and attitude data is streamed to the drone at 20 Hz via a Xbee93B radio. The Teensy 4.1 reads this position data along with IMU data at 100 Hz. The on-board barrier computation issues angular rates commands at 100 Hz to a flight controller running Betaflight, an open source flight

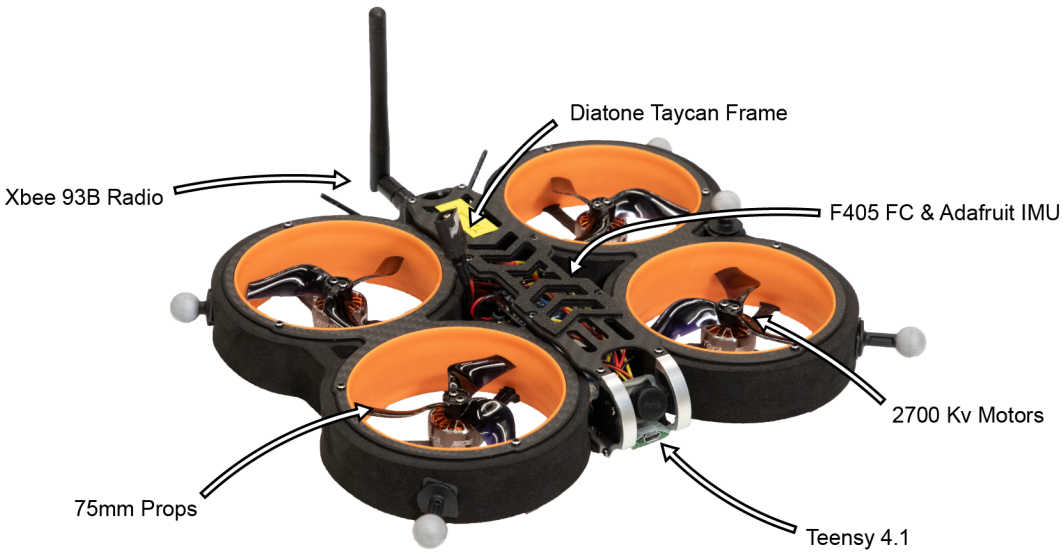
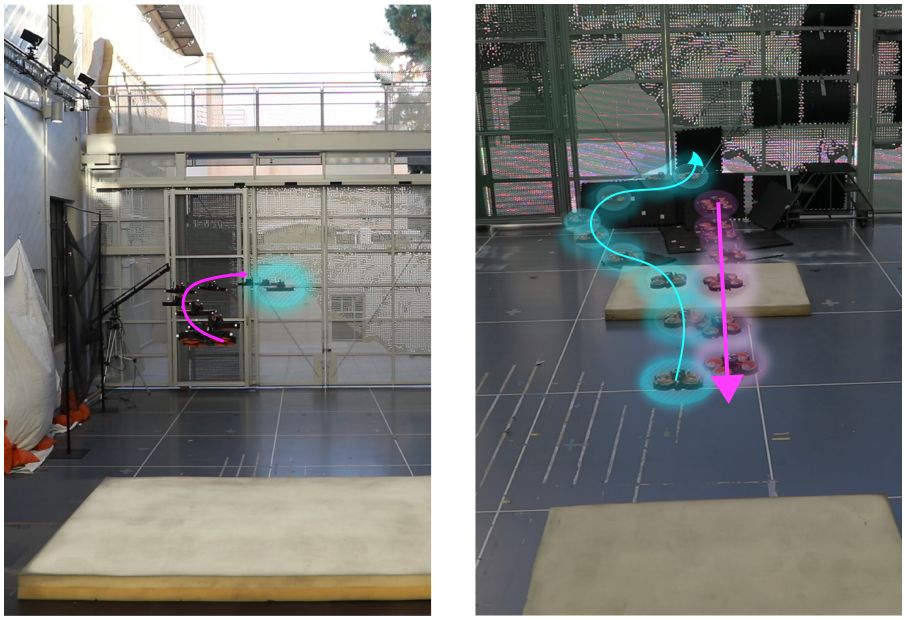



Figure 3.23: The Cinewhoop quadrotor used in the experiments

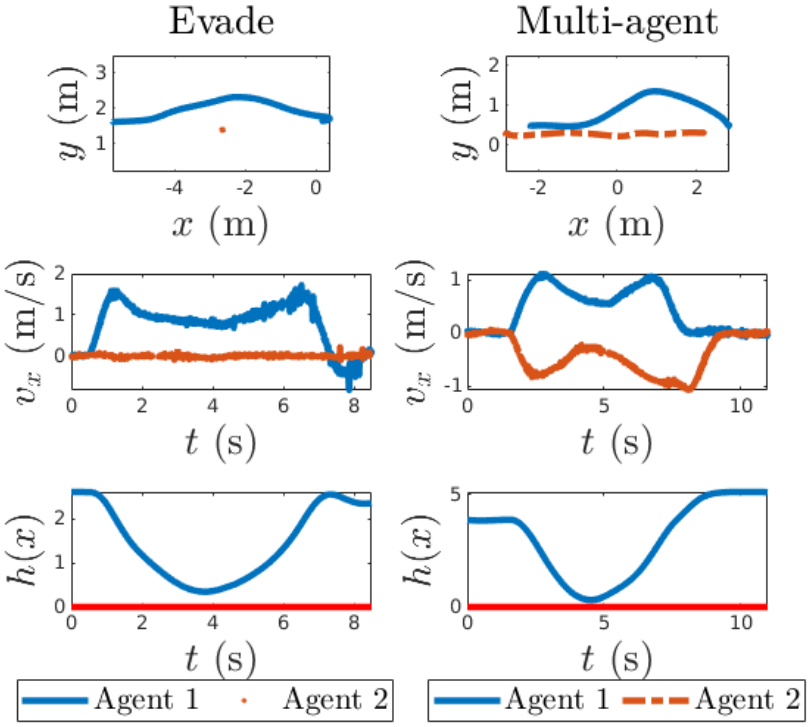
control software which is tightly integrated with the flight controller. This allows angular rate tracking at the internal IMU update frequency of 8 kHz.

To validate the tractability and robustness of our algorithms in flight, we recreate the simulation results showcased in Figures 3.22b and 3.22d with two identical "Cinewhoop" drones. The results are demonstrated in Figure 3.24b.

The only major difference between the simulation and the hardware was the choice of the evade maneuver to move to the side rather than up. This was done due to limit the effects of the downward airflow on the other agent. As demonstrated in the plots, the TBC's ran quite well despite the very noisy velocity data, as well as the computational constraints of a microcontroller. However, the CBF and all other onboard computations were easily ran at the update rate of the Betaflight's desired inputs, 100 Hz.



(a) Illustration of the experimental results demonstrating safe flight of multiple drones through the use of time-varying backup controllers.




(b) Plots of the hardware results showcased above. For more hardware results, please refer to the attached video.

Figure 3.24: Hardware results using time-varying backup maneuvers.

*Chapter 4*

# DISCRETE-TIME SAFETY REGULATION

## 4.1 Safe policy synthesis in multi-agent POMDPs via discrete-time barrier functions

Complex mission planning of multiple heterogeneous robots, e.g., flying and ground robots, presents an inherent tension between the need for greater autonomy and the absolute necessity of strong safety [6] and performance guarantees [5]. Safety is crucial for the duration of a safety-critical mission, for example, those involving human-robot interactions [154]. The planning problem becomes even more involved in the presence of partial or uncertain information about the world, as well as stochastic actions and noisy sensors [119, 132].

Multi-agent partially observable Markov decision processes [101, 14] provide a sequential decision-making formalism for high-level planning of multiple autonomous agents under partial observation and uncertainty. In MPOMDPs, the agents share their local observations and make decisions based on a global information state (the joint belief). Despite this unique modeling paradigm, the computational complexity of MPOMDPs is PSPACE-complete [26, 64]. Therefore, several promising approximate methods for solving MPOMDPs have been proposed in the literature, e.g., sampling-based methods [14] and point-based methods [133]. However, it is difficult to provide safety assurances when one employs approximate methods for solving MPOMDPs, as such methods either use discretization techniques [62] or finite-state controllers [135].

In this section, we extend the application of barrier functions from low-level safety constraints of dynamical systems to high-level safety objectives of MPOMDPs. Our results are based on the observation that the joint belief evolution of an MPOMDP is described by a discrete-time system [9, 10, 7]. We begin by formulating a, both necessary and sufficient, theorem for safety verification of a given set for discrete-time systems based on *discrete-time barrier functions* (DTBFs) and we demonstrate that our formulation allows for more complicated safe belief sets described by Boolean compositions of DTBFs. Then, we apply these DTBFs to study the safety of a given set of safe beliefs. We propose online methods based on one-step greedy algorithms to either synthesize a safe policy for an MPOMDP or synthesize a safety

filter for an MPOMDP given a nominal planning policy. We illustrate the efficacy of the proposed approach by applying it to an exploration scenario of a team of heterogeneous robots in a high-fidelity simulation environment.

An MPOMDP [101, 14] provides a sequential decision-making formalism for high-level planning of multiple autonomous agents under partial observation and uncertainty. At every time step, the agents take actions and receive observations. These observations are shared via (noise and delay free) communication and the agents decide in a centralized framework.

**Definition 11.** *An MPOMDP is a tuple*

$$\left(I, Q, p^0, \{A_i\}_{i\in I}, T, R, \{Z_i\}_{i\in I}, O\right),$$

*wherein*

- $I$ *denotes a index set of agents;*
- $Q$ *is a finite set of states with indices* $\{1, 2, \ldots, n\}$ *(which can be described as the product space of the states of all agents);*
- $p^0 : Q \to [0, 1]$ *defines the distribution of the initial states, i.e.,* $p^0(q)$ *denotes the probability of starting at* $q \in Q$*;*
- $A_i$ *is a finite set of actions for agent* $i$ *and* $A = \times_{i\in I} A_i$ *is the set of joint actions;*
- $T : Q \times A \times Q \to [0, 1]$ *is the transition probability, where* $T(q, a, q') := P(q^t = q' | q^{t-1} = q, a^{t-1} = a)$,
  $\forall t \in \mathbb{Z}_{\geq 1}, q, q' \in Q, a \in A$, *i.e., the probability of moving to state* $q'$ *from* $q$ *when the joint actions* $a$ *are taken;*
- $R : Q \times A \to \mathbb{R}$ *is the immediate reward function for taking the joint action* $a$ *at state* $q$*;*
- $Z_i$ *is the set of all possible observations for agent* $i$ *and* $Z = \times_{i\in I} Z_i$*, representing outputs of discrete sensors. Often,* $z \in Z$ *are incomplete projections of the world states* $q$*, contaminated by sensor noise;*
- $O : Q \times A \times Z \to [0, 1]$ *is the observation probability (sensor model), where* $O(q', a, z) := P(z^t = z | q^t = q', a^{t-1} = a)$, $\forall t \in \mathbb{Z}_{\geq 1}, q \in Q, a \in A, z \in Z$, *i.e., the probability of seeing joint observations* $z$ *given joint actions* $a$ *were taken and resulting in state* $q'$*.*

Since the states are not directly accessible in an MPOMDP, decision making requires the history of joint actions and joint observations. Therefore, we must define the notion of a joint *belief* or the posterior as sufficient statistics for the history [21]. Given an MPOMDP, the joint belief at $t = 0$ is defined as $b^0(q) = p^0(q)$ and $b^t(q)$ denotes the probability of the system being in state $q$ at time $t$. At time $t + 1$, when joint action $a \in A$ is taken and joint observation $z \in Z$ is observed, the belief is updated via a Bayesian filter as

$$\begin{aligned} b^t(q') &= P(q'|z^t, a^{t-1}, b^{t-1}) \\ &= \frac{P(z^t|q', a^{t-1}, b^{t-1})P(q'|a^{t-1}, b^{t-1})}{P(z^t|a^{t-1}, b^{t-1})} \\ &= \frac{P(z^t|q', a^{t-1}, b^{t-1})}{P(z^t|a^{t-1}, b^{t-1})} \\ &\quad \times \sum_{q \in Q} P(q'|a^{t-1}, b^{t-1}, q)P(q|a^{t-1}, b^{t-1}) \\ &= \frac{O(q', a^{t-1}, z^t) \sum_{q \in Q} T(q, a^{t-1}, q')b^{t-1}(q)}{\sum_{q' \in Q} O(q', a^{t-1}, z^t) \sum_{q \in Q} T(q, a^{t-1}, q')b^{t-1}(q)}, \end{aligned} \tag{4.1}$$

where the beliefs belong to the belief unit simplex

$$\mathcal{B} = \left\{ b \in [0, 1]^{|Q|} \mid \sum_{q \in Q} b^t(q) = 1, \ \forall t \right\}.$$

A policy in an MPOMDP setting is then a mapping $\pi : \mathcal{B} \to A$, i.e., a mapping from the continuous joint beliefs space into the discrete and finite joint action space. The special case of $I$ being a singleton (only one agent) is known as a partially observable Markov decision process (POMDP) [141].

The execution of an MPOMDP is carried out in the following steps [112]. At every time step $t$, each agent $i$ observes $z_i^t$ and communicates its own observation $z_i^t$ to all other agents. The agent then in return receives observations of others $z \setminus \{z_i\}$ and uses the joint observations $z^t$ and the previous joint action $a^{t-1}$ to update the new joint belief $b^t$ from (4.1). Finally, the agent looks up the joint action from the joint policy $\pi(b^t) = a^t$ and executes its individual action $a_i^t$.

Noting that the joint belief evolution of an MPOMDP (11) is described by a discrete-time system [9, 10, 7], we propose conditions based on DTBFs for safety analysis of discrete-time systems.

In [3], the barrier function method was extended to discrete-time dynamical systems. Unfortunately, with the latter formulation of the (reciprocal) barrier functions, we

cannot study the solutions of the discrete-time system outside of the invariant set, i.e., if the solution is on the boundary of the set or when it leaves the set. To overcome this difficulty, we next extend the notion of (zeroing) barrier functions to discrete-time systems.

We consider the following discrete-time system

$$x^{t+1} = f(x^t), \quad t \in \mathbb{N}_{\geq 0}, \tag{4.2}$$

with $f : \mathcal{X} \to \mathcal{X} \subset \mathbb{R}^n$ and a safe set defined in (1.2), we have the following definition of a DTBF.

**Definition 12** (Discrete-Time Barrier Function)**.** *For the discrete-time system* (4.2)*, the continuous function $h : \mathbb{R}^n \to \mathbb{R}$ is a discrete-time barrier function for the set $\mathcal{S}$ as defined in* (1.2)*, if there exists $\alpha \in \mathcal{K}$ satisfying $\alpha(r) < r$ for all $r > 0$ and a set $\mathcal{D}$ with $\mathcal{S} \subseteq \mathcal{D} \subset \mathbb{R}^n$ such that*

$$h(x^{t+1}) - h(x^t) \geq -\alpha(h(x^t)), \quad \forall x \in \mathcal{D}. \tag{4.3}$$

In fact, the discrete-time barrier function would more correctly be called a discrete-time zeroing barrier function per the literature, but we drop the "zeroing" as it is the only form of barrier function that will be considered throughout the rest of this section.

We can show that the existence of a DTBF is both necessary and sufficient for invariance.

**Theorem 17.** *Consider the discrete-time system* (4.2)*. Let $\mathcal{S} \subseteq \mathcal{D} \subset \mathbb{R}^n$ with $\mathcal{S}$ as described in* (1.2)*. Then, $\mathcal{S}$ is invariant if and only if there exists a DTBF as defined in Definition 12.*

*Proof.* We begin by proving the sufficiency. If (4.3) holds, we have $h(x^t) \geq (\mathrm{Id} - \alpha) \circ h(x^{t-1})$. Furthermore, since $\alpha(r) \leq r$, $(\mathrm{Id} - \alpha) \circ (r) < r$ and $(\mathrm{Id} - \alpha) \in \mathcal{K}$ [73]. For $t = 0$, we have

$$h(x^1) \geq (\mathrm{Id} - \alpha) \circ h(x^0).$$

Similarly, for $t = 1$, we have

$$h(x^2) \geq (\mathrm{Id} - \alpha) \circ h(x^1).$$

From the inequality obtained at $t = 0$, we obtain $h(x^2) \geq (\mathrm{Id} - \alpha) \cdot h(x^1) \geq (\mathrm{Id} - \alpha) \circ (\mathrm{Id} - \alpha) \circ h(x^0)$. Then, by induction, we conclude

$$h(x^t) \geq (\mathrm{Id} - \alpha)^t \circ h(x^0), \quad t \in \mathbb{N}, \tag{4.4}$$

where $(\mathrm{Id} - \alpha)^t$ denotes composition $t$ times.

At this point, we check invariance of $\mathcal{S}$ and asymptotic convergence (followed by invariance) of solutions to $\mathcal{S}$ for the two cases of $x^0 \in \mathcal{S}$ and $x^0 \in \mathcal{D} \setminus \mathcal{S}$, respectively.

For any $x^0 \in \mathcal{S}$, since $h(x^0) \geq 0$ by definition of $\mathcal{S}$ and $(\mathrm{Id} - \alpha) \in \mathcal{K}$, we can deduce from (4.4) that $h(x^t) \geq 0$ for all $t \in \mathbb{N}$, implying that $\mathcal{S}$ is invariant. This is simply because if $(\mathrm{Id} - \alpha) \circ (r) < r$, then there exist a constant $\gamma \in (0, 1)$ such that $(\mathrm{Id} - \alpha) \circ (r) \leq \gamma r$ and hence $(\mathrm{Id} - \alpha)^t \circ (r) \leq \gamma^t r$.

For any $x^0 \in \mathcal{D} \setminus \mathcal{S}$, inequality (4.4) implies that as $t \to \infty$, we have $h(x^t) \geq 0$. That is, all solutions of system (4.2) starting at $x^0 \in \mathcal{D} \setminus \mathcal{S}$, asymptotically converge to $\mathcal{S}$.

We next prove the converse direction. We set $\mathcal{S} = \mathcal{D}$ in Theorem 17. If $\mathcal{S}$ is forward invariant, we have $x^{t-1} \in \mathcal{S}$ and $x_t \in \mathcal{S}$ for all $t \in \mathbb{N}$. From the definition of the set $\mathcal{S}$, this implies that if $h(x^{t-1}) \geq 0$ then $h(x^t) \geq 0$ for all $t \in \mathbb{N}$. Furthermore, we claim if $\mathcal{S}$ is forward invariant, then $h(x^t) - h(x^{t-1}) \geq 0$. Because if $h(x^t) - h(x^{t-1}) \leq 0$ or alternatively $h(x^t) \leq h(x^{t-1})$, for all $x^{t-1} \in \partial\mathcal{S}$, we have $h(x^t) \leq 0$. That is, $x^t \notin \mathcal{S}$ which is a contradiction. Hence, we have $h(x^t) - h(x^{t-1}) \geq 0$ for all $t \in \mathbb{N}$.

For any $r \geq 0$, the set $\{x \in \mathbb{R}^n \mid 0 \leq h(x) \leq r\}$ is a compact subset of $\mathcal{S}$. Define a function $\alpha : [0, \infty) \to \mathbb{R}$ by

$$\alpha(r) = - \inf_{\{x' | 0 \leq h(x') \leq r\}} \inf_{\{x | 0 \leq h(x) \leq r\}} (h(x') - h(x)) .$$

Using the compactness property stated above and the fact that the difference of two continuous functions is continuous, $\alpha$ is a well-defined, non-decreasing function on $\mathbb{R}_{\geq 0}$ satisfying

$$h(x^t) - h(x^{t-1}) \geq -\alpha \circ h(x^{t-1}), \quad \forall x^{t-1} \in \mathcal{S}.$$

Moreover, if $\mathcal{S}$ is forward invariant, $h(x^t) \geq 0$ for all $t \in \mathbb{N}$. That is, $h(x^t) \geq (\mathrm{Id} - \alpha) \circ h(x^{t-1})$. Since $h(x^{t-1}) \geq 0$, $(\mathrm{Id} - \alpha) \cdot (r) > 0$, which implies $\alpha(r) < r$. This completes the proof. □

Note that a simple example of the function $\alpha$ in inequality (4.3) is when $\alpha$ is a constant $\alpha_0 \in (0, 1)$. In this case, from the proof of Theorem 17, we infer that

$$h(x^t) \geq (1 - \alpha_0)^t h(x^0), \quad t \in \mathbb{N}.$$

Indeed, we can control the rate of convergence of the DTBF by changing the value of $\alpha_0$.

As a technical remark, we point out that, unlike proving the converse BF theorem for continuous-time systems, we did not invoke Nagumo's theorem on the boundary of the set $\mathcal{S}$. This is simply because such condition does not imply invariance for discrete-time systems [27, Section 3.2].

It is often desirable to consider sets defined by Boolean composition of multiple barrier functions. In this regard, in [58], the authors proposed non-smooth barrier functions as a means to analyze composition of barrier functions by Boolean logic, i.e., $\vee$ (disjunction), $\wedge$ (conjunction), and $\neg$ (negation). Similarly, we use max to represent $\vee$ and min to show $\wedge$. In other words, if $x \in \{x \in \mathbb{R}^n \mid \max_{i=1,\ldots,k} h_i(x) \geq 0\}$, then there exists at least one $i_* \in \{1, \ldots, k\}$ such that $h_{i_*}(x) \geq 0$ and if $x \in \{x \in \mathbb{R}^n \mid \min_{i=1,\ldots,k} h_i(x) \geq 0\}$, then for all $i \in \{1, \ldots, k\}$ we have $h_{i_*}(x) \geq 0$. The negation operator is trivial and can be shown by checking if $-h$ satisfies the invariance property.

In the following, we propose conditions for checking Boolean compositions of barrier functions. Fortunately, since we are concerned with discrete time systems, this does not require non-smooth analysis.

In the context of DTBFs, we have the following result.

**Proposition 8.** *Let $\mathcal{S}_i = \{x \in \mathbb{R}^n \mid h_i(x) \geq 0\}$, $i = 1, \ldots, k$ denote a family of safe sets with the boundaries and interior defined analogous to $\mathcal{S}$ in* (1.2)*. Consider the discrete-time system* (4.2)*. If there exist a $\alpha \in \mathcal{K}$ satisfying $\alpha(r) < r$ for $\forall r > 0$ such that*

$$\min_{i=1,\ldots,k} h_i(x^{t+1}) - \min_{i=1,\ldots,k} h_i(x^t) \geq -\alpha\left(\min_{i=1,\ldots,k} h_i(x^t)\right) \tag{4.5}$$

*then the set $\{x \in \mathbb{R}^n \mid \wedge_{i=1,\ldots,k} h_i(x) \geq 0\}$ is forward invariant. Similarly, if there exist a $\alpha \in \mathcal{K}$ satisfying $\alpha(r) < r$ for all $r > 0$ such that*

$$\max_{i=1,\ldots,k} h_i(x^{t+1}) - \max_{i=1,\ldots,k} h_i(x^t) \geq -\alpha\left(\max_{i=1,\ldots,k} h_i(x^t)\right) \tag{4.6}$$

*then the set $\{x \in \mathbb{R}^n \mid \vee_{i=1,\ldots,k} h_i(x) \geq 0\}$ is forward invariant.*

*Proof.* We prove the case for conjunction and the proof for disjunction is similar. If (4.5) holds from the proof of Theorem 17, we can infer that

$$\min_{i=1,\dots,k} h_i(x^t) \geq (\mathrm{Id} - \alpha)^t \circ \left( \min_{i=1,\dots,k} h_i(x^0) \right).$$

That is, if $x^0 \in \{x \in \mathbb{R}^n \mid \min_{i=1,\dots,k} h_i(x) \geq 0\}$, then $\min_{i=1,\dots,k} h_i(x^t) \geq 0$ for all $t \in \mathbb{N}_{\geq 0}$, which in turn implies that $h_i(x) \geq 0$ for all $i \in \{1, \dots, k\}$. □

The next section shows how the results in this section can be used to provide safety assurances for MPOMDPs.

Since the states are not directly observable in MPOMDPs, we are interested in guaranteeing safety in a probabilistic setting in the joint belief space. To this end, we define the set of safe joint beliefs as

$$\mathcal{B}_s := \{b \in \mathcal{B} \mid h(b) \geq 0\}, \tag{4.7}$$

$$\mathrm{Int}(B_s) := \{b \in \mathcal{B} \mid h(b) > 0\}, \tag{4.8}$$

$$\partial\mathcal{B}_s := \{b \in \mathcal{B} \mid h(b) = 0\}, \tag{4.9}$$

where $h : \mathcal{B} \to \mathbb{R}$ is a given function. We denote by $\pi_n : \mathcal{B} \to \mathcal{A}$ a nominal joint policy mapping each joint belief into a joint action. We use subscript $n$ to denote variables corresponding to the nominal policy designed offline.

We are interested in solving the following problems for MPOMDPs.

**Problem 1.** *Given an MPOMDP as defined in Definition 11, a corresponding belief update* (4.1)*, and a safe joint belief set $\mathcal{B}_s$, design a sequence of actions $a^t$, $t \in \mathbb{N}_{\geq 0}$ such that $b^t \in \mathcal{B}_s$, $\forall t \in \mathbb{N}$ and the instantaneous rewards $r^t = \sum_{q^t \in Q} b(q^t) R(q^t, a^t)$ are maximized for all $t \in \mathbb{N}_{\geq 0}$.*

**Problem 2.** *Given an MPOMDP as defined in Definition 11, a corresponding belief update equation* (4.1)*, a safe joint belief set $\mathcal{B}_s$, and a nominal planning policy $\pi_n$, determine a sequence of actions $a^t$, $t \in \mathbb{N}_{\geq 0}$ such that $b^t \in \mathcal{B}_s$, $\forall t \in \mathbb{N}_{\geq 0}$ and the quantity $\|r^t - r_n^t\|^2$ is minimized for all $t \in \mathbb{N}_{\geq 0}$, where $r_n^t$ denotes the nominal immediate reward at time step $t$.*

As can be inferred from Problems 1 and 2, we seek to ensure safety in addition to motion planning at every time step. Such problems are prevalent in multi-agent robot applications, where safety is of significant importance, e.g., robots in performing tasks in the presence of human coworkers [105].

**Algorithm 6** The one-step greedy algorithm for finding the safe action at time $t$.

**Require:** System information $I$, $Q$, $A$, $T$, $R$, $Z$, $O$, safe belief set $\mathcal{B}_s$, current observation $z^t$, the past belief $b^{t-1}$

$i = 1$
**for** $i = 1, 2, \ldots, |A|$ **do**
    $b^t(q') = \frac{O(z^t|q',a(i)) \sum_{q\in Q} T(q'|q,a(i)) b^{t-1}(q)}{\sum_{q'\in Q} O(z^t|q',a(i)) \sum_{q\in Q} T(q'|q,a(i)) b^{t-1}(q)}$

    **if** $h(b^t) - h(b^{t-1}) \geq -\alpha(h(b^{t-1}))$ **then**
        $r(i) = \left(\sum_{q'\in Q} b(q') R(q', a(i))\right)$
    **end if**
**end for**
$i_* = \arg\max_{i=1,2,\ldots,|A|} r(i)$
**return** $a^* = a(i_*)$.

Next, we use the result in Theorem 17 to ensure safety of a team of heterogeneous autonomous agents described by an MPOMDP. To this end, we solve the following discrete optimization problem at each time step $t$:

$$a^* = \arg\max_{a\in A} \left(\sum_{q'\in Q} b(q') R(q', a)\right)$$
$$s.t.\ \ h(b(q')) - h(b(q)) \geq -\alpha(b(q)). \tag{4.10}$$

Algorithm 6 summarizes the steps involved in finding the safe action based on barrier functions at each time step. At every time step, the algorithm picks a joint action $a(i)$ from $|A|$ combinations of actions (recall that $\times_{i\in I} A_i = A$). For each joint action $a(i)$, it computes the next joint belief and checks whether if the next joint belief satisfies the safety constraint. If the safety constraint is satisfied, it computes the corresponding reward function $r(i)$ for the joint action $a(i)$. After checking all actions, the algorithm returns the joint action maximizing the reward function.

Algorithm 6 designs a safe and myopic optimal action at each time step based on the current observation and the belief state at the step before. Therefore, it does not require a full memory of past actions and observations. This synthesis algorithm for POMDPs parallels those using control barrier functions for dynamical systems wherein safety for all time and optimality at each time instance is required.

**Algorithm 7** The one-step greedy algorithm for finding the safe action at time $t$ when agents have different safety constraints.

**Require:** System information $I$, $Q$, $A$, $T$, $R$, $Z$, $O$, safe belief set $\mathcal{B}_s$, current observation $z^t$, the past belief $b^{t-1}$

$i = 1$
**for** $i = 1, 2, \ldots, |A|$ **do**
$b^t(q') = \frac{O(z^t|q',a(i)) \sum_{q\in Q} T(q'|q,a(i)) b^{t-1}(q)}{\sum_{q'\in Q} O(z^t|q',a(i)) \sum_{q\in Q} T(q'|q,a(i)) b^{t-1}(q)}$
**if** $h_k(b_k^t(q')) - h_k(b_k^{t-1}(q)) \geq -\alpha_k(h_k(b_k^{t-1}(q)))$ for all $k \in I$ **then**
$r(i) = \left(\sum_{q'\in Q} b(q') R(q', a(i))\right)$
**end if**
**end for**
$i_* = \arg\max_{i=1,2,\ldots,|A|} r(i)$
**return** $a^* = a(i_*)$.

Note that, if the safety requirement is defined by Boolean logic and we need to check either inequality (4.5) or (4.6), we can just replace the inequality in the "if" statement in Algorithm 6 with either inequality (4.5) or (4.6).

Furthermore, we remark that stability is not an issue in MPOMDP problems, since the beliefs evolve in the probabilistic belief simplex. However, we can encode instability in an MPOMDP problem as a set of bad states, that is, $\mathcal{B} \setminus \mathcal{B}_s$.

Each autonomous agent might have a different safety requirement, characterized by sets $\mathcal{B}_i$, $i \in I$, i.e., $\mathcal{B}_i$ is the safe set for agent $i$. Then, we just need to check the safety of each agent separately. We denote by $b_i$, $i \in I$, the subset of joint beliefs concerning agent $i$, e.g., beliefs showing the location of the agent. Algorithm 7 demonstrates how we can check the safety requirement of each agent separately at every time step.

In many real world multi-robot navigation scenarios, an offline policy for path planning exists (e.g., based on point-based methods [133]). However, such policy may not guarantee safety. We can use the barrier functions to design an online method for ensuring safety while remaining as much faithful as possible to the offline policy (see [28, 59] for analogous formulations for systems described by nonlinear differential equations).

Algorithm 8 illustrates how barrier functions can filter the agent actions to ensure safety. At every time step $t$, the algorithm first computes the next joint belief

**Algorithm 8** The one-step greedy algorithm for filtering the nominal policy with a safe action at every time-step $t$.

**Require:** System information $I$, $Q$, $A$, $T$, $R$, $Z$, $O$, nominal policy $\pi_n$, safe belief set $\mathcal{B}_s$, current observation $z^t$, the past belief $b^{t-1}$

$b^t(q') = \frac{O(z^t|q',a_n^t)\sum_{q\in Q} T(q'|q,a_n^t)b^{t-1}(q)}{\sum_{q'\in Q} O(z^t|q',a_n^t)\sum_{q\in Q} T(q'|q,a_n^t)b^{t-1}(q)}$
**if** $h(b^t) - h(b^{t-1}) < -\alpha(h(b^{t-1}))$ **then**
  $i = 1$
  **for** $i = 1, 2, \ldots, |A|$ **do**
    $b^t(q') = \frac{O(z^t|q',a(i))\sum_{q\in Q} T(q'|q,a(i))b^{t-1}(q)}{\sum_{q'\in Q} O(z^t|q',a(i))\sum_{q\in Q} T(q'|q,a(i))b^{t-1}(q)}$
    **if** $h(b^t) - h(b^{t-1}) \geq -\alpha(h(b^{t-1}))$ **then**
      $r(i) = \left(\sum_{q'\in Q} b(q')R(q', a(i))\right)$
    **end if**
  **end for**
  $i_* = \arg\min_{i=1,2,\ldots,|A|} \|r(i) - r_n^t\|^2$
**return** $a^* = a(i_*)$
**end if**
**return** $a^* = a_n^t$.

$b^t$ given the nominal action $a_n$ designed based on the nominal policy $\pi_n$. It then checks whether that action leads to a safe joint belief update (this is allowed since the existence of a DTBF $h$ satisfying (4.3) is both necessary and sufficient for safety). If yes, the algorithm returns $a_n$ for implementation. If no, the algorithm finds a safe joint action that minimally changes the immediate reward from the nominal immediate reward $r_n^t$ in a least squares sense.

### 4.1.1 Applications: Multi-Robot Exploration

To demonstrate our method, we consider a system of three heterogeneous robots exploring an unknown environment. The mission objective is to retrieve a sample located somewhere in the robots' vicinity. Each robot has different and limited capabilities to explore and observe the environment, so coordination and communication between the robots is required in order to complete the mission.

The robot team consists of a drone and two ground vehicles. The drone can rapidly explore the environment from above, but is unable to explore any covered or underground regions. The ground vehicles include a Rover Robotics Flipper, and a modified Segway. The Flipper is a small, tracked vehicle capable of traversing in tight spaces and rough terrain, while the Segway is a larger, wheeled robot without

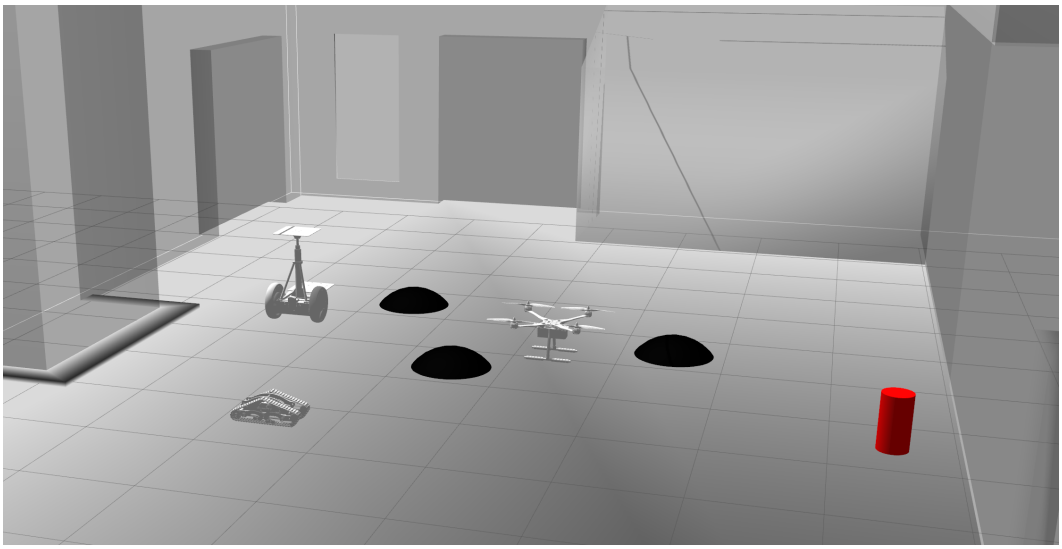


Figure 4.1: The agents, the obtacles (black), and the sample (red) in ROS simulation environment.

external sensing capabilities, whose purpose is to retrieve the sample.

The set of agents includes the UAV, $A_U$, the Flipper, $A_F$, and the Segway, $A_S$. These agents all inhabit a planar $n \times m$ grid, with the drone located two meters above the ground vehicles. The beliefs of the vehicle locations in this grid are updated after each action is completed, based on the previous belief and the observation made. The initial vehicle locations are known, and the remaining system states are given by the environmental model.

In order to capture the heterogeneity of the team, each grid in the environment has two states that measure the habitability of the grid for the Segway, as well as the probability that the grid in question contains the sample. These states are initialized to 0.5, and observations of these states can be made when the Flipper or the UAV are within a certain distance from the cell. The Flipper can make accurate observations about the traversability of the terrain, but cannot sense the location of the sample well. The drone is more suited to locating the sample, but less suited to gauging traversability.

The set of actions for each agent $A_i$ is the same, and consists of five actions: remaining in the same grid, and moving either forwards, backwards, left, or right to an adjacent grid. Thus, the total number of actions is 125. The transitions between states are handled by controllers on the low-level dynamics, and the transition probability $T$ when moving from one grid to another is modeled as a high chance to

move to the desired grid, and equal, smaller chances of landing in one of the eight grids adjacent to the desired grid.

The components of the reward function for the Flipper and the UAV are measures of how much information will be gained from moving in that direction. The reward function also includes a heavy reward for the Segway moving towards a cell likely to contain the sample, and a heavy cost towards moving to a potentially dangerous cell. The observations for each agent update the environment states based on the observation made (binary detection) and the beliefs of the vehicles locations.

The exploratory mission is concluded when the sample has been collected, resulting in a mission success. In terms of the system states, mission objective is satisfied when the Segway inhabits the same grid as the sample. If the Segway enters an uninhabitable region, this results in a mission failure. Thus, the safe set of beliefs is defined as all states in which the Segway does not coincide with an uninhabitable region. For this mission, given the partial observability constraint, we require that there is a 95% probability of the Segway entering a habitable grid with each action. It is important to note that safety for this problem does not depend on the entirety of the belief space. Thus, it is possible to verify safety without computing the beliefs of each of the states.

The simulation is carried out in a ROS environment as depicted in Figure 4.1. Occupancy grids are utilized to represent the states of the system, which are updated after each action is taken. When an action is initialized, the low-level dynamics of the vehicles are simulated, and a message is published when the action is complete. Utilizing the observations made by this action, the beliefs are updated, and a new joint action is generated. The simulation is ended when the true position of the robot inhabits the same grid as the true position of the sample, or when the robot coincides with an uninhabitable cell.

To demonstrate the efficacy of the policy filter, a near-optimal policy that violates the belief safety filter was passed through Algorithm 3. The trajectory of the Segway under this policy is shown in Figure 4.2. While the policy is successful in simulation, due to perfect control over the states, it does not meet the imposed requirements for probabilistic 95% safety.

The resulting trajectory of the Segway after the policy filter is shown in Figure 4.3. The first filtered action occurred at the time of the first image. While the desired trajectory of the Segway was to move towards the uninhabitable terrain, as shown

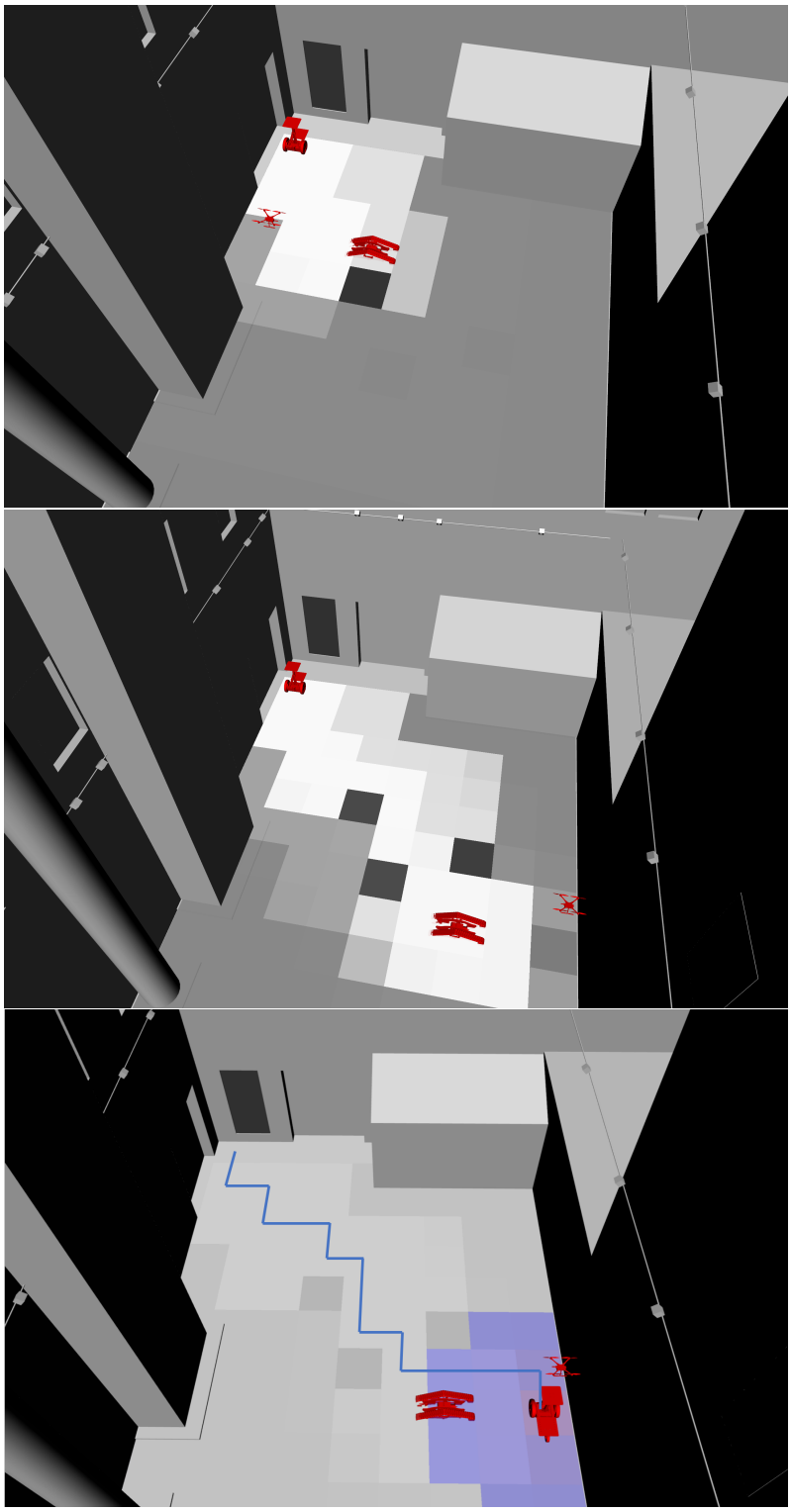


Figure 4.2: Implementation of the nominal policy. Darker cells represent unsafe terrain, and the blue cells in the third image represent the belief of the Segway location.

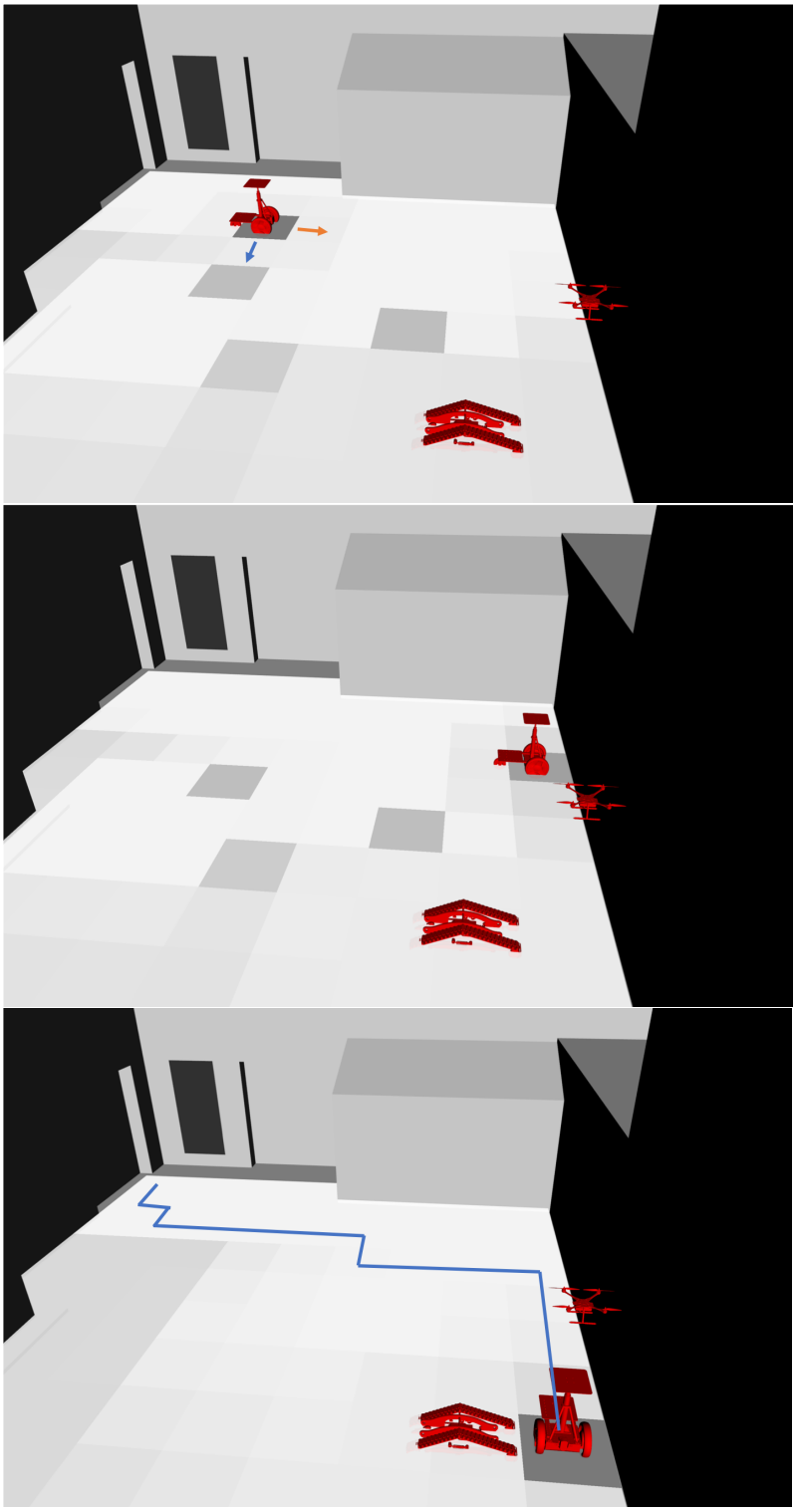


Figure 4.3: Implementation of the safety filter. The blue arrow in the first image represents the desired action, and the orange arrow is the filtered action.

in blue, the safety filter rejected this action. Instead, the action with the next highest reward, indicated by the orange arrow, was taken. This process continues, and the final trajectory is shown to move to the right wall of the building and then to the sample location. Thus, this filter was able to circumvent the unsafe policy, while still achieving the objective of the mission. While the resulting route is less optimal, it is a policy that could be implemented on a real system with realistic safety guarantees.

### 4.2 Finite-time DTBFs and DTL specifications

In this section, we employ discrete time barrier functions (DTBFs) to enforce safety/mission specifications in terms of LDTL specifications in Multi-agent POMDPs in *run time* with *minimum interference* (see Fig. 1). To this end, we represent the joint belief evolution of an MPOMDP as a discrete-time system [8]. The main contributions of this section are then as follows: (i) We enrich the DTBFs for enforcing invariance with finite-time DTBFs for assuring finite time reachability and (ii) we propose Boolean compositions of these finite-time DTBFs. (iii) We propose a LDTL safety-shield method based on one-step greedy algorithms [44, Chapter 16] to synthesize a safety-shield for an MPOMDP given a nominal planning policy. We illustrate the efficacy of the proposed approach by applying it to an exploration scenario of a team of heterogeneous robots in ROS simulation environment.

We formally describe high-level mission specifications that are defined in temporal logic. Temporal logic has been used as a formal way to allow the user to intuitively specify high-level specifications, in for example, robotics [89]. The temporal logic we use in this section can be used for specifying tasks for stochastic systems with partial state information. This logic is suitable for problems involving significant state uncertainty, in which the state is estimated on-line. The syntactically co-safe linear distribution temporal logic (scLDTL) describes co-safe linear temporal logic properties of probabilistic systems [74]. We consider a modified version to scLDTL, the linear distribution temporal logic (LDTL), which includes the additional temporal operator $\Box$ "always". The latter operator is important since it can be used to describe notions such as *safety*, *liveness*, and *invariance*.

LDTL has predicates of the type $\zeta < 0$ with $\zeta \in \mathcal{F}_Q = \{f \mid f : \mathcal{B} \to \mathbb{R}\}$, i.e., $\mathcal{F}_Q$ is the class of (nonlinear) functions mapping from the belief simplex into reals, and state predicates $q \in A$ with $A \in 2^Q$.

**Definition 13** (LDTL Syntax)**.** *An LDTL formula over predicates $\mathcal{F}_Q$ and $Q$ is*

*inductively defined as*

$$\varphi := A|\neg A|\zeta|\neg\zeta|\varphi \vee \varphi|\varphi \wedge \varphi|\varphi \mathfrak{U} \varphi|\bigcirc \varphi|\Diamond\varphi|\Box\varphi, \tag{4.11}$$

*where $A \in 2^Q$ is a set of states, $\zeta \in \mathcal{F}_Q$ is a belief predicate, and $\varphi$ is an LDTL formula.*

Satisfaction over pairs of hidden state paths and sequences of belief states can then be defined as follows.

**Definition 14** (LDTL Semantics)**.** *The semantic of LDTL formulae is defined over words $\omega \in (Q \times \mathcal{B})^\infty$. Let $(q^i, b^i)$ be the ith letter in $\omega$. The satisfaction of a LDTL formula $\varphi$ at position $i$ in $\omega$, denoted by $\omega^i \models \varphi$ is recursively defined as follows:*

- $\omega^i \models A$ *if* $q^i \in A$,
- $\omega^i \models \neg A$ *if* $q^i \notin A$,
- $\omega^i \models f$ *if* $f(b^i) < 0$,
- $\omega^i \models \neg f$ *if* $f(b^i) \geq 0$,
- $\omega^i \models \varphi_1 \wedge \varphi_2$ *if* $\omega^i \models \varphi_1$ *and* $\omega^i \models \varphi_2$,
- $\omega^i \models \varphi_1 \vee \varphi_2$ *if* $\omega^i \models \varphi_1$ *or* $\omega^i \models \varphi_2$,
- $\omega^i \models \bigcirc\varphi$ *if* $\omega^{i+1} \models \varphi$,
- $\omega^i \models \varphi_1 \mathfrak{U} \varphi_2$ *if there exists a* $j \geq i$ *such that* $\omega^j \models \varphi_2$ *and for all* $i \leq k < j$ *it holds that* $\omega^k \models \varphi_1$,
- $\omega^i \models \Diamond\varphi$ *if there exists* $j \geq i$ *such that* $\omega^j \models \varphi$, *and*
- $\omega^i \models \Box\varphi$ *if, for all* $j \geq i$, $\omega^j \models \varphi$.

*The word $\omega$ satisfies a formula $\varphi$, i.e., $\omega \models \varphi$, iff $\omega^0 \models \varphi$.*

Designing policies that guarantee LDTL formulas as defined in Definition 14 can only be carried if the system is linear and subject to Guassian noise [155]. We show here that DTBFs can be used to enforce LDTL formulas for any finite POMDP.

### 4.2.1 Finite-time DTBF

One class of problems we are interested in involve checking whether the solution of a discrete time system can reach a set in finite time. To this end, we define a finite time DTBF (see [144] for the continuous time variant).

**Definition 15** (Finite Time DTBF)**.** *For the discrete-time system* (4.2)*, the continuous function $\tilde{h} : \mathcal{B} \to \mathbb{R}$ is a finite time DTBF for the set $\mathcal{S}$ as defined in* (1.2)*, if there exist constants $0 < \rho < 1$ and $\varepsilon > 0$ such that*

$$\tilde{h}(b^{t+1}) - \rho\tilde{h}(b^t) \geq \varepsilon(1-\rho), \quad \forall b^t \in \mathcal{B}. \tag{4.12}$$

We then have the following result to check finite time reachability of a set for a discrete-time system.

**Theorem 18.** *Consider the discrete-time system* (4.2)*. Let $\mathcal{S} \subset \mathcal{B} \subset \mathbb{R}^n$ be as described in* (1.2)*. If there exists a finite time DTBF $\tilde{h}$ as in Definition 15, then for all $b^0 \in \mathcal{B} \setminus \mathcal{S}$, there exists a $t^* \in \mathbb{N}_{\geq 0}$ such that $b^{t^*} \in \mathcal{S}$. Furthermore,*

$$t^* \leq \log\left(\frac{\varepsilon - \tilde{h}(b^0)}{\varepsilon}\right) / \log\left(\frac{1}{\rho}\right), \tag{4.13}$$

*where the constants $\rho$ and $\varepsilon$ are as defined in Definition 15.*

*Proof.* We prove by induction. With some manipulation inequality (4.12) can be modified to $\tilde{h}(b^{t+1}) - \varepsilon \geq \rho\tilde{h}(b^t) - \rho\varepsilon = \rho\left(\tilde{h}(b^t) - \varepsilon\right)$. Thus, for $t = 0$, we have $\tilde{h}(b^1) - \varepsilon \geq \rho\left(\tilde{h}(b^0) - \varepsilon\right)$. For $t = 1$, we have $\tilde{h}(b^2) - \varepsilon \geq \rho\left(\tilde{h}(b^1) - \varepsilon\right) \geq \rho^2\left(\tilde{h}(b^0) - \varepsilon\right)$, where we used the inequality for $t = 0$ to obtain the last inequality above. Then, by induction, we have $\tilde{h}(b^t) - \varepsilon \geq \rho^t\left(\tilde{h}(b^0) - \varepsilon\right)$. Hence, $\tilde{h}(b^t) \geq \rho^t(\tilde{h}(b^0) - \varepsilon) + \varepsilon$. Since $0 < \rho < 1$ and $b^0 \in \mathcal{B} \setminus \mathcal{S}$, i.e., $\tilde{h}(b^0) < 0$, as $t$ increases $b^t$ approaches $\mathcal{S}$ because by definition $h(b^t) \geq 0$ implies $b^t \in \mathcal{S}$. Re-arranging the terms gives

$$\varepsilon - \tilde{h}(b^t) \leq \rho^t\left(\varepsilon - \tilde{h}(b^0)\right). \tag{4.14}$$

Since $b^0 \in \mathcal{B} \setminus \mathcal{S}$, i.e., $\tilde{h}(b^0) < 0$, $\varepsilon - \tilde{h}(b^0)$ is a positive number. Dividing both sides of (4.14) with the positive quantity $\varepsilon - \tilde{h}(b^0)$ yields $\frac{\varepsilon - \tilde{h}(b^t)}{\varepsilon - \tilde{h}(b^0)} \leq \rho^t$. Taking the logarithm of both sides of the above inequality gives $\log\left(\frac{\tilde{h}(b^t) - \varepsilon}{\tilde{h}(b^0) - \varepsilon}\right) \leq t\log(\rho)$, or equivalently

$$-\log\left(\frac{\tilde{h}(b^0) - \varepsilon}{\tilde{h}(b^t) - \varepsilon}\right) \leq -t\log(\frac{1}{\rho}).$$

Since $0 < \rho < 1$, $\log(\frac{1}{\rho})$ is a positive number. Dividing both sides of the inequality above with the negative number $-\log(\frac{1}{\rho})$ obtains $t \leq \log\left(\frac{\varepsilon-\tilde{h}(b^0)}{\varepsilon-\tilde{h}(b^t)}\right)/\log\left(\frac{1}{\rho}\right)$. Also, by definition, $b^t$ reaches $\mathcal{S}$ at least at the boundary at $t^*$ when $\tilde{h}(b^t) = 0$. Substituting $\tilde{h}(b^t) = 0$ in the last inequality for $t$ gives $t^* \leq \log\left(\frac{\varepsilon-\tilde{h}(b^0)}{\varepsilon}\right)/\log\left(\frac{1}{\rho}\right)$, which gives an upper bound for the first time $b^t \in \mathcal{S}$. □

**Proposition 9.** *Let $\mathcal{S}_i = \{b \in \mathcal{B} \mid \tilde{h}_i(b) \geq 0\}$, $i = 1, \ldots, k$ denote a family of sets defined analogous to $\mathcal{S}$ in* (1.2)*. Consider the discrete-time system* (4.2)*. If there exist constants $0 < \rho < 1$ and $\varepsilon > 0$ such that*

$$\min_{i=1,\ldots,k} \tilde{h}_i(b^{t+1}) - \rho \min_{i=1,\ldots,k} \tilde{h}_i(b^t) \geq \varepsilon(1-\rho),\ \forall b \in \mathcal{B}, \tag{4.15}$$

*then there exists*

$$t^* \leq \log\left(\frac{\varepsilon - \min_{i=1,\ldots,k} \tilde{h}_i(b^0)}{\varepsilon}\right)/\log\left(\frac{1}{\rho}\right) \tag{4.16}$$

*such that if $b^0 \in \mathcal{B} \setminus \bigcup_{i=1}^k \mathcal{S}_i$ then $b^{t^*} \in \left\{b \in \mathcal{B} \mid \wedge_{i=1,\ldots,k} \left(\tilde{h}_i(b) \geq 0\right)\right\}$. Similarly, the disjunction case follows by replacing* min *with* max *in* (4.15) *and* (4.16)*.*

*Proof.* We prove the conjunction case and the disjunction case follows the same lines. If (4.15) holds, from the proof of Theorem 18, we can infer that

$$\min_{i=1,\ldots,k} \tilde{h}_i(b^t) - \varepsilon \geq \rho^t\Big(\min_{i=1,\ldots,k} \tilde{h}_i(b^0) - \varepsilon\Big),$$

which implies that

$$t \leq \log\left(\frac{\varepsilon - \min_{i=1,\ldots,k} \tilde{h}_i(b^0)}{\varepsilon - \min_{i=1,\ldots,k} \tilde{h}_i(b^t)}\right)/\log\left(\frac{1}{\rho}\right).$$

If $b^0 \in \mathcal{B} \setminus \bigcup_{i=1}^k \mathcal{S}_i$, then by definition $\tilde{h}_i(b^0) < 0$, $i = 1, \ldots, k$. Hence, $\min_{i=1,\ldots,k} \tilde{h}_i(b^0) < 0$. Moreover, because $t$ is a positive integer,

$$\varepsilon - \min_{i=1,\ldots,k} \tilde{h}_i(b^0) \geq \varepsilon - \min_{i=1,\ldots,k} \tilde{h}_i(b^t).$$

That is, $\min_{i=1,\ldots,k} \tilde{h}_i(b^0) \leq \min_{i=1,\ldots,k} \tilde{h}_i(b^t)$ along the solutions $b^t$ of the discrete-time system (4.2). Furthermore, $b^t \in \left\{b \in \mathcal{B} \mid \wedge_{i=1,\ldots,k} \left(\tilde{h}_i(b) \geq 0\right)\right\}$ whenever $\min_{i=1,\ldots,k} \tilde{h}_i(b^t) \geq 0$. The upper-bound for this $t$ happens when $\min_{i=1,\ldots,k} \tilde{h}_i(b^t) = 0$, i.e., when all $\tilde{h}_i(b^t)$ are either positive or zero. This by definition implies that $b^t \in \left\{b \in \mathcal{B} \mid \wedge_{i=1,\ldots,k} \left(\tilde{h}_i(b) \geq 0\right)\right\}$. Then, setting $\min_{i=1,\ldots,k} \tilde{h}_i(b^t) = 0$ gives $t^* \leq \log\left(\frac{\varepsilon-\min_{i=1,\ldots,k} \tilde{h}_i(b^0)}{\varepsilon}\right)/\log\left(\frac{1}{\rho}\right)$. □

| LDTL Specification | DTBF Implementation |
|---|---|
| $\omega^i \models A$ | $h(b^i) = \sum_{s\in A} b^i(s) - 1$ |
| $\omega^i \models \neg A$ | $h(b^i) = \sum_{q\in Q\setminus A} b^i(s) - 1$ |
| $\omega^i \models f$ | $h(b^i) = -f(b^i) + \delta$ |
| $\omega^i \models \neg f$ | $h(b^i) = f(b^i)$ |
| $\omega^i \models \varphi_1 \wedge \varphi_2$ | $h(b^i) = \min\{h_1(b^i), h_2(b^i)\}$ |
| $\omega^i \models \varphi_1 \vee \varphi_2$ | $h(b^i) = \max\{h_1(b^i), h_2(b^i)\}$ |
| $\omega^i \models \bigcirc\varphi$ | $h(b^{i+1}) = h_\varphi(b)$ |
| $\omega^i \models \varphi_1 \mathfrak{U} \varphi_2$ | $h_2(b^j) < 0 \implies h = h_1(b^j), \forall j \geq i$ |
| $\omega^i \models \Diamond\varphi$ | $h(b^j) = \tilde{h}(b^j),\ \forall i \leq j \leq t^*$ |
| $\omega^i \models \Box\varphi$ | $h(b^j) = h_\varphi(b^j), \forall j \geq i$ |

Table 4.1: LDTL specifications and the DTBF implementation.

Now, we describe how the semantics of LDTL as given in Definition 14 can be represented as set invariance and reachability conditions over the belief simplex. The structure of the DTBFs for each specification are summarized in Table 4.1.

We describe each row of the table as follows. (1) $\omega^i \models A \subset Q$: can be encoded as verifying whether $q^i \in A$. In the belief simplex, this is equivalent to checking whether $b^i \in \mathcal{B}_s = \{b^i \in \mathcal{B} \mid \sum_{q\in A} b^i(q) \geq 1\}$, which can be checked by considering the DTBF $h(b^i) = \sum_{q\in A} b^i(q) - 1$. (2) $\omega^i \models \neg A \subset Q$: can be cast as checking whether $b^i \in \mathcal{B}_s = \{b^i \in \mathcal{B} \mid \sum_{q\in Q\setminus A} b^i(q) \geq 1\}$ by considering the DTBF $h(b^i) = \sum_{q\in Q\setminus A} b^i(q) - 1$. (3),(4) $\omega^i \models f$ and $\omega^i \models \neg f$: these formulas are defined in the belief space, since $\omega^i \models f$ implies $f(b^i) < 0$ and $\omega^i \models \neg f$ implies $f(b^i) \geq 0$. They can be checked by considering DTBFs $h(b^i) = -f(b^i) + \delta$ with $0 < \delta << 1$ for $\omega^i \models f$ and $h(b^i) = f(b^i)$ for $\omega^i \models \neg f$. (5),(6) $\omega^i \models \varphi_1 \wedge \varphi_2$ and $\omega^i \models \varphi_1 \vee \varphi_2$: can be implemented by Boolean composition of the barrier functions. (7) $\omega^i \models \bigcirc\varphi$: can be implemented by checking whether $\varphi$ is satisfied in the next step. (8) $\omega^i \models \varphi_1 \mathfrak{U} \varphi_2$: can be enforced by checking whether formula $\varphi_1$ is satisfied until $\varphi_2$. To this end, we can check whether formula $\varphi_2$ is not satisfied at every time step $i$ by checking inequality $h_2(b) < 0$ where $h_2$ is the DTBF for formula $\varphi_2$. If $\varphi_2$ is not satisfied, then $\varphi_1$ is checked via a corresponding DTBF $h_1$. (9) $\omega^i \models \Diamond\varphi$: can be checked using the finite time DTBF given by Theorem 18. Note that the property is checked until $t^*$, since after $t^*$ the formula $\varphi$ is ensured to hold. (10) $\omega^i \models \Box\varphi$: can be simply enforced by checking whether $\varphi$ is satisfied for all time using the corresponding DTBF $h_\varphi$.

### 4.2.2 Applications: LDTL specifications on multi-agent simulation

We take the identical simulation environment from the previous application.

The first mission objective given to the Segway (located at $q_S$) is to not collide with the Flipper (located at $q_F$) with probability 0.9. This can be represented as $\Box\neg f_1$, for $f_1 = 0.1 - b(q_S)b(q_F)$. The next requirement is that the Segway must not collide with the three obstacles (located at $q_{o_i}$, $i = 1, 2, 3$), again with probability 0.9. This can be enforced with the formula $\Box\neg f_2$, for $f_2 = 0.1 - \wedge_{i=1}^{3} b(q_S)b(q_{o_i})$. To enforce these objectives as a single specification, the formula is $\Box\neg(f_1 \vee f_2)$. Note that, if the agents meet this specification at time $t = 0$, then there always exists an action that meets this specification, as the agents can stop or remain in place.

In order to enforce LDTL formula $\Box\neg(f_1 \vee f_2)$, we use the DTBF $h(b) = \min(f_1, f_2)$, where we used De Morgan's laws to obtain $\neg(f_1 \vee f_2) = \neg f_1 \wedge \neg f_2$, the fourth row of Table 4.1, and Proposition 9. Figure 4.4(b) illustrates the safety shield enforcing this specification over the beliefs of the agents and the obstacles. Despite the obstacle being one cell away from the desired Segway position, the uncertainty stemming from the Flipper measurements of the obstacles as well as the state estimator of the Segway prevent the robot from moving into the desired position.

While the safety shield is able to keep the robots safe under this specification, there is no requirement of progression towards the objective, to retrieve the sample (located at $q_G$). Retrieving the sample with probability 0.5 can be written as $\Diamond f_3$, with $f_3 = 0.5 - b(q_S)b(q_G)$.

Combining all three objectives into one yields the final mission specification, given by the formula:

$$\varphi = \Box\neg(f_1 \vee f_2) \wedge \Diamond f_3, \tag{4.17}$$

which ensures that the Segway *always* avoids the Flipper and the three obstacles with more than 0.90 probability and *eventually* reaches the goal with more than 0.5 probability.

The finite time DTBF $\tilde{h}(b) = b(q_S)b(q_G) - 0.5$ where we used the third and ninth rows of Table 4.1 to enforce $\Diamond f_3$. For the finite time DTBF condition (4.12), the parameters $\rho$ and $\epsilon$ must be set to tune how quickly the set must be reached. To allow for more freedom of operation, we choose $\rho = 0.99$ and $\epsilon = 0.1$.

Figure 4.4(c-d) shows the results in our high-fidelity simulation environment. In particular, Figure 4.4(d) depicts the evolution of the DTBFs over the whole experiment. As it can be seen, for the nominal policy, the Segway fails to satisfy the

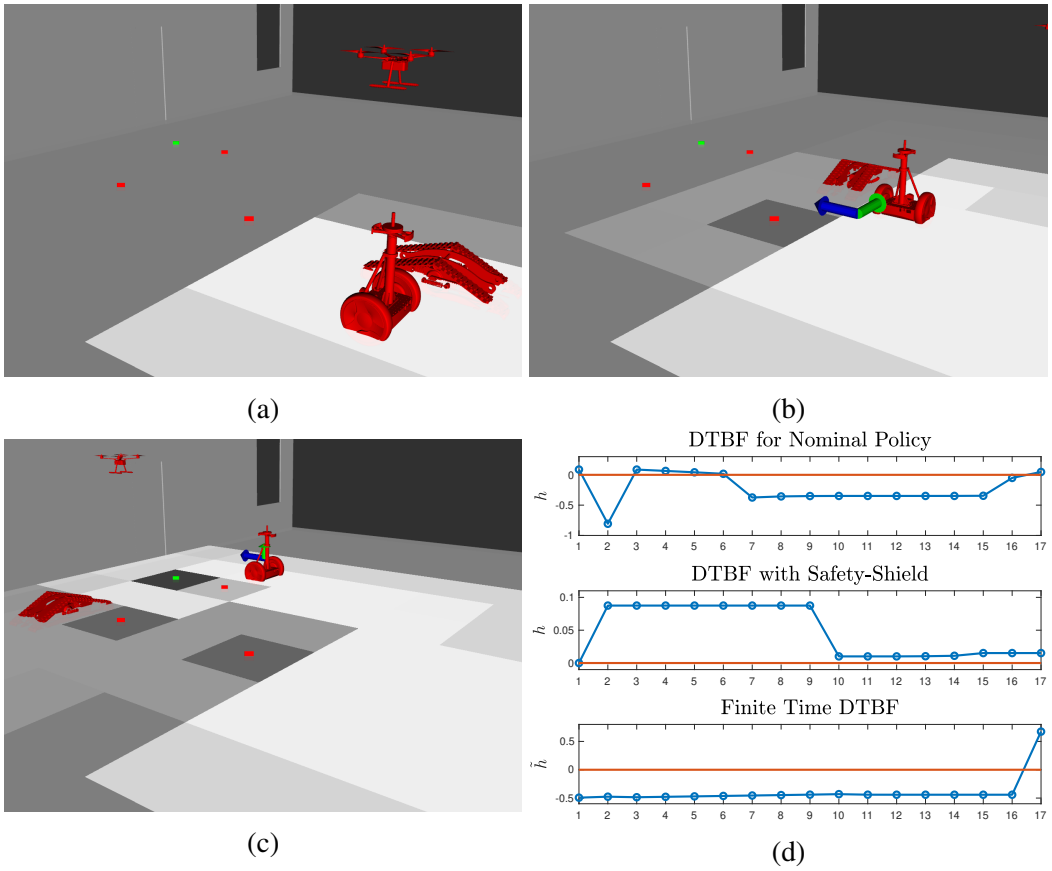



Figure 4.4: Simulation results of the multi-agent system. **(a)** The initial positions of the three agents, obstacles (red), and sample (green). **(b)** Example of the nominal action (blue) being overwritten by the safety shield (green). **(c)** Updated costmaps reflected in grayscale after a longer period of exploration. **(d)** The plots of the DTBFs for the experiment, as explained above.

mission specifications (since $h$ becomes negative in many instances). However, with the safety-shield the satisfaction of mission specifications is guaranteed ($h$ is always positive). Furthermore, the finite time DTBF becomes positive at the end of the experiment, which shows that the *eventually* specification in (4.17) is satisfied.

## 4.3 Accounting for uncertainty: risk control barrier functions

Autonomous robotic systems are being increasingly deployed in real-world settings where safety is critical. With this transition to practice, the associated risk that stems from unknown and unforeseen circumstances is correspondingly on the rise [149]. In the context of safety-critical scenarios, such as those found in aerospace and human-robot applications, it is essential that decision making accounts for risk. These risks are often associated with uncertainty due to extremely intricate nonlinear

dynamics, e.g. bipedal robots [121], and/or extreme unstructured environments, e.g. subterranean or extraterrestrial exploration [127].

Mathematically speaking, risk can be quantified in numerous ways, such as chance constraints [114, 157], exponential utility functions [82], and distributional robustness [165]. However, applications in autonomy and robotics require more "nuanced assessments of risk" [99]. Artzner *et. al.* [20] characterized a set of natural properties that are desirable for a risk measure, called a coherent risk measure, and have obtained widespread acceptance in finance and operations research, among other fields. An important example of a coherent risk measure is the conditional value-at-risk (CVaR) that has received significant attention in decision making problems, such as Markov decision processes (MDPs) [40, 39, 118, 24]. For stochastic discrete-time dynamical systems, a model predictive control technique with coherent risk objectives was proposed in [137], wherein the authors also proposed Lyapunov condition for risk-sensitive exponential stability. Moreover, a method based on stochastic reachability analysis was proposed in [32] to estimate a CVaR-safe set of initial conditions via the solution to an MDP.

Our approach to risk-sensitive safety is based on a special class of control barrier functions. Recently, for a class of stochastic (Ito) differential equations, safety in probability and statistical mean was studied in [41, 130] via stochastic barrier functions.

This section goes beyond the conventional notions of safety in probability and statistical mean through the use of coherent risk measures. To this end, for discrete-time systems subject to stochastic uncertainty, we define safety and finite-time reachability in the risk-sensitive sense, i.e., in the context of the worst possible realizations, via coherent risk measures. We then propose *risk control barrier functions* (RCBFs), together with finite-time RCBFs, as tools to enforce risk-sensitive safety and reachability, respectively. The main result of this section establishes that RCBFs ensure safety in a risk sensitive fashion. Finite-time RCBFs allow for the extension of this result to risk-sensitive reachability. Furthermore, for safe and goal sets defined as Boolean compositions of multiple function level-sets, we propose conditions that ensure safety and reachability of these sets based on RCBFs and their finite-time counterparts. Importantly, in all cases, the risk-sensitive controllers are designed to minimally invasive with respect to a given system legacy controller. We show the efficacy of our approach through simulation on a nonlinear cart-pole system (see Figure 4.5).

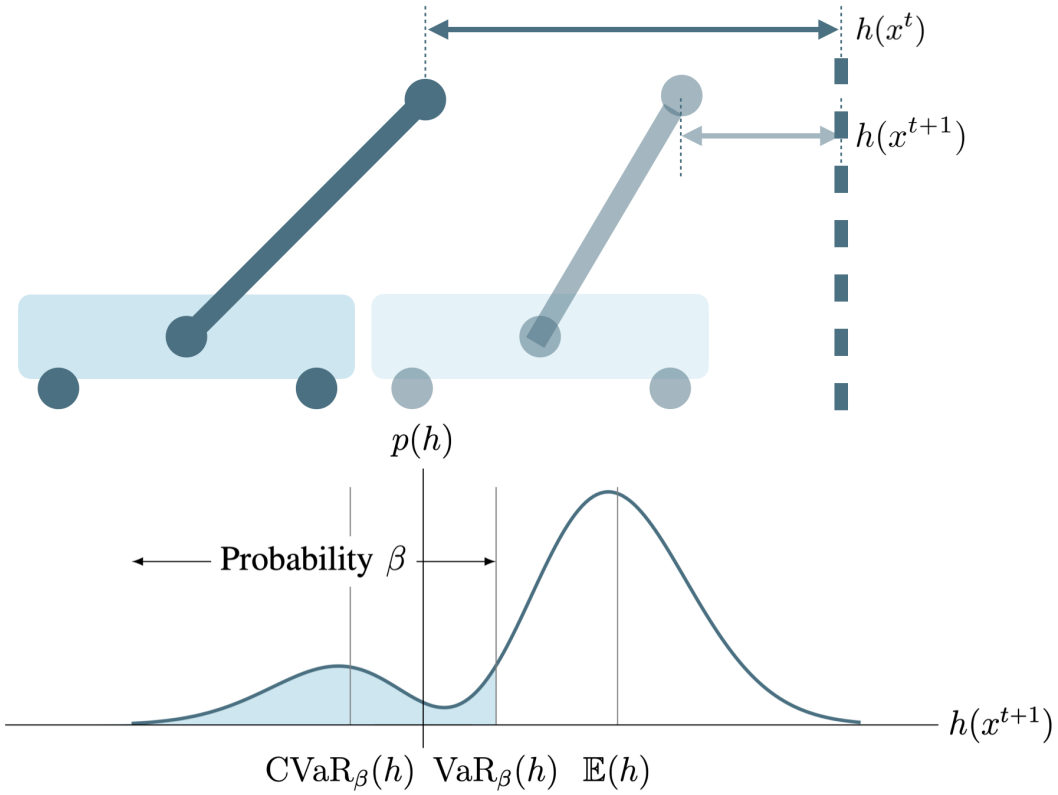



Figure 4.5: The value of the safe-set $h(x^t)$ is known at time $t$, but stochastic uncertainty makes $h(x^{t+1})$ a random variable. We must pick $u^t$ such that $h(x^{t+1})$ is safe subject to a risk measure taken over the worst $\beta$ probability.

### 4.3.1 Coherent risk measures

The goal of this subsection is to introduce conditional risk measures with a view toward defining risk control barrier functions subsequently. In this context, consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, a filtration $\mathcal{F}_0 \subset \cdots \mathcal{F}_N \subset \mathcal{F}$, and an adapted sequence of random variables $h^t$, $t = 0, \ldots, N$, where $N \in \mathbb{N}_{\geq 0} \cup \{\infty\}$. For $t = 0, \ldots, N$, we further define the spaces $\mathcal{H}_t = \mathcal{L}_p(\Omega, \mathcal{F}_t, \mathbb{P})$, $p \in [0, \infty)$, $\mathcal{H}_{t:N} = \mathcal{Z}_t \times \cdots \times \mathcal{Z}_N$ and $\mathcal{H} = \mathcal{H}_0 \times \mathcal{H}_1 \times \cdots$. We assume that the sequence $\boldsymbol{h} \in \mathcal{H}$ is almost surely bounded (with exceptions having probability zero), *i.e.*, $\operatorname{ess\,sup}_t |h^t(\omega)| < \infty$. In order to describe how one can evaluate the risk of sub-sequence $h_t, \ldots, h_N$ from the perspective of stage $t$, we require the following definitions.

**Definition 16** (Conditional Risk Measure)**.** *A mapping $\rho_{t:N} : \mathcal{H}_{t:N} \to \mathcal{H}_t$, where $0 \leq t \leq N$, is called a* **conditional risk measure***, if it has the following monotonicity property:*

$$\rho_{t:N}(\boldsymbol{h}) \leq \rho_{t:N}(\boldsymbol{h}'), \quad \forall \boldsymbol{h}, \forall \boldsymbol{h}' \in \mathcal{H}_{t:N} \text{ such that } \boldsymbol{h} \leq \boldsymbol{h}'.$$

*A* **dynamic risk measure** *is a sequence of conditional risk measures* $\rho_{t:N} : \mathcal{H}_{t:N} \to \mathcal{H}_t$, $t = 0, \dots, N$.

One fundamental property of dynamic risk measures is their consistency over time [128, Definition 3]. That is, if $h$ will be as good as $h'$ from the perspective of some future time $\theta$, and they are identical between times $\tau$ and $\theta$, then $h$ should not be worse than $h'$ from the perspective at time $\tau$.

If a risk measure is time-consistent, we can define the one-step conditional risk measure $\rho_t : \mathcal{H}_t \to \mathcal{H}_{t-1}$, $t = 0, \dots, N-1$ as follows:

$$\rho_t(h^t) = \rho_{t-1,t}(0, h^t), \tag{4.18}$$

and for all $t = 1, \dots, N$, we obtain:

$$\begin{aligned}\rho_{t,N}(h^t, \dots, h^N) = \rho_t\big(h^t + \rho_{t+1}(h^{t+1} + \rho_{t+2}(h^{t+2} + \cdots \\ + \rho_{N-1}\left(h^{N-1} + \rho_N(h^N)\right)\cdots))\big).\end{aligned} \tag{4.19}$$

Note that the time-consistent risk measure is completely defined by one-step conditional risk measures $\rho_t$, $t = 0, \dots, N-1$ and, in particular, for $t = 0$, (4.19) defines a risk measure of the entire sequence $\boldsymbol{h} \in \mathcal{H}_{0:N}$. This leads to the notion of a coherent risk measure.

**Definition 17** (Coherent Risk Measure)**.** *We call the one-step conditional risk measures* $\rho_t : \mathcal{H}_{t+1} \to \mathcal{H}_t$, $t = 1, \dots, N-1$ *as in* (4.19) *a* **coherent risk measure** *if it satisfies the following conditions*

- ***Convexity:*** $\rho_t(\lambda h + (1-\lambda)h') \leq \lambda\rho_t(h) + (1-\lambda)\rho_t(h')$, *for all* $\lambda \in (0,1)$ *and all* $h, h' \in \mathcal{H}_t$;
- ***Monotonicity:*** *If* $h \leq h'$ *then* $\rho_t(h) \leq \rho_t(h')$ *for all* $h, h' \in \mathcal{H}_t$;
- ***Translational Invariance:*** $\rho_t(h + h') = c + \rho_t(h')$ *for all* $h \in \mathcal{H}_{t-1}$ *and* $h' \in \mathcal{H}_t$;
- ***Positive Homogeneity:*** $\rho_t(\beta h) = \beta\rho_t(h)$ *for all* $h \in \mathcal{H}_t$ *and* $\beta \geq 0$.

All risk measures studied in this section are time-consistent coherent risk measures. Concretely, we briefly review two examples of coherent risk measures.

**Total Conditional Expectation:** The simplest risk measure is the total conditional expectation given by

$$\rho_t(h^t) = \mathbb{E}\left[h^t \mid \mathcal{F}_{t-1}\right]. \tag{4.20}$$

It is easy to see that total conditional expectation satisfies the properties of a coherent risk measure as outlined in Definition 17. Unfortunately, total conditional expectation is agnostic to realization fluctuations of the stochastic variable $h$ and is only concerned with the mean value of $h$ at large number of realizations. Thus, it is a risk-neutral measure of performance.

**Conditional Value-at-Risk:** Let $h \in \mathcal{H}$ be a stochastic variable for which higher values are of interest[1]. For a given confidence level $\beta \in (0, 1)$, value-at-risk ($\text{VaR}_\beta$) denotes the $\beta$-quantile value of a stochastic variable $h \in \mathcal{H}$ described as $\text{VaR}_\beta(h) = \sup_{\zeta \in \mathbb{R}}\{\zeta \mid \mathbb{P}(h \leq \zeta) \leq \beta\}$. Unfortunately, working with VaR for non-normal stochastic variables is numerically unstable, optimizing models involving VaR are intractable in high dimensions, and VaR ignores the values of $h$ with probability less than $\beta$ [124].

In contrast, CVaR overcomes the shortcomings of VaR. CVaR with confidence level $\beta \in (0, 1)$ denoted $\text{CVaR}_\beta$ measures the expected loss in the $\beta$-tail given that the particular threshold $\text{VaR}_\beta$ has been crossed, i.e., $\text{CVaR}_\beta(h) = \mathbb{E}\left[h \mid h \leq \text{VaR}_\beta(h)\right]$. An optimization formulation for CVaR was proposed in [124] that we use in this section. That is, $\text{CVaR}_\beta$ is given by

$$\text{CVaR}_\beta(h) := -\inf_{\zeta \in \mathbb{R}} \mathbb{E}\left[\zeta + \frac{(-h-\zeta)_+}{\beta}\right]. \tag{4.21}$$

Note that the above formulation of CVaR is concerned with the left-tail of distributions (higher values of $h$ are preferred).

A value of $\beta \to 1$ corresponds to a risk-neutral case, i.e., $\text{CVaR}_1(h) = \mathbb{E}(h)$; whereas, a value of $\beta \to 0$ is rather a risk-averse case, i.e., $\text{CVaR}_0(h) = \text{VaR}_0(h) = \operatorname{ess\,inf}(h)$ [123]. Figure 4.5 illustrates these notions for an example $h$ variable with distribution $p(h)$.

### 4.3.2 Risk-Sensitive Safety and Reachability

We assume the robot dynamics of interest is described by a discrete-time stochastic system given by

$$x^{t+1} = f(x^t, u^t, w^t), \quad x^0 = x_0, \tag{4.22}$$

[1]For example, greater values of $h$ indicate safer performance as will be discussed in the next section.

where $t \in \mathbb{N}_{\geq 0}$ denotes the time index, $x \in \mathcal{X} \subset \mathbb{R}^n$ is the state, $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, $w \in \mathcal{W}$ is the stochastic uncertainty/disturbance, and the function $f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{W} \to \mathbb{R}^n$. We assume that the initial condition $x_0$ is deterministic and that $|\mathcal{W}|$ is finite, *i.e.,* $\mathcal{W} = \{w_1, \ldots, w_{|\mathcal{W}|}\}$. At every time-step $t$, for a state-control pair $(x^t, u^t)$, the process disturbance $w^t$ is drawn from set $\mathcal{W}$ according to the probability mass function $p(w) = [p(w_1), \ldots, p(w_{|\mathcal{W}|})]^T$, where $p(w_i) := \mathbb{P}(w^t = w_i)$, $i = 1, 2, \ldots, |\mathcal{W}|$. Note that the probability mass function for the process disturbance is time-invariant, and that the process disturbance is independent of the process history and of the state-control pair $(x^t, u^t)$.

Note that, in particular, system (4.22) can capture stochastic hybrid systems, such as Markovian Jump Systems [169].

In the presence of stochastic uncertainty $w$, assuring almost sure (with probability one) invariance or safety may not be feasible. Moreover, enforcing safety in expectation is only meaningful if the law of large numbers can be invoked and we are interested in the long term performance, independent of the realization fluctuations. In this work, instead, we propose safety in the dynamic coherent risk measure sense with conditional expectation as an special case.

**Definition 18** ($\rho$-Safety)**.** *Given a safe set $\mathcal{S}$ as given in* (1.2) *and a time-consistent, dynamic coherent risk measure $\rho_{0:t}$ as described in* (4.19)*, we call the solutions to* (4.22)*, starting at $x_0 \in \mathcal{S}$, $\rho$-**safe** if and only if*

$$\rho_{0,t}\left(0, 0, \ldots, h(x)\right) \geq 0, \quad \forall t \in \mathbb{N}_{\geq 0}. \tag{4.23}$$

In order to understand (4.23), consider the case where $\rho$ is the conventional total expectation. Then, (4.23) implies safety in expectation. As mentioned earlier, the definition of safety for general coherent risk measures goes beyond the traditional total expectation.

Another interesting property we study in this section arises when $x_0 \in \mathcal{X} \setminus \mathcal{S}$. That is, when instead of safety, we are interested in reaching a set of interest in finite time.

**Definition 19** ($\rho$-Reachability)**.** *Consider system* (4.22) *with initial condition $x_0 \in \mathcal{X} \setminus \mathcal{S}$. Given a set $\mathcal{S}$ as given in* (1.2) *and a time-consistent, dynamic coherent risk measure $\rho_{0:t}$ as described in* (4.19)*, we call the set $\mathcal{S}$ $\rho$-**reachable**, if and only if there exists a constant $t^*$ such that*

$$\rho_{0,t^*}\left(0, 0, \ldots, h(x)\right) \geq 0. \tag{4.24}$$

### 4.3.3 Risk control barrier functions

In order to check and enforce risk sensitive safety, i.e., $\rho$-safety, we introduce *risk control barrier functions*. We then extend these to a finite-time variation, which allows us to establish risk-sensitive reachability, i.e., $\rho$-reachability.

**Definition 20** (Risk Control Barrier Function)**.** *For the discrete-time system* (4.22) *and a dynamic coherent risk measure $\rho$, the continuous function $h : \mathbb{R}^n \to \mathbb{R}$ is a* **risk control barrier function** *for the set $\mathcal{S}$ as defined in* (1.2)*, if there exists a convex $\alpha \in \mathcal{K}$ satisfying $\alpha(r) < r$ for all $r > 0$ such that*

$$\rho(h(x^{t+1})) \geq \alpha(h(x^t)), \quad \forall x^t \in \mathcal{X}. \tag{4.25}$$

Note that a simple choice for the function $\alpha$ is $\alpha = \alpha_0$, where $\alpha_0 \in (0, 1)$ is a constant.

In the first main contribution of the section, we demonstrate that the existence of an RCBF implies invariance/safety in the coherent risk measure.

**Theorem 19.** *Consider the discrete-time system* (4.22) *and the set $\mathcal{S}$ as described in* (1.2)*. Let $\rho$ be a given coherent risk measure. Then, $\mathcal{S}$ is $\rho$-safe if there exists an RCBF as defined in Definition 20.*

*Proof.* The proof is carried out by induction and using the properties of a coherent risk measure as outlined in Definition 17. If (4.25) holds, for $t = 0$, we have

$$\rho(h(x^1)) \geq \alpha(h(x_0)). \tag{4.26}$$

Similarly, for $t = 1$, we have

$$\rho(h(x^2)) \geq \alpha(h(x^1)). \tag{4.27}$$

Since $\rho$ is monotone, composing both sides of (4.27) with $\rho$ does not change the inequality and we obtain

$$\rho \circ \rho(h(x^2)) \geq \rho(\alpha(h(x^1))). \tag{4.28}$$

Since $\alpha$ is a convex function, from Theorem 3 in [38] (Jensen's Inequality for coherent risk measures), we obtain[2]

$$\rho \circ \rho(h(x^2)) \geq \rho(\alpha(h(x^1))) \geq \alpha(\rho(h(x^1))).$$

[2]In particular, if $\alpha \in (0, 1)$ is a constant, from positive homogeneity property of $\rho$, we have

$$\rho \circ \rho(h(x^2)) \geq \rho(\alpha h(x^1)) = \alpha\rho(h(x^1)).$$

Then, using inequality (4.26), we have

$$\rho \circ \rho(h(x^2)) \geq \alpha(\rho(h(x^1))) \geq \alpha \circ \alpha(h(x_0)).$$

Therefore, by induction, at time $t$, we can show that $\rho^t(h(x^t)) \geq \alpha^t(h(x_0))$. The left-hand side of the above inequality is equal to $\rho_{0,t}(0, \ldots, h(x^t))$. Hence,

$$\rho_{0,t}(0, \ldots, h(x^t)) \geq \alpha^t(h(x_0)). \tag{4.29}$$

If $x_0 \in \mathcal{S}$, from the definition of the set $\mathcal{S}$, we have $h(x_0) \geq 0$. Since $\alpha \in \mathcal{K}$, then we can infer that (4.23) holds. Thus, the system is $\rho$-safe. □

Note that, in the case when $x_0 \in \mathcal{X} \setminus \mathcal{S}$, the existence of an RCBF implies asymptotic convergence to the set $\mathcal{S}$ in the coherent risk measure $\rho$. This can be inferred from (4.29). In fact, if $\alpha(r) < r$, then there exist a constant $\delta \in (0, 1)$ such that $\alpha(r) \leq \delta r$ and hence

$$\alpha^t(r) \leq \delta^t r, \quad t \in \mathbb{N}_{\geq 0}. \tag{4.30}$$

If $x_0 \in \mathcal{X} \setminus \mathcal{S}$, then $h(x_0) < 0$. However, from (4.30), as $t \to \infty$, $\alpha \circ \cdots \circ \alpha(r) \to 0$, since the compositions of class $\mathcal{K}$ functions is also class $\kappa$ (hence non-negative). We then obtain $\rho_{0,t}(0, \ldots, h(x^t)) \geq 0$, which implies that the solutions become $\rho$-safe.

In practice, we are often interested in satisfying system specifications characterized by the set $\mathcal{S}$ in finite time. To this end, we define finite-time RCBFs.

**Definition 21** (Finite-Time RCBF)**.** *For the discrete-time system* (4.22) *and a dynamic coherent risk measure $\rho$, the continuous function $h : \mathcal{X} \to \mathbb{R}$ is a* **finite-time RCBF** *for the set $\mathcal{S}$ as defined in* (1.2)*, if there exist constants $0 < \gamma < 1$ and $\varepsilon > 0$ such that*

$$\rho(h(x^{t+1})) - \gamma h(x^t) \geq \varepsilon(1 - \gamma), \quad \forall x^t \in \mathcal{X}. \tag{4.31}$$

In the second key contribution of the section, we show that the existence of a finite-time RCBF implies $\rho$-reachability.

**Theorem 20.** *Consider the discrete-time system* (4.22) *and a dynamic coherent risk measure $\rho$. Let $\mathcal{S} \subset \mathcal{X}$ be as described in* (1.2)*. If there exists a finite-time RCBF $h : \mathcal{X} \to \mathbb{R}$ as in Definition 21, then for all $x^0 \in \mathcal{X} \setminus \mathcal{S}$, there exists a $t^* \in \mathbb{N}_{\geq 0}$ such that $\mathcal{S}$ is $\rho$-reachable, i.e., inequality* (4.24) *holds. Furthermore,*

$$t^* \leq \log\left(\frac{\varepsilon - h(x^0)}{\varepsilon}\right) / \log\left(\frac{1}{\gamma}\right), \tag{4.32}$$

*where the constants $\gamma$ and $\varepsilon$ are as defined in Definition 21.*

*Proof.* Similar to the proof of Theorem 19, we use induction and properties of coherent risk measures. We prove by induction. From (4.31), we have $\rho(h(x^{t+1})) - \varepsilon \geq \gamma h(x^t) - \gamma\varepsilon = \gamma\left(h(x^t) - \varepsilon\right)$. Hence, for $t = 0$, we have

$$\rho(h(x^1)) - \varepsilon \geq \gamma(h(x_0) - \varepsilon). \tag{4.33}$$

For $t = 1$, we have

$$\rho(h(x^2)) - \varepsilon \geq \gamma(h(x^1) - \varepsilon). \tag{4.34}$$

Since $\rho$ is monotone, composing both sides of the above inequality with $\rho$ does not change the inequality and we obtain

$$\rho \circ \rho(h(x^2) - \varepsilon) \geq \rho(\gamma(h(x^1) - \varepsilon)) = \gamma\rho(h(x^1) - \varepsilon),$$

where in the last equality we used the positive homogeneity property of $\rho$ since $\gamma \in (0, 1)$. Since $\varepsilon > 0$ is a constant, translational invariance property of $\rho$ yields

$$\rho \circ \rho(h(x^2)) - \varepsilon \geq \gamma(\rho(h(x^1)) - \varepsilon).$$

Moreover, from inequality (4.33), we infer

$$\rho \circ \rho(h(x^2)) - \varepsilon \geq \gamma(\rho(h(x^1)) - \varepsilon) \geq \gamma^2(h(x_0) - \varepsilon).$$

Thus, by induction, we see that at time step $t$, the following inequality holds:

$$\rho^t(h(x^t)) - \varepsilon \geq \gamma^t(h(x_0) - \varepsilon).$$

Taking $\varepsilon$ to the right-hand side and noting that the left-hand side of the above inequality is equal to $\rho_{0,t}(0, \ldots, h(x^t))$, we have the following inequality:

$$\rho_{0,t}(0, \ldots, h(x^t)) \geq \gamma^t(h(x_0) - \varepsilon) + \varepsilon. \tag{4.35}$$

Since $0 < \gamma < 1$ and $x^0 \in \mathcal{X} \setminus \mathcal{S}$, *i.e.,* $h(x^0) < 0$, as $t$ increases $x^t$ approaches $\mathcal{S}$ in the dynamic risk measure $\rho_{0,t}$, because by definition $h(x^t) \geq 0$ implies $x^t \in \mathcal{S}$. Hence, $\mathcal{S}$ is $\rho$-reachable in finite time.

by definition, $x^t$ reaches $\mathcal{S}$ at least at the boundary by $t^*$ when $\tilde{h}(x^t) = 0$. Substituting $\tilde{h}(x^t) = 0$ in (4.35) yields

$$0 \geq \gamma^{t^*}(h(x_0) - \varepsilon) + \varepsilon, \tag{4.36}$$

where we used the fact that $\rho_{0,t}(0, \ldots, h(x^{t^*})) = \rho_{0,t}(0, \ldots, 0) = 0$. Re-arranging the term and noting that $h(x_0) \leq 0$ and therefore $h(x_0) - \varepsilon \leq 0$, we obtain

$$\frac{\varepsilon}{\varepsilon - h(x_0)} \leq \gamma^t.$$

Taking the logarithm of both sides of the above inequality gives $\log\left(\frac{\varepsilon}{\varepsilon - h(x_0)}\right) \leq t\log(\gamma)$, or equivalently

$$-\log\left(\frac{\varepsilon - h(x_0)}{\varepsilon}\right) \leq -t\log(\frac{1}{\gamma}).$$

Since $0 < \gamma < 1$, $\log(\frac{1}{\gamma})$ is a positive number. Dividing both sides of the inequality above with the negative number $-\log(\frac{1}{\gamma})$ obtains $t \leq \log\left(\frac{\varepsilon - \tilde{h}(b^0)}{\varepsilon}\right)/\log\left(\frac{1}{\rho}\right)$. □

The upper bound described by inequality (4.32) in Theorem 20 is dependent on the two parameter $\gamma$ and $\varepsilon$. In our experiments, we often fix $0 < \gamma < 1$ and carry out a line search over $\varepsilon$ until the finite-time RCBF condition (4.31) does not hold anymore. Then, we pick the corresponding $t^*$ as the upper-bound on the earliest time the solutions can enter the goal set $\mathcal{S}$.

We have proposed RCBFs and finite-time RCBFs as means to verify $\rho$-safety and $\rho$-reachability, respectively. We now propose conditions for verifying $\rho$-safety and $\rho$-reachability for Boolean compositions of several control barrier functions.

**Proposition 10.** *Let $\mathcal{S}_i = \{x \in \mathbb{R}^n \mid h_i(x) \geq 0\}$, $i = 1, \ldots, k$ denote a family of safe sets with the boundaries and interior defined analogous to $\mathcal{S}$ in* (1.2) *and $\rho$ be a given dynamic coherent risk measure. Consider the discrete-time system* (4.22)*. If there exist a $\alpha \in (0, 1)$ such that*

$$\rho\left(\min_{i=1,\ldots,k} h_i(x^{t+1})\right) \geq \alpha \min_{i=1,\ldots,k} h_i(x^t) \tag{4.37}$$

*then the set $\{x \in \mathbb{R}^n \mid \wedge_{i=1,\ldots,k}\,(h_i(x) \geq 0)\}$ is $\rho$-safe. Similarly, if there exist a $\alpha \in (0, 1)$ such that*

$$\rho\left(\max_{i=1,\ldots,k} h_i(x^{t+1})\right) \geq \alpha \max_{i=1,\ldots,k} h_i(x^t) \tag{4.38}$$

*then the set $\{x \in \mathbb{R}^n \mid \vee_{i=1,\ldots,k}\,(h_i(x) \geq 0)\}$ is $\rho$-safe.*

*Proof.* If (4.37) holds from the proof of Theorem 19, we can infer that

$$\rho^t\left(\min_{i=1,\ldots,k} h_i(x^t)\right) \geq \alpha^t \min_{i=1,\ldots,k} h_i(x^0).$$

That is, if $x^0 \in \{x \in \mathbb{R}^n \mid \min_{i=1,\ldots,k} h_i(x) \geq 0\}$, then

$$\rho_{0,t}\left(0, \ldots, \left(\min_{i=1,\ldots,k} h_i(x^t)\right)\right) = \rho^t\left(\min_{i=1,\ldots,k} h_i(x^t)\right) \geq 0$$

for all $t \in \mathbb{N}_{\geq 0}$. Let $h_{i^*}(x^t)$ be the smallest among $h_i(x^t)$, $i = 1, 2, ..., k$, i.e., it satisfies $h_j(x^t) \geq \cdots \geq h_{i^*}(x^t), \ \ \forall j \neq i^*$. From the assumption that $\rho$ is a coherent risk measure, we infer that $\rho$ is monotone (see Definition 16). Then, the latter inequality implies $\rho_{0,t}\left(0, \ldots, h_j(x^t)\right) \geq \cdots \geq \rho_{0,t}\left(0, \ldots, h_{i^*}(x^t)\right), \ \ \forall j \neq i^*$. Since $\rho_{0,t}\left(0, \ldots, \min_{i=1,\ldots,k} h_i(x^t)\right) = \rho_{0,t}\left(0, \ldots, h_{i^*}(x^t)\right) \geq 0$ for all $t \in \mathbb{N}_{\geq 0}$, we have

$$\rho_{0,t}\left(0, \ldots, h_j(x^t)\right) \geq \cdots \geq \rho_{0,t}\left(0, \ldots, h_{i^*}(x^t)\right) \geq 0,$$

for all $j \neq i^*$. Thus, $\rho_{0,t}\left(0, \ldots, h_i(x)\right) \geq 0$ for all $i \in \{1, \ldots, k\}$.

Similarly, if (4.38) holds, we can infer that

$$\rho^t\left(\max_{i=1,\ldots,k} h_i(x^t)\right) \geq \alpha^t \max_{i=1,\ldots,k} h_i(x^0).$$

Hence, using similar arguments as the proof of the conjunction case,

$$\rho^t\left(\max_{i=1,\ldots,k} h_i(x^t)\right) \geq 0$$

for all $t \in \mathbb{N}_{\geq 0}$. That is, there exists at least an $i \in \{1, \ldots, k\}$ for which

$$\rho_{0,t}\left(0, \cdots, h_i(x^t)\right) = \rho^t\left(h_i(x^t)\right) \geq 0.$$

□

We next propose conditions for risk-sensitive finite-time reachability of sets composed of Boolean compositions of several functions $h$ as described in (1.2).

**Proposition 11.** *Let $\mathcal{S}_i = \{x \in \mathbb{R}^n \mid h_i(x) \geq 0\}$, $i = 1, \ldots, k$ denote a family of sets with the boundaries and interior defined analogous to $\mathcal{S}$ in* (1.2) *and $\rho$ be a given dynamic coherent risk measure. Consider the discrete-time system* (4.22)*. If there exist constants $0 < \gamma < 1$ and $\varepsilon > 0$ such that*

$$\rho\left(\min_{i=1,\ldots,k} h_i(x^{t+1})\right) - \gamma \min_{i=1,\ldots,k} h_i(x^t) \geq \varepsilon(1-\gamma) \tag{4.39}$$

*then the set $\{x \in \mathbb{R}^n \mid \wedge_{i=1,\ldots,k}\left(h_i(x) \geq 0\right)\}$ is $\rho$-reachable. Then, there exists a constant $t^*$ satisfying*

$$t^* \leq \log\left(\frac{\varepsilon - \min_{i=1,\ldots,k} h_i(x^0)}{\varepsilon}\right) / \log\left(\frac{1}{\gamma}\right), \tag{4.40}$$

*such that if $x^0 \in \mathcal{X} \setminus \cup_{i=1,\ldots,k} \mathcal{S}_i$ then $x^{t^*} \in \cap_{i=1,\ldots,k} \mathcal{S}_i$. Similarly, the disjunction case follows by replacing* min *with* max *in* (4.39) *and* (4.40)*.*

*Proof.* We prove the conjunction case and the disjunction case follows the same lines. From (4.35) in proof of Theorem 20, we infer that if (4.39) holds, we have

$$\rho_{0,t}(0,\ldots,\min_{i=1,\ldots,k} h_i(x^t)) \geq \gamma^t(\min_{i=1,\ldots,k} h_i(x_0) - \varepsilon) + \varepsilon, \tag{4.41}$$

for all $t \in \mathbb{N}_{\geq 0}$. This implies that

$$t \leq \log\left(\frac{\varepsilon - \min_{i=1,\ldots,k} h_i(x^0)}{\varepsilon - \rho_{0,t}(0,\ldots,\min_{i=1,\ldots,k} h_i(x^t))}\right) / \log\left(\frac{1}{\gamma}\right). \tag{4.42}$$

If $x^0 \in \mathcal{X} \setminus \cup_{i=1,\ldots,k}\mathcal{S}_i$, then by definition $h_i(x^0) < 0$, $i = 1,\ldots,k$. Hence, $\min_{i=1,\ldots,k} h_i(x^0) < 0$. Furthermore, since $0 < \gamma < 1$, $\log(1/\gamma) > 0$. Therefore, because $t \in \mathbb{N}_{\geq 0}$, we have

$$\varepsilon - \rho_{0,t}(0,\ldots,\min_{i=1,\ldots,k} h_i(x^t)) \leq \varepsilon - \min_{i=1,\ldots,k} h_i(x^0).$$

That is,

$$\rho_{0,t}(0,\ldots,\min_{i=1,\ldots,k} h_i(x^t)) \geq \min_{i=1,\ldots,k} h_i(x^0)$$

along the solutions to (4.22). Furthermore, $x^t$ reaches $\wedge_{i=1,\ldots,k}\,(h_i(x) \geq 0)$ in the sense of $\rho$ whenever

$$\rho_{0,t}(0,\ldots,\min_{i=1,\ldots,k} h_i(x^t)) \geq 0.$$

In addition, the latest $x^t$ reaches $\wedge_{i=1,\ldots,k}\,(h_i(x) \geq 0)$ is when $x^t$ crosses the boundary of $\wedge_{i=1,\ldots,k}\,(h_i(x) \geq 0)$, which is by definition $\wedge_{i=1,\ldots,k}\,(h_i(x) = 0)$. Denoting this time $t^*$, we have

$$\rho_{0,t^*}(0,\ldots,\min_{i=1,\ldots,k} h_i(x^{t^*})) = \rho_{0,t^*}(0,\ldots,0) = 0.$$

Substituting the latter equality into (4.42) yields (4.40). □

### 4.3.4 Applications: Uncertain cart-pole system

In order to illustrate the results of these risk-aware guarantees, we apply our method in the case of the cart-pole, modeled as a nonlinear, control-affine discrete-time system.

$$x^{t+1} = x^t + \begin{bmatrix} v_x \\ \dot{\theta} \\ \frac{u^t + m_p \sin\theta(l\dot{\theta}^2 + g\cos\theta)}{m_c + m_p \sin^2\theta} \\ \frac{-u^t\cos\theta - m_p l\dot{\theta}^2\cos\theta\sin\theta - (m_c+m_p)g\sin\theta}{l(m_c + m_p\sin^2\theta)} \end{bmatrix} \Delta_t + w^t \tag{4.43}$$

The disturbance $w^t \in W$ enters the system linearly, and is described by a pmf over the states. This could include the modeling error from this Euler-approximated discrete-time model, but in this case, it is a simple pmf normally distributed around 0 with standard deviation $\sigma = \{0.05, 0.05, 0.2, 0.2\}$ for the four states $x = \left[p_x, \theta, v_x, \dot{\theta}\right]$.

The safety set is described by

$$h(x^t) = -2a_{\max} p_x^t - {v_x^t}^2 \operatorname{sgn}(v_x^t), \tag{4.44}$$

where $a_{\max} > 0$ is a tuneable parameter that designates the maximum linear acceleration at any point. This function is positive when $p_x < 0$, but allows $h(x^t) > 0$ when $p_x > 0$ if $v_x$ is sufficiently negative.

While this safety set is nonlinear in the control inputs, the one-step nature of this optimization problem results in no issues solving such a program in real-time, using modern solvers such as IPOPT or NLOPT. In future work, we plan to show how nonlinear CBFs can be linearized to result in an affine RCBF constraint, with the error included in the stochastic uncertainty to result in formal safety guarantees.

The RCBF was solved using PAGMO's integrated SLSQP solver from NLOPT. Each solution took roughly 0.7 ms to compute on a modern laptop, resulting in a maximum control frequency of 1428 Hz. Three trajectories are shown in Figure 4.6. The desired trajectory shows the trajectory with only the nominal controller, which clearly surpasses the safe set at $x = 0$. The trajectory corresponding to $\mathbb{E}[h]$ was filtered subject to the total conditional expectation coherent risk measure, which also corresponds to CVaR with $\beta = 1$. While this filter guarantees safety in the expectation, safety is frequently violated due to the stochastic uncertainty. Finally, the trajectory corresponding to CVaR with $\beta = 0.01$ results in safety over the entire trajectory.

Similarly, Figure 4.6 also demonstrates the same three trajectories with the finite-time reachability RCBF. Specifically, we utilize constants $\gamma = 0.05$ and $\epsilon = 0.1$, with an initial safety violation of $h(x^0) = -0.2$. From (4.32), this suggests a $t^* \leq 0.3667$s. While this is not reflected in the plot, which only shows $p_x^t$ rather that $h(x^t)$, we find that $h(x^{t^*}) > 0$ at $t^* = 0.08$s, well below the theoretical guarantee.

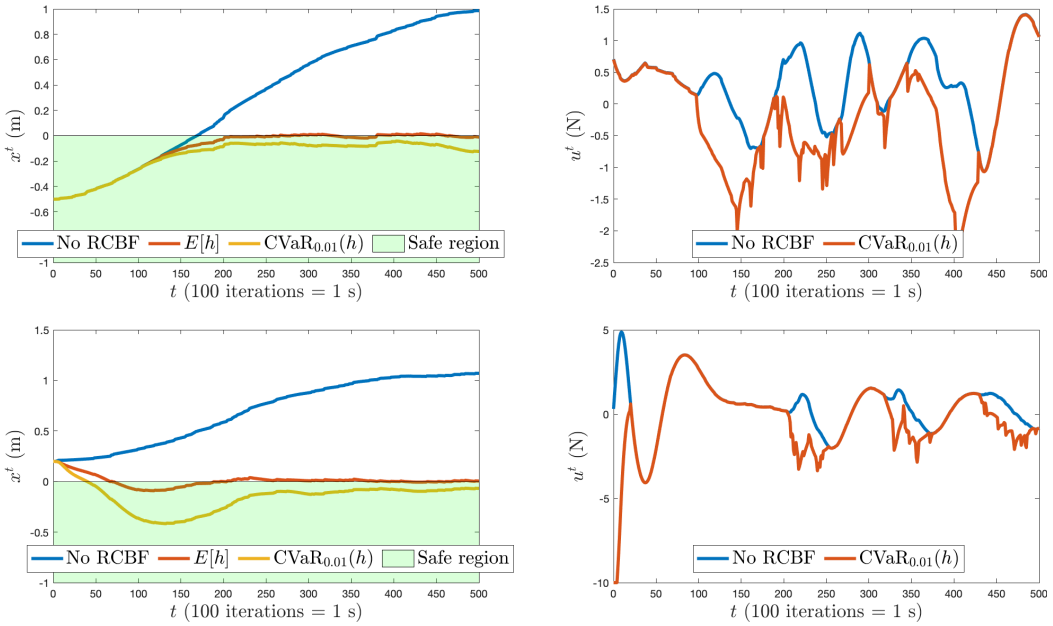



Figure 4.6: Simulation results for the cart-pole system with no RCBF filter, and with standard RCBF (top) and finite-time RCBF (bottom) filters using total conditional expectation and CVaR.

*Chapter 5*

# TOTAL SYSTEM SAFETY: MULTI-LAYER APPROACH AND FUTURE DIRECTIONS

In the following section, we take a brief break from safety regulation of a desired reference signal, and instead formulate the desired controller with a Model Predictive Control (MPC) framework. While this serves as a departure from the main focus of this thesis, the concepts illustrated and implemented are valuable to the study of safety regulation.

## 5.1 Unified Multi-Rate Control: from Low-Level Actuation to High-Level Planning

Control design for complex cyber-physical systems, which are described by continuous and discrete variables, is usually divided into different layers [161, 162, 146, 13, 63, 62, 87, 134, 65]. Each layer is designed using model of increasing accuracy and complexity, which allow the controller to take high-level decision, e.g. perform an overtaking maneuver, and to compute low-level commands, e.g. the input current to a motor. High-level decisions and low-level control actions are computed at different frequencies and the interaction between layers should be taken into account to guarantee safety of the closed-loop system [161].

In this section, we present a multi-rate hierarchical control scheme for nonlinear systems operating in partially observable environments. Our architecture, which is composed by three layers running at different frequencies, guarantees constraint satisfaction and maximization of the closed-loop probability of satisfying the high-level specifications. At the lowest level, we leverage the continuous time nonlinear system model and guarantee a bounded tracking error. The mid-level planning layer computes a reference trajectory using a simplified prediction model and the low-level tracking error bounds. Finally, at the highest level of abstraction we model the system-environment interaction using Mixed Observable Markov Decision Processes (MOMDPs), which allows us to account for partial environment observations. This control architecture is illustrated in Figure 5.1, and the environment used is shown in Figure 5.2.

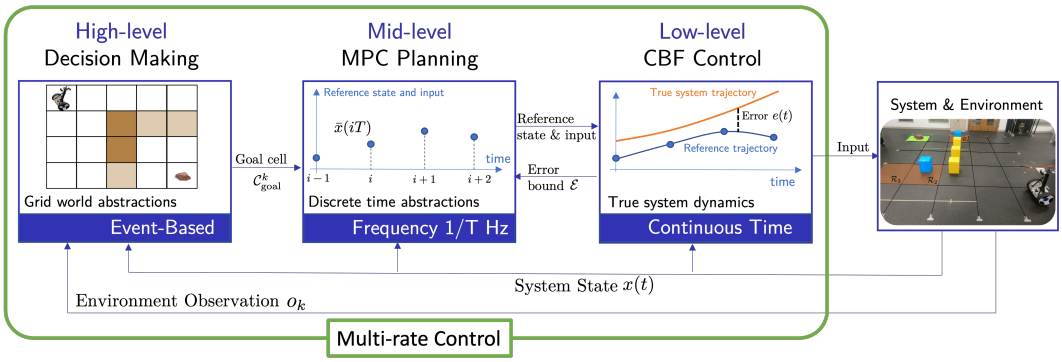



Figure 5.1: Multi-rate control architecture. The high-level decision maker leverages the system's state $x(t)$ and partial environment observations $o_k$ to compute a goal cell, the constraint set and the goal positions, which are fed to the mid-level MPC planner. The planner computes a reference trajectory given the tracking error bounds $\mathcal{E}$ from the low-level tracking controller. Finally at the lowest level, the control action is computed summing up the mid-level input $u_m(t)$ and the low-level input $u_l(t)$.

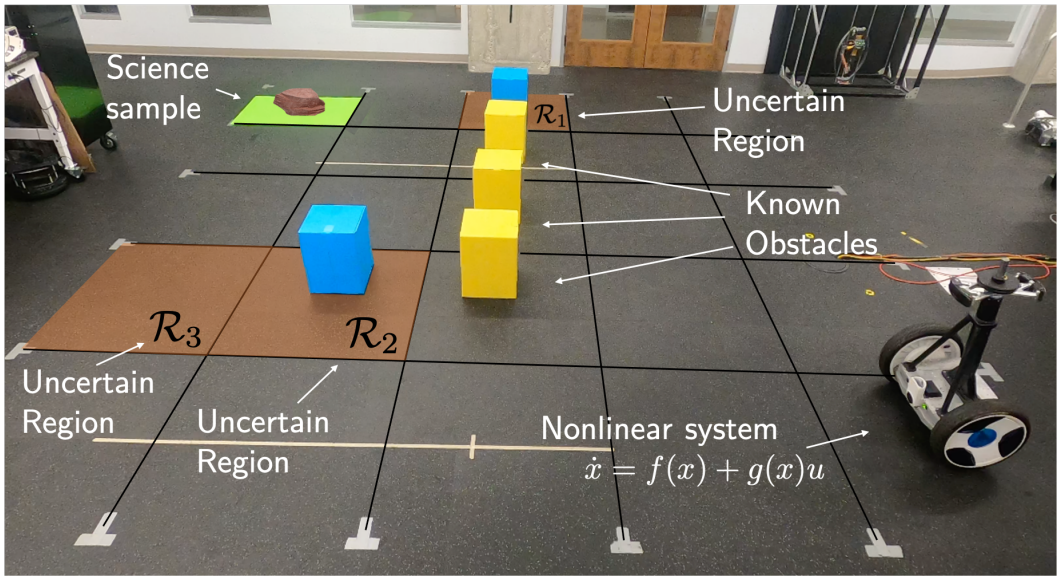



Figure 5.2: This figure shows an environment composed of 25 cells, 3 obstacles (yellow and blue boxes) and 3 uncertain regions (light brown). In this example the goal of the controller is to explore the state space in order to find a science sample.

### 5.1.1 Problem formulation

In this work, we consider the same dynamics (1.1), but we denote the state $x(t) = [p^\top(t), q^\top(t)]^\top \in \mathbb{R}^{n_x}$ for the position vector $p(t) \in \mathbb{R}^{n_p}$ and the vector $q(t) \in \mathbb{R}^{n_q}$ collecting the remaining states. Furthermore, the above system is subject to the following state and input constraints:

$$u(t) \in \mathcal{U}, p(t_i) \in \mathcal{X}_p \text{ and } q(t_i) \in \mathcal{X}_q \tag{5.1}$$

for all $t \in \mathbb{R}_{0+}$ and $t_i = iT$ for all $i \in \mathbb{Z}_{0+}$. The time constant $T$ is specified by the user and, as it will be clear later on, it defines the frequency at which the controller updates the planned trajectory. In the above equation (5.1), $\mathcal{X}_p$ represents free space and $\mathcal{X}_q$ is a user-defined constraint set.

High-level objectives are expressed using syntactically co-safe Linear Temporal Logic (scLTL) specifications. For a set of atomic proposition $\mathcal{AP}$, an scLTL specification is defined as follows:

$$\psi := p \mid \neg p \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \psi_1 U \psi_2 \mid \bigcirc \psi,$$

where the atomic proposition $p \in \mathcal{AP}$ and $\psi, \psi_1, \psi_2$ are scLTL formulas, which can be defined using the logic operators negation ($\neg$), conjunction ($\wedge$) and disjunction ($\vee$). Furthermore, scLTL formulas can be specified using the temporal operators until ($U$) and next ($\bigcirc$). Each atomic proposition $p$ is associated with a subset of the high-level state space $\mathcal{P} \subset \mathcal{S} \times \mathcal{Z}$ and a high-level state $\omega_k$ satisfies the proposition $p$ if $\omega_k \in \mathcal{P}$. Finally, satisfaction of a specification $\psi$ for the trajectory $\boldsymbol{\omega}_k = [\omega_k, \omega_{k+1}, \ldots]$, denoted by

$$\boldsymbol{\omega}_k \models \psi \tag{5.2}$$

is recursively defined as follows: *i)* $\boldsymbol{\omega}_k \models p \iff \omega_k \in \mathcal{P}$, *ii)* $\boldsymbol{\omega}_k \models \psi_1 \wedge \psi_2 \iff (\boldsymbol{\omega}_k \models \psi_1) \wedge (\boldsymbol{\omega}_k \models \psi_1)$, *iii)* $\boldsymbol{\omega}_k \models \psi_1 \vee \psi_2 \iff (\boldsymbol{\omega}_k \models \psi_1) \vee (\boldsymbol{\omega}_k \models \psi_1)$, *iv)* $\boldsymbol{\omega}_k \models \psi_1 U \psi_2 \iff \boldsymbol{\omega}_l \models \psi_2$ and $\boldsymbol{\omega}_j \models \psi_2$, $\forall j \in \{k, \ldots, l-1\}$, *v)* $\boldsymbol{\omega}_k \models \bigcirc\psi \iff \boldsymbol{\omega}_{k+1} \models \psi$. Please refer to [25, Chapter 3] for further details.

We consider nonlinear dynamical systems operating in partially observable environments, which are partitioned into $C_1, \ldots, C_c$ cells as in the example from Figure 5.2. We assume that the state of the system is perfectly observable, but we are given only partial observations about the environment state. Thus, at the highest level of abstraction, we model the interaction between the nonlinear system (1.1) and the environment using a Mixed Observable Markov Decision Process (MOMDP).

A MOMDP provides a sequential decision-making formalism for high-level planning under mixed full and partial observations [113] and it is defined as tuple $(\mathcal{S}, \mathcal{Z}, \mathcal{A}, \mathcal{O}, T_s, T_z, O)$, where

- $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ is a set of fully observable states;
- $\mathcal{Z} = \{1, \ldots, |\mathcal{Z}|\}$ is a set of partially observable states;
- $\mathcal{A} = \{1, \ldots, |\mathcal{A}|\}$ is a set of actions;
- $\mathcal{O} = \{1, \ldots, |\mathcal{O}|\}$ is the set of observations for the partially observable state $z \in \mathcal{Z}$;
- The indicator function[1] $T_s : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \times \mathcal{S} \to \{0, 1\}$ equals one if the system will transition to a state $s'$ given the action $a$ and current state $(s, z)$, i.e.,

$$T_s(s, z, a, s') = \begin{cases} 1 & \text{If } s' = f_s(s, z, a) \\ 0 & \text{Else} \end{cases},$$

  where the high-level update function $f_s : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \to \mathcal{S}$.

- The function $T_z : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \times \mathcal{S} \times \mathcal{Z} \to [0, 1]$ describes the probability of transitioning to a state $z'$ given the action $a$, the successor observable state $s'$ and the system's current state $(s, z)$, i.e.,

$$\begin{aligned} &T_z(s, z, a, s', z') \\ &\quad := P(z_{k+1} = z' | s_k = s, z_k = z, a_k = a, s_{k+1} = s'); \end{aligned}$$

- The function $O : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$ describes the probability of observing the measurement $o \in \mathcal{O}$, given the current state of the system $(s', z') \in \mathcal{S} \times \mathcal{Z}$ and the action $a$ applied at the previous time step, i.e.,

$$O(s', z', a, o) := P(o_k = o | s_k = s', z_k = z', a_{k-1} = a);$$

MOMDPs were introduced in [113] to model systems where a subspace of the state space is perfectly observable. In this work, the high-level observable state $s$ represents the location of the system, i.e., the grid cell containing the position vector $p(t)$ which is part of state $x(t) = [p^\top(t), q^\top(t)]^\top$ of the nonlinear system (1.1).

[1] We introduced the indicator function as it will be used later on to compute the belief vector update.

On the other hand, the definition of the partially observable state $z$ depends on the application, and it describes how the environment may affect the evolution of the system. For example, it may be used to model external events (e.g., rain, wind, etc) that would affect the traversability of specific regions of the state space. The evolution of the environment state $z$ may be stochastic and, most importantly, it is not perfectly observable. Thus, the controller has to make decisions based on the belief about the environment. For example, when the objective is to reach a goal location before a deadline and only partial knowledge about the traversability of the terrain is given, the controller should follow a path that maximizes the probability of reaching the goal in time given our belief about the environment.

More formally, control actions are computed based on the environment belief vector $b_k \in \mathcal{B} = \{b \in \mathbb{R}^{|\mathcal{Z}|} : \sum_{z=1}^{|\mathcal{Z}|} b^{(z)} = 1\}$ representing the posterior probability that the partially observable state environment $z_k$ equals $z \in \mathcal{Z}$, i.e., $b_k = [b_k^{(1)}, \ldots, b_k^{(|\mathcal{Z}|)}]$ with

$$b_k^{(z)} = \mathbb{P}(z_k = z | \boldsymbol{o}_k, \boldsymbol{s}_k, \boldsymbol{a}_{k-1}),\ \forall z \in \{1 \ldots, |\mathcal{Z}|\},$$

where at time $k$ the observation vector $\boldsymbol{o}_k = [o_0, \ldots, o_k]$, the observable state vector $\boldsymbol{s}_k = [s_0, \ldots, s_k]$, and the action vector $\boldsymbol{a}_{k-1} = [a_0, \ldots, a_{k-1}]$. Notice that the evolution of the environment belief vector $b_k$ is stochastic as it is a function of the noisy observation vector $\boldsymbol{o}_k = [o_0, \ldots, o_k]$. Therefore, the planned path that maximizes the probability of completing the task should be computed online at after collecting the observation $o_k$ and updating the belief vector $b_k$.

Given the system's state $x(t) \in \mathbb{R}^n$ and $k$ observations $\boldsymbol{o}_{k-1} = [o_0, \ldots, o_{k-1}] \in O^k$ about the environment, our goal is to design a control policy

$$\pi : \mathbb{R}^n \times O^k \to \mathcal{U}, \tag{5.3}$$

which maps the state $x(t)$ and the observation vector $\boldsymbol{o}_{k-1}$ to the continuous control action $u \in \mathcal{U}$. Furthermore, the control policy (5.3) should guarantee that state and input constraints (5.1) are satisfied and that the probability of satisfying the specification (5.2) is maximized.

### 5.1.2 Multi-rate control architecture

In this subsection, we describe the multi-rate control architecture. First, we design a low-level CLF-CBF controller, which tracks a reference state-input trajectory and guarantees bounded tracking errors. Afterwards, we show how to update the state-input reference trajectory leveraging an MPC, which is designed using a goal state

computed from a discrete high-level decision maker. Finally, we introduce the hierarchical multi-rate architecture, which guarantees that the synthesis objectives are satisfied.

**Low-Level Control**

We leverage CBFs and CLFs to design a low-level tracking controller for the nonlinear system (1.1). CBFs guarantee safety for nonlinear system, but they are suboptimal as the control action is computed without forecasting the system's trajectory. For this reason, we use CBFs to enforce safety around a reference state-input trajectory that is computed at low frequency by the mid-level planner, as shown in Figure 5.1.

**Error Model:** At the lowest layer, the goal of the controller is to track a reference trajectory $\bar{x}(t)$. We assume that the reference trajectory is given by the following Linear Time-Varying (LTV) model:

$$\Sigma_{\bar{x}} : \begin{cases} \dot{\bar{x}}(t) = A_{\lfloor t/T \rfloor}\bar{x}(t) + B_{\lfloor t/T \rfloor}u_m(t), & t \in \mathcal{T} \\ \bar{x}^+(t) = \Delta_{\bar{x}}(x(t)), & t \in \mathcal{T}^c \end{cases}, \tag{5.4}$$

where $\mathcal{T}^c = \cup_{j=0}^{\infty}\{jT\}$, $\mathcal{T} = \cup_{j=0}^{\infty}(jT, (j+1)T)$ and the time $T$ from (5.1) is specified by the user. Furthermore, we denote $\bar{x}^-(t) = \lim_{\tau \to t} \bar{x}(\tau)$ and $x^+(t) = \lim_{\tau \to t} \bar{x}(\tau)$ as the right and left limits of the reference trajectory $\bar{x}(t) \in \mathbb{R}^n$, which is assumed right continuous. In the above system, the reference input $u_m(t) \in \mathbb{R}^d$ and the *reset map* $\Delta_{\bar{x}} : \mathbb{R}^n \to \mathbb{R}^n$ maps the current state of the system $x(t)$ to the state $\bar{x}(t)$ of the reference trajectory. Both the reference input and the reset map are given by the middle layer. Finally, the time-varying matrices $(A_{\lfloor t/T \rfloor}, B_{\lfloor t/T \rfloor})$ are known and, in practice, may be computed linearizing the system dynamics (1.1).

Given the nonlinear system (1.1) and the LTV model (5.4), we define the error state $e(t) = x(t) - \bar{x}(t)$ and the associated error dynamics:

$$\Sigma_e : \begin{cases} \dot{e}(t) = f_e(x(t), \bar{x}(t), u_l(t) + u_m(t), t), & t \in \mathcal{T} \\ e^+(t) = x^+(t) - \bar{x}^+(t), & t \in \mathcal{T}^c \end{cases} \tag{5.5}$$

where the time-varying error dynamics are

$$\begin{aligned} &f_e(x,\bar{x}, u_l + u_m, t) \\ &\quad = f(x) + g(x)(u_l + u_m) - (A_{\lfloor t/T \rfloor}\bar{x} + B_{\lfloor t/T \rfloor}u_m). \end{aligned}$$

In the above definition, we dropped the dependence on time for states and inputs to simplify the notation. Furthermore, we introduce the low-level input constraint set

$\mathcal{U}_l \subset \mathcal{U}$ and the mid-level input constraint set $\mathcal{U}_m \subset \mathcal{U}$ which partition the input space, i.e.,

$$\mathcal{U}_l \oplus \mathcal{U}_m = \mathcal{U}.$$

Next, we design a low-level controller which guarantees that the reference trajectory $\bar{x}(t)$ from the LTV model (5.4) is tracked within some error bounds.

**Control Barrier and Lyapunov Functions:** We show how to design a tracking controller using CBFs and CLFs. First, we define the candidate Lyapunov function

$$V(e) = ||e||_Q, \tag{5.6}$$

where $||e||_Q = e^\top Q e$. Furthermore, we introduce the following safe set for the error dynamics (5.5):

$$\mathcal{E} = \{e \in \mathbb{R}^n : h_e(e) \geq 0\} \subset \mathbb{R}^n. \tag{5.7}$$

The above function $h_e$ is defined by the user and it depends on the application as discussed in the result section.

Finally, the CBF associated with the safe set (5.7), and the CLF from (5.6) are used to define the following CLF-CBF Quadratic Program (QP):

$$\begin{aligned} v_l^*(t) = \underset{v_l \in \mathcal{U}_l, \gamma}{\operatorname{argmin}} \; & ||v_l||_2 + c_1\gamma^2 \\ \text{s.t.} \quad & \frac{\partial V(e)}{\partial e} f_e(x, \bar{x}, v_l + u_m) \leq -c_2 V(e) + \gamma \\ & \frac{\partial h_e(e)}{\partial e} f_e(x, \bar{x}, v_l + u_m) \geq -\alpha_2(h_e(e)), \end{aligned} \tag{5.8}$$

where we dropped the time dependence to simplify the notation, and $v_l \in \mathcal{U}_l$ is the low-level control action. In the above QP, the parameters $c_1 \in \mathbb{R}_{0+}$, $c_2 \in \mathbb{R}_{0+}$, $\alpha_1 \in \mathcal{K}^e$ and $\alpha_2 \in \mathcal{K}^e$. Given the optimal control action $v_l^*(t)$ from the QP (5.8), the low-level control policy is defined as follows:

$$u_l(t) = \pi_l\big(x(t), \bar{x}(t), u_m(t)\big) = v_l^*(t). \tag{5.9}$$

**Assumption 5.** *The CLF-CBF QP* (5.8) *is feasible for all* $e = x - \bar{x} \in \mathcal{E}$ *and for all* $u_m \in \mathcal{U}_m$.

**Remark 9.** *We underline that Assumption 5 is satisfied for some* $\alpha_1 \in \mathcal{K}^e$ *and* $\alpha_2 \in \mathcal{K}^e$ *when the set* $\mathcal{E}$ *is robust control invariant for system* (5.5) *with* $u_m(t) \in \mathcal{U}_m$ *and mild assumptions on the Lie derivative of* (5.5) *hold. The set* $\mathcal{E}$ *may be hard to compute and standard techniques are based on HJB reachability analysis [65], SOS programming [139], Lyapunov-based methods [138] and Lipschitz properties of the system dynamics [37, 167].*

The low-level control policy (5.9) guarantees that the difference between the evolution of the nonlinear system (1.1) and the LTV model (5.4) is bounded. Indeed, when Assumption 5 is satisfied, the CLF-CBF QP (5.8) guarantees invariance of the safe set (5.7) for all $t \in (iT, (i+1)T)$ and $i \in \mathbb{Z}_{0+}$. Next, we show how to design a mid-level planner which leverages the safe set $\mathcal{E}$ from (5.7).

**Mid-Level Planning**

In this section we describe the mid-level planning strategy. At this level of abstraction, we assume that we are given a goal grid cell where we would like to steer the system. Afterwards, we compute a reference state-input trajectory using an MPC, which leverages a simplified model and the tracking error bounds from the previous section.

**Grid Model:** Given the state $x(t) = [p^\top(t), q^\top(t)]^\top$, we define the current grid cell $C^k_{\text{curr}}$, which contains the nonlinear system (1.1) for time $t \in [t^k, t^{k+1})$, i.e.,

$$p(t) \in C^k_{\text{curr}} \subset \mathcal{X}_p, \ \forall t \in [t^k, t^{k+1}). \tag{5.10}$$

Similarly, we define the goal cell $C^k_{\text{goal}}$, which represents the region where we want to steer the system for time $t \in [t^k, t^{k+1})$. Finally, we introduce the goal equilibrium sets $\mathcal{X}^k_{\text{curr}}$ and $\mathcal{X}^k_{\text{goal}}$, which collect the unforced equilibrium states that are contained into $C^k_{\text{curr}}$ and $C^k_{\text{goal}}$, i.e., for $i \in \{\text{curr}, \text{goal}\}$

$$\mathcal{X}^k_i = \{x = [p, q] \in \mathbb{R}^n | p \in C^k_i, \dot{x} = f(x) = 0\} \subset \mathbb{R}^n. \tag{5.11}$$

Throughout this section, we assume that $t^k$, $\mathcal{X}^k_{\text{goal}}$, $C^k_{\text{curr}}$ and $C^k_{\text{goal}}$ are given by the high-level planner and we synthesize a controller to drive the system from the current cell $C^k_{\text{curr}}$ to the goal cell $C^k_{\text{goal}}$.

**Model Predictive Control:** We design an MPC to compute the mid-level input $u_m(t)$ that defines the evolution of the reference trajectory (5.4) and to define the reset map $\Delta_{\bar{x}}$ for the LTV model (5.4). The MPC problem is solved at 1/T Hz and therefore the reference mid-level input is piecewise constant, i.e.,

$$\dot{u}_m(t) = 0 \ \forall t \in \mathcal{T} = \cup_{k=0}^{\infty}(kT, (k+1)T).$$

Next, we introduce the following discrete time linear model:

$$\bar{x}^d\big((i+1)T\big) = \bar{A}_i \bar{x}^d\big(iT\big) + \bar{B}_i u\big(iT\big), \tag{5.12}$$

where for all $i \in \mathbb{Z}_{0+}$

$$\bar{A}_i = e^{A_{\lfloor iT/T \rfloor}T} \text{ and } \bar{B}_i = \int_0^T e^{A_{\lfloor iT/T \rfloor}(T-\eta)} B_{\lfloor iT/T \rfloor} d\eta,$$

and the matrices $A_{\lfloor iT/T \rfloor}$ and $B_{\lfloor iT/T \rfloor}$ are defined in (5.4). Now notice that, as the mid-level input $u_m$ is piecewise constant, if at time $t_i = iT$ the state $\bar{x}(iT) = \bar{x}^+(iT) = \bar{x}^d(iT)$, then at time $t_{i+1} = (i+1)T$ we have that

$$\bar{x}^+((i+1)T) = \bar{x}^d((i+1)T). \tag{5.13}$$

Given the discrete time model (5.12), at time $t_i = iT \in \mathcal{T}^c$ we solve the following finite time optimal control problem:

$$\begin{aligned}
&J(x(iT), N) = \\
&\min_{\boldsymbol{v}_t, x^d_{i|i}} \quad ||x^d_{i|i} - x(iT)||_{Q_e} + \sum_{t=i}^{i+N-1} h\big(x^d_{t|i}, v_{t|i}\big) \\
&\qquad\qquad\qquad\qquad + ||p^d_{i+N|i} - p^k_{\text{goal}}||_{Q_f} \\
&\text{s.t.} \quad x^d_{t+1|i} = \bar{A}_t x^d_{t|i} + \bar{B}_t v^d_{t|i} \\
&\qquad x^d_{t|i} = \begin{bmatrix} p^d_{t+1|i} \\ q^d_{t+1|i} \end{bmatrix} \in \mathcal{X}^k_{p,q} \ominus \mathcal{E},\ v^d_{t|i} \in \mathcal{U}_m \\
&\qquad x^d_{i|i} - x(iT) \in \mathcal{E} \\
&\qquad x^d_{i+N|i} \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p, \forall t = \{i, \dots, i+N-1\},
\end{aligned} \tag{5.14}$$

where $\mathcal{E}$ is defined in (5.7), $||p||_Q = p^\top Q p$,

$$\mathcal{X}^k_{p,q} = \left\{ x = \begin{bmatrix} p \\ q \end{bmatrix} \in \mathbb{R}^n | p \in C^k_{\text{curr}} \cup C^k_{\text{goal}} \text{ and } q \in \mathcal{X}_q \right\} \tag{5.15}$$

and

$$\mathcal{E}_p = \left\{ e = \begin{bmatrix} e_p \\ 0 \end{bmatrix} \in \mathbb{R}^n | \exists e_q \in \mathbb{R}^{n_p} \text{ and } \begin{bmatrix} e_p \\ e_q \end{bmatrix} \in \mathcal{E} \right\}. \tag{5.16}$$

Notice that the MPC problem (5.14) is designed based on the time-varying components $\mathcal{X}^k_{\text{goal}}, C^k_{\text{curr}}, C^k_{\text{goal}}, p^k_{\text{goal}}$ which are given by the high-level decision maker, as shown in Figure 5.1. Problem (5.14) computes a sequence of open loop actions $\boldsymbol{v}^d_t = [v^d_{t|t}, \dots, v^d_{t+N|t}]$ and an initial condition $x^d_{i|i}$ such that the predicted trajectory steers the system to the terminal set $\mathcal{X}^k_{\text{goal}}$, while minimizing the cost and satisfying state and input constraints. Let $\boldsymbol{v}^{d,*}_t = [v^{d,*}_{t|t}, \dots, v^{d,*}_{t+N|t}]$ be the optimal solution and $[x^{d,*}_{t|t}, \dots, x^{d,*}_{t+N|t}]$ the associated optimal trajectory, then the mid-level policy is

$$\Pi_m : \begin{cases} u_m(t) = \pi_m\big(x(t), N\big) = v^{d,*}_{t|t} & t \in \mathcal{T}^c \\ \dot{u}_m(t) = 0 & t \in \mathcal{T} \end{cases}. \tag{5.17}$$

Finally, we define the reset map from the LTV model (5.4) as follows:

$$\Delta_{\bar{x}}(x(t)) = x^{d,*}_{t|t}. \tag{5.18}$$

**Assumption 6.** *Consider the equilibrium set $\mathcal{X}^k_{curr}$ defined in Equation* (5.10)*. For all states $x(t) \in \mathcal{X}^k_{curr} \oplus \mathcal{E}$ Problem* (5.14) *is feasible with horizon $N$.*

The above assumption is satisfied when any equilibrium state $\bar{x} \in \mathcal{X}^k_{\text{curr}}$ of the discrete time system (5.12) can be steered to the goal equilibrium set $\mathcal{X}^k_{\text{goal}}$ in at most $N$ time steps. More formally, Assumption 6 holds when, for the discrete time system (5.12), $\mathcal{X}^k_{\text{goal}}$ is $N$-step backward reachable from the set $\mathcal{X}^k_{\text{curr}}$.

Later, we will show that when the nonlinear system (1.1) and the LTV system (5.4) are in closed-loop with the low-level policy (5.9) and the mid-level policy (5.17), then state and input constraints (5.1) are satisfied for system (1.1). Furthermore, the nonlinear system (1.1) is steered from the current cell $C^k_{\text{curr}}$ to the goal cell $C^k_{\text{goal}}$ in finite time.

**Remark 10.** *We highlight that also RRT-based methods can be combined with CLF-CBF to design a multi-rate control architecture. In particular, it would be possible to leverage sampling-based methods, such as [76, 19, 56, 88], to repeatedly solve online problem* (5.14)*. Notice that it important to consider the constraint tightening from problem* (5.14) *that accounts for the low-level tracking error. Indeed, this constraint tightening strategy allow us to guarantee safety of the nonlinear system* (1.1) *in closed-loop with the proposed multi-rate control architecture, as we will discuss later on.*

### High Level Decision Making

Here, we first describe how to compute a control policy which maximizes the probability of satisfying the specifications. Afterwards, we show how to compute the time-varying components $p^k_{\text{goal}}$, $C^k_{\text{curr}}$, $C^k_{\text{goal}}$ and $\mathcal{X}^k_{\text{goal}}$ used in the MPC problem (5.14).

**Belief Model:** For the MOMDP, we introduced the belief vector $b_k \in \mathcal{B}$ that represents the posterior probability that the partially observable state $z_k$ equals $z \in \mathcal{Z}$.

The belief is a sufficient statistic and, for all $z' \in \mathcal{Z}$, it evolves accordingly to the following update equation:

$$b_{k+1}^{(z')} = \eta O(s_{k+1}, z', a_k, o_k) \times \sum_{z \in \mathcal{Z}} T_s(s_k, z, a_k, s_{k+1}) T_z(s_k, z, a_k, s_{k+1}, z_{k+1}) b_k^{(z)},$$

where $\eta$ is a normalization constant [113]. Notice that the above update equation can be written in a compact form, i.e.,

$$b_{k+1} = f_b(s_{k+1}, s_k, o_k, a_k, b_k), \tag{5.19}$$

where $f_b : \mathcal{S} \times \mathcal{S} \times \mathcal{O} \times \mathcal{A} \times \mathcal{B} \to \mathcal{B}$. Finally, given the belief $b_k$, we introduce the maximum likelihood environment state estimate:

$$\hat{z}_k = \underset{z \in \mathcal{Z}}{\operatorname{argmax}}\, \mathbb{P}(z_k = z | \boldsymbol{o}_k, \boldsymbol{s}_k, \boldsymbol{a}_{k-1}) = \underset{z \in \mathcal{Z}}{\operatorname{argmax}}\; b^{(z)}. \tag{5.20}$$

**Quantitative Control Policy:** At the highest level of abstraction our goal is to compute a control policy $\pi_h$, which maximizes the probability that the high-level trajectory $\boldsymbol{\omega}$ satisfies the specifications $\psi$. Such control control policy can be computed solving the following quantitative problem:

$$\pi_h = \underset{\pi}{\operatorname{argmax}}\, \mathbb{P}^{\pi}[\boldsymbol{\omega} \models \psi], \tag{5.21}$$

where $\mathbb{P}^{\pi}[\boldsymbol{\omega} \models \psi]$ represents the probability that the specification $\psi$ is satisfied for the closed-loop trajectory $\boldsymbol{\omega}$ under the policy $\pi$. The solution to the above qualitative problem can be approximated using point-based and simulation-based strategies [29, 63, 155, 62, 159]. In this work, we used the point-based strategy discussed in [126]. The resulting high-level control policy maps the high-level state $s_k$ and the environment belief $b_k$ to the high-level control action $a_k$, i.e.,

$$a_k = \pi_h(s_k, b_k). \tag{5.22}$$

The high-level policy (5.21) is leveraged in Algorithm 9 to compute the goal position $p_{\text{goal}}^k$ and the sets $C_{\text{curr}}^k$ and $C_{\text{goal}}^k$, which are used in the MPC problem (5.14). In Algorithm 9, we first use the function `getState`, which maps the current state $x(t)$ to the high-level state $s_k$ representing the cell containing the nonlinear system 1.1 (line 2). Then, we compute the current cell $C_{\text{curr}}^k$ associated with the high-level state $s_k$ using the function `getCell` (line 3). Afterwards, we update the belief state $b_k$

**Algorithm 9** Update High-Level

**inputs:** $x(t)$, $o_k$, $s_{k-1}$, $a_{k-1}$, $b_{k-1}$
set current high-level state $s_k = \texttt{getState}(x(t))$
compute current set $C^k_{\text{curr}} = \texttt{getCell}(s_k)$
update belief $b_k$ using (5.19)
compute high-level action $a_k = \pi_h(s_k, b_k)$
compute maximum likely estimate $\hat{z}_k$ using (5.20)
update state $s_{k+1} = f_s(s_k, \hat{z}_k, a_k)$
compute goal set $C^k_{\text{goal}} = \texttt{getCell}(s_{k+1})$
computed the forecasted action $\hat{a} = \pi_h(s_{k+1}, b_k)$
compute the forecasted state $\hat{s}_{k+2} = f_s(s_{k+1}, \hat{z}_k, \hat{a})$
set forecasted set $C^k_{\text{forc}} = \texttt{getCell}(s_{k+1})$
get forecasted cell center $c^{\text{forc}} = \texttt{getCenter}(C^k_{\text{forc}})$
compute goal position $p^k_{\text{goal}} = \text{Proj}(c^{\text{forc}}, C^k_{\text{goal}})$
**return:** $a_k$, $b_k$, $s_k$, $C^k_{\text{goal}}$, $C^k_{\text{curr}}$, $p^k_{\text{goal}}$

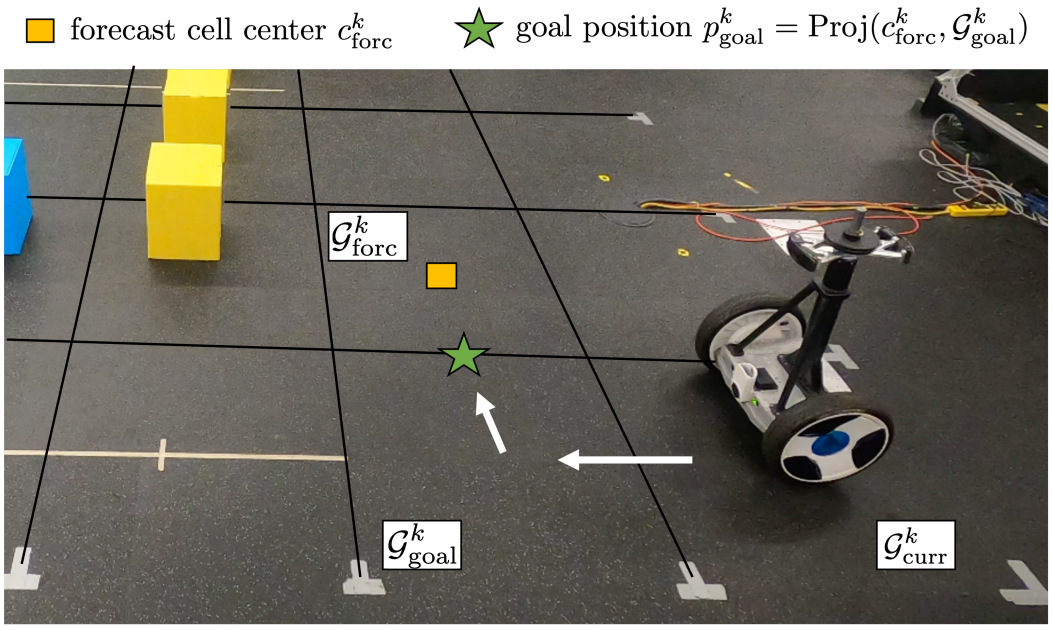



Figure 5.3: The above figure illustrated the high-level updated from Algorithm 9 that is used to compute the goal position (green star).

and we compute the control action $a_k$ (lines 4 – 5). Given the control action $a_k$ and the maximum likelihood estimator of the environment state $\hat{z}_k$, we update the high-level state and we compute the goal cell $C^k_{\text{goal}}$ (lines 6 – 8). Next, given the current belief $b_k$, we compute the action $\hat{a}$ that the high-level planner would select at the next update $k + 1$ assuming the belief $b_k$ is unchanged. We leverage this action to estimate the high-level state $\hat{s}_{k+2}$, which represents the location where we

should steer the system after transitioning to the high-level state $s_{k+1}$. The state $\hat{s}_{k+2}$ is used to incorporate forecast into the high-level planner. In particular, given the $\hat{s}_{k+2}$ we compute the forecasted cell center $c^{\text{forc}} \in \mathbb{R}^{n_p}$ and the forecasted cell $C^k_{\text{forc}}$ where the system should be steered to, if no informative observations are collected (lines $11-12$). Finally, the goal cell $C^k_{\text{goal}}$ and the forecasted center $c^{\text{forc}} \in \mathbb{R}^{n_p}$ are used to compute the goal position $p^{\text{goal}}$ (line 13).

Figure 5.3 illustrates the high-level update from Algorithm 9 that is used to compute the goal position leveraged in the design of the mid-level MPC. In this example, the Segway is located in the top left corner of the grid and the current high-level action $a_k$ is to move east. The figure also shows the forecasted action $\hat{a}$ that the Segway would take from the goal region, if the belief $b_k$ is not updated. Basically, $\hat{a}$ is a high-level open-loop prediction of the future control action and it is used to incorporate forecast into the high-level decision maker. Indeed, the goal position $p^k_{\text{goal}}$ is computed projecting the forecasted cell center $c^{\text{forc}}$ onto the goal cell $C^k_{\text{goal}}$.

**Control Architecture**

Finally, we introduce the multi-rate hierarchical control architecture which leverages the low-level, mid-level and high-level control policies from the previous sections. The multi-rate control Algorithm 10 details the architecture depicted in Figure 5.1. When the nonlinear system (1.1) reaches the goal cell (i.e., $p(t) \in C^k_{\text{goal}}$), the high-level decision maker reads the new observations $o_{k+1}$ and updates high-level state, action, goal position $p^k_{\text{goal}}$, goal cell $C^k_{\text{goal}}$ and current cell $C^k_{\text{curr}}$ (lines $3-4$). Finally, it updates the high-level time $k$ and it initializes the MPC horizon $N^k_i = N$. Afterwards, the mid-level planner (lines $8-20$) updates the mid-level time counter $i$ and the planned trajectory at a constant frequency of $1/T$ Hz. First, it solves the MPC problem (5.14) with $N = N^k_i$ and time-varying components $\mathcal{X}^k_{\text{goal}}, C^k_{\text{goal}}, C^k_{\text{curr}}$ and $p^k_{\text{goal}}$. If the MPC problem is not feasible, the planner computes a contingency plan (lines $10-14$), otherwise it updates the prediction horizon (lines 15-16). Note that the MPC problem solved in line 9 of Algorithm 9 may be not feasible as the terminal constraint set $\mathcal{X}^k_{\text{goal}}$ is updated by the high-level planner. For this reason, we introduced the contingency plan (lines $10-14$), where the MPC problem from line 11 is constructed using the terminal constraint set $\mathcal{X}^{k-1}_{\text{goal}}$. As we will show in the proof of Theorem 21, when the MPC problem constructed with terminal constraint set $\mathcal{X}^k_{\text{goal}}$ is not feasible, we can guarantee the feasibility of the contingency MPC with $\mathcal{X}^{k-1}_{\text{goal}}$ as terminal constraint. This fact allows us to guarantee safety for the closed-loop system. Finally, Algorithm 10 computes the low-level control action

**Algorithm 10** Multi-Rate Control

**inputs:** $k$, $s_k$, $b_k$, $a_k$, $i$, $x(t)$, $u_m(t)$, $\bar{x}(t)$, $C^k_{\text{curr}}$, $C^k_{\text{goal}}$, $p^k_{\text{goal}}$, $N^k_i$, $C^{k-1}_{\text{curr}}$, $C^{k-1}_{\text{curr}}$, $p^{k-1}_{\text{goal}}$, $N^{k-1}_i$

**if** $q(t) \in C^k_{\text{goal}}$ or $k = 0$ **then**

`//Update high-level goal`

measure $o_{k+1}$

update $a_{k+1}$,$b_{k+1}$, $s_{k+1}$, $\mathcal{X}^k_{\text{goal}}$, $C^{k+1}_{\text{goal}}$, $C^{k+1}_{\text{curr}}$, $p^{k+1}_{\text{goal}}$ using Algorithm 9 with $x(t)$, $o_{k+1}$, $s_k$, $a_k$, $b_k$

set $N^{k+1}_i = N$

$k = k + 1$

**end if**

**if** $t \in \mathcal{T}^c = \cup_{j=0}^{\infty}\{jT\}$ **then**

`//Update mid-level plan`

solve MPC problem (5.14) with $N = N^k_i$, and $\mathcal{X}^k_{\text{goal}}$, $C^k_{\text{curr}}$, $C^k_{\text{goal}}$, $p^k_{\text{goal}}$

**if** the MPC problem (5.14) is not feasible **then**

solve MPC problem (5.14) with $N = N^{k-1}_i$, and $\mathcal{X}^{k-1}_{\text{goal}}$, $C^{k-1}_{\text{curr}}$, $C^{k-1}_{\text{goal}}$, $p^{k-1}_{\text{goal}}$

set $N^{k-1}_{i+1} = \max(1, N^{k-1}_i - 1)$

set $N^k_{i+1} = N^k_i$

**else**

set $N^{k-1}_{i+1} = N^{k-1}_i$

set $N^k_{i+1} = \max(1, N^k_i - 1)$

**end if**

set $u_m(t) = v^{d,*}_{t|t} + K(x(t) - \bar{x}^{d,*}_{t|t})$

update $\bar{x}(t) = \Delta_{\bar{x}}(x(t)) = \bar{x}^{d,*}_{t|t}$

$i = i + 1$

**end if**

`//Compute low-level control`

solve the CBF problem (5.8)

Compute total input $u(t) = u_l(t) + u_m(t)$

**Return:** $u(t)$, $k$, $s_k$, $b_k$, $a_k$, $i$, $x(t)$, $u_m(t)$, $\bar{x}(t)$, $C^k_{\text{curr}}$, $C^k_{\text{goal}}$, $p^k_{\text{goal}}$, $N^k_i$, $N^{k-1}_i$

solving the CLF-CBF QP (5.8) and the total control input $u(t) = u_l(t) + u_m(t)$.

### 5.1.3 Safety and performance guarantees

In this subsection we show the properties of the proposed multi-rate control architecture. We consider the augmented system:

$$\Sigma_{\text{aug}} : \begin{cases} \dot{x}(t) = f\big(x(t)\big) + g\big(x(t)\big)\big(u_l(t) + u_m(t)\big), & t \geq 0 \\ \dot{\bar{x}}(t) = A_{\lfloor t/T \rfloor}\bar{x}(t) + B_{\lfloor t/T \rfloor}u_m(t), & t \in \mathcal{T} \\ \bar{x}^+(t) = \Delta_{\bar{x}}(x(t)), & t \in \mathcal{T}^c \end{cases} \tag{5.23}$$

where the nonlinear dynamics for state $x(t) \in \mathbb{R}^n$ are defined in (1.1) and the LTV model for the nominal state $\bar{x}(t) \in \mathbb{R}^n$ is defined in (5.4) for the reset map (5.18) given by the MPC. In what follows, we analyse the properties of the proposed multi-rate control Algorithm 10 in closed-loop with system (5.23). We show that the closed-loop system satisfies state and input constraints (5.1) and that the proposed algorithm maximizes the probability of satisfying the specifications. Notice that in practice the state $x(t)$ is given by the nonlinear system (1.1), whereas the nominal state $\bar{x}(t)$ is computed by the low-level layer to update the tracking error $e(t)$, as shown in Figure 5.1.

**Proposition 12.** *Consider the closed-loop system* (5.9) *and* (5.23) *with mid-level input* $u_m(t) \in \mathcal{U}_m$ *and* $\dot{u}_m(t) = 0, \forall t \in \mathcal{T}$. *If Assumption 5 holds and the error* $e(kT) = x(kT) - \bar{x}(kT) \in \mathcal{E}$ *for all* $k \in \mathbb{Z}_{0+}$, *then the control policy* (5.9) *guarantees that* $e(t) \in \mathcal{E}$ *and* $u_l(t) \in \mathcal{U}_l$, $\forall t \in [kT, (k+1)T)$.

*Proof.* The proof follows from standard CBF arguments. First, we notice that the error $e(kT) = x(kT) - \bar{x}(kT)$ follows the error dynamics in (5.5). Furthermore, by construction the time-varying matrices $(A_{\lfloor t/T \rfloor}, B_{\lfloor t/T \rfloor})$ are constant for $t \in [kT, (k+1)T)$. Therefore, for all $k \in \mathbb{Z}_{0+}$ and $t \in [kT, (k+1)T)$, we have that error dynamics in (5.5) are nonlinear control affine for the low-level input $u_l$. This fact implies that, if at time $t = kT$ the error $e(kT) = x(kT) - \bar{x}(kT) \in \mathcal{E}$, then from the feasibility of the CLF-CBF QP (5.8) from Assumption 5 we have that $e(t) = x(t) - \bar{x}(t) \in \mathcal{E}$, $\forall t \in [kT, (k+1)T)$. □

Proposition 12 shows that between time $t_i = iT$ and $t_{i+1} = (i+1)T$ the difference between the state $x$ and the state $\bar{x}$ of the reference trajectory is bounded. Next, we show that this property allows us to guarantee safety and convergence in finite time to the goal cell $C^k_{\text{goal}}$ for the nonlinear system (1.1). In turns, convergence in finite time allows us to show that the high-level specifications are satisfied, when the following assumption holds.

**Assumption 7.** *Algorithm 9 returns a goal cell* $C^k_{goal}$ *which is contained in the feasible set* $\mathcal{X}_p$.

**Remark 11.** *The above assumption is satisfied when a perfect measurement is available when the system is in a grid cell adjacent to an uncertain region. In this case, the high-level planner is able to identify the obstacle-free cells and it will not return a goal cell* $C^k_{goal}$ *where an obstacle is located.*

**Theorem 21.** *Let Assumptions 5-7 hold and consider system* (5.23) *in closed-loop with Algorithm 10. If at time $t_i = iT$ the MPC problem* (5.14) *is feasible with $N_i^k = N$ and time-varying components $\mathcal{X}_{goal}^k, C_{curr}^k, C_{goal}^k$ and $p_{goal}^k$, then there exists a $j \in \{i, \dots, i+N-1\}$ such that the closed-loop system satisfies state and input constraints* (5.1) *for all $t \in \{iT, \dots, jT\}$ and the state $x((j+1)T) = [p^\top((j+1)T), q^\top((j+1)T)]^\top$ reaches the goal cell $C_{goal}^k$, i.e., $p((j+1)T) \in C_{goal}^k$.*

*Proof.* From Assumption 7 we have that the high-level policy (5.21), takes a high-level action $a_k$ which avoids collision with the obstacles, i.e.,

$$C_{\text{goal}}^k \subset \mathcal{X}_p. \tag{5.24}$$

Next, we show that if at time $t_i = iT$ the MPC problem (5.14) is feasible with $\mathcal{X}_{\text{goal}}^k, C_{\text{goal}}^k, C_{\text{curr}}^k, p_{\text{goal}}^k$ and $N_i^k > 1$, then at time $t_{i+1} = (i+1)T$ the MPC problem (5.14) is feasible with $\mathcal{X}_{\text{goal}}^k, C_{\text{goal}}^k, C_{\text{curr}}^k, p_{\text{goal}}^k$ and $N_{i+1}^k = N_i^k - 1$. Let

$$[x_{i|i}^{d,*}, x_{i+1|i}^{d,*}, \dots, x_{i+N_i^k|i}^{d,*}] \text{ and } [u_{i|i}^{d,*}, \dots, u_{i+N_i^k-1|i}^{d,*}]$$

be the optimal state input sequence to the MPC problem (5.14) at time $t_i = iT$. Then, from Proposition 12, equation (5.13) and the definition of the reset map (5.18), we have that

$$x((i+1)T) - \bar{x}_{i+1|i}^{d,*} = x((i+1)T) - \bar{x}^-((i+1)T) \in \mathcal{E} \tag{5.25}$$

and therefore, by standard MPC arguments, the following sequences of $N_i^k - 1$ states and $N_i^k - 2$ inputs

$$[x_{i+1|i}^{d,*}, \dots, x_{i+N_i^k|i}^{d,*}] \text{ and } [u_{i+1|i}^{d,*}, \dots, u_{i+N_i^k-1|i}^{d,*}] \tag{5.26}$$

are feasible at time $t_{i+1} = (i+1)T$ for the MPC problem (5.14) with $\mathcal{X}_{\text{goal}}^k, C_{\text{goal}}^k, C_{\text{curr}}^k, p_{\text{goal}}^k$ and $N_{i+1}^k = N_i^k - 1$.

Now, we show that state and input constraints are satisfied until the system reaches the goal set $C_{\text{goal}}^k$. Recall that by assumption the MPC problem is feasible at time $t_i = iT$ with $\mathcal{X}_{\text{goal}}^k, C_{\text{goal}}^k, C_{\text{curr}}^k, p_{\text{goal}}^k, N_i = N$ and assume that $p(jT) \notin C_{\text{goal}}^k$ for all $j \in \{i, \dots, i+N-1\}$. By induction the MPC problem (5.14) with $\mathcal{X}_{\text{goal}}^k, C_{\text{goal}}^k, C_{\text{curr}}^k, p_{\text{goal}}^k$ and $N_j^k = N_i^k - j$ is feasible for all $j \in \{i, \dots, i+N-1\}$. Consequently, Algorithms 9 returns a feasible mid-level control action[2] $u_m(t) \in \mathcal{U}_m$.

[2]Note that as $p(jT) \notin C_{\text{goal}}^k$ for all $j \in \{i, \dots, i+N-1\}$ the MPC time-varying components are not updated.

Furthermore, from Proposition 12 we have the low-level controller returns a feasible control action $u_l(t) \in \mathcal{U}_l$ and therefore

$$u(t) = u_l(t) + u_m(t) \in \mathcal{U}_l \oplus \mathcal{U}_m = \mathcal{U}, \forall t \in \mathbb{R}_{0+}. \tag{5.27}$$

The feasibility of the state-input sequences in (5.26) for the MPC problem solved with $\mathcal{X}^k_{\text{goal}}, C^k_{\text{goal}}, C^k_{\text{curr}}, p^k_{\text{goal}}$ implies that

$$\begin{aligned} x^{d,*}_{j|j} &\in \mathcal{X}^k_{p,q} \ominus \mathcal{E} \\ x(jT) - x^{d,*}_{j|j} &\in \mathcal{E}, \end{aligned} \tag{5.28}$$

$\forall j \in \{i, \ldots, i+N-1\}$. Consequently, from the above equation and definition (5.15), we have that

$$p(jT) \in \mathcal{X}_p \text{ and } q(jT) \in \mathcal{X}_q, \forall j \in \{i, \ldots, i+N-1\}.$$

Finally, we show that the state $x(t)$ of the augmented system (5.23) in closed-loop with Algorithm 10 converges to the goal cell $C^k_{\text{goal}}$ in finite time. We have shown that, if $p(jT) \notin C^k_{\text{goal}}$ for all $j \in \{i, \ldots, i+N-1\}$, then the MPC problem is feasible for all time $t_k = kT$ and $k \in \{i, \ldots, i+N-1\}$. Now we notice that by feasibility of the MPC problem at time $t_{i+N-1} = (i+N-1)T$ with $N_{i+N-1} = 1$, we have that the optimal planned trajectory satisfies

$$x^{d,*}_{i+N|i+N-1} \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p.$$

From Proposition 12, equation (5.13) and the definition of the reset map (5.18), we have that

$$x((i+N)T) - \bar{x}^{d,*}_{i+N|i} = x((i+N)T) - \bar{x}((i+N)T) \in \mathcal{E}.$$

The above equation together with definition (5.16) imply that at time $t_{i+N} = (i+N)T$

$$x((i+N)T) = \begin{bmatrix} p((i+N)T) \\ q((i+N)T) \end{bmatrix} \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p \oplus \mathcal{E}$$

and therefore $p((i+N)T) \in C^k_{\text{goal}}$.

Concluding, if for all time $t_j = jT$ and $j \in \{i, \ldots, i+N-1\}$ we have that $p(jT) \notin C^k_{\text{goal}}$, then $p((i+N)T) \in C^k_{\text{goal}}$. Thus, the closed-loop system converges to the goal cell $C^k_{\text{goal}}$ in finite time. □

Finally, we leverage Theorem 21 to show that the multi-rate control Algorithm 10 steers the system in finite time to goal cell $C^k_{\text{goal}}$ for all $k \in \mathbb{Z}_{0+}$ and, consequently, the closed-loop systems satisfies the high-level specifications when Assumption 7 is satisfied. In particular, we show that the contingency plan from lines 10–14 of Algorithm 10 guarantees feasibility of the planner when the time-varying components are updated.

**Theorem 22.** *Let Assumptions 5-7 hold and consider system* (5.23) *in closed-loop with Algorithm 10. If* $x(0) \in \mathcal{X}^k_{curr} \oplus \mathcal{E}$*, then the closed-loop system* (10) *and* (5.23) *maximizes the probability that the closed-loop satisfies the high-level specifications.*

*Proof.* The proof follows by induction. Assume that at time $t_i = iT$ the closed-loop system reaches the goal cell $C^k_{\text{goal}}$, i.e., $p(iT) \in C^k_{\text{goal}}$. Then, at time $t_i = iT$ we have that the high-level decision maker from Algorithm 10 (lines 2–9) updates the high-level time and the time-varying components $\mathcal{X}^{k+1}_{\text{goal}}, C^{k+1}_{\text{curr}}, C^{k+1}_{\text{goal}}, p^{k+1}_{\text{goal}}$ used to design the MPC problem (5.14). After the high-level update, the MPC problem with $N = N^{k+1}_j$, $\mathcal{X}^{k+1}_{\text{goal}}, C^{k+1}_{\text{curr}}, C^{k+1}_{\text{goal}}$, and $p^{k+1}_{\text{goal}}$ may be either feasible or unfeasible[3]. Thus, we analyse the following three cases for $j \geq i$:

*Case 1:* The MPC problem with $C^{k+1}_{\text{goal}}, C^{k+1}_{\text{curr}}, p^{k+1}_{\text{goal}}$ and $N = N^{k+1}_j$ is feasible, therefore from Theorem 21 we have that Algorithm 10 steers the nonlinear system to the goal $C^{k+1}_{\text{goal}}$.

*Case 2:* The MPC problem with $C^{k+1}_{\text{goal}}, C^{k+1}_{\text{curr}}, p^{k+1}_{\text{goal}}$ and $N = N^{k+1}_j$ is not feasible and $N^k_j = 1$. Then from Theorem 21, we have that the contingency MPC with $N^k_j$, $C^k_{\text{goal}}, C^k_{\text{curr}}$ and $p^k_{\text{goal}}$ is feasible and Algorithm 9 returns a feasible control action. Furthermore, as $N^k_j = 1$ the terminal state of the optimal predicted trajectory is

$$x^{d,*}_{j+1|j} \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p.$$

The above equation together with equation (5.25) imply that

$$x((j+1)T) \in \mathcal{X}^k_{\text{goal}} \ominus \mathcal{E}_p \oplus \mathcal{E} \subset \mathcal{X}^k_{\text{goal}} \oplus \mathcal{E}.$$

Therefore, from Assumption 6, we have that at the next time step $t_{j+1} = (j+1)T$ the MPC problem with $N^{k+1}_{j+1} = N$, $\mathcal{X}^{k+1}_{\text{goal}}, C^{k+1}_{\text{goal}}, C^{k+1}_{\text{curr}}$ and $p^{k+1}_{\text{goal}}$ is feasible and, from Theorem 21, we have that Algorithm 10 steers the nonlinear system to the goal $C^{k+1}_{\text{goal}}$ in finite time.

[3]Unfeasiblity may be caused by the update of $C^{k+1}_{\text{goal}}$, $C^{k+1}_{\text{curr}}$ and $\mathcal{X}^{k+1}_{\text{curr}}$.

*Case 3:* The MPC problem with $C^{k+1}_{\text{goal}}, C^{k+1}_{\text{curr}}, p^{k+1}_{\text{goal}}$ and $N = N^{k+1}_j$ is not feasible and $N^k_j > 1$. Then from Theorem 21, we have that the contingency MPC with $N^k_j$, $C^k_{\text{goal}}, C^k_{\text{curr}}$ and $p^k_{\text{goal}}$ is feasible (lines 10–13 in Algorithm 10).

Concluding, we have that by assumption $x(0) \in \mathcal{X}^k_{\text{curr}} \oplus \mathcal{E}$, which from Assumption 6 implies that at time $t = 0$ the MPC is feasible and therefore by Theorem 21 Algorithm 10 steers system (1.1) to $\mathcal{G}^0_{\text{goal}}$. Afterwards, the conditions form one the above Cases 1–3 are met. Now notice that at each time step $N^k_{j+1} = N^k_j - 1$ (line 16), thus Case 3 occurs at most $N$ times. Therefore, after at most $N$ time steps the conditions from either Case 1 or Case 2 are met and Algorithm 10 will steer the system to $C^{k+1}_{\text{goal}}$. Consequently, as the high-level policy (5.21) maximizes the probability of satisfying the specifications by definition, we have that the closed-loop system maximizes the probability of satisfying the specifications. □

### 5.1.4 Applications: Multi-rate control on the Segway

We tested the proposed strategy in simulation and experiment on navigation tasks inspired by the Mars exploration mission [62, 63, 110]. We control a Segway-like robot and our goal is to explore the environment to find science samples which may be located with some probability in known goal regions $\mathcal{G}_i$ shown in Figure 5.4. The specification $\psi = \neg\texttt{collision} U((\texttt{Goal}_1 \wedge \texttt{sample}_1) \vee (\texttt{Goal}_2 \wedge \texttt{sample}_2))$, where the atomic proposition $\texttt{sample}_i$ is satisfied if the region $\mathcal{G}_i$ contains a science sample and the atomic proposition $\texttt{Goal}_i$ is satisfied if the Segway is in a goal cell $\mathcal{G}_i$. The high-level control policy associated with specification $\psi$ is computed solving a reach-avoid problem for the product MOMDP, which is computed preforming the cross-product between an automata associated the specification $\psi$ and the original MOMDP[4]. For further details on how to convert a specification into a finite state automata and the computation of the product, please refer to [25]. While performing the search task, we have to collect observations to determine the state of the uncertain region $\mathcal{R}_i$, which may be traversable with some probability. The controller has access to only partial observations about the environment. In particular, the Segway receives a perfect observation about the state of the uncertain region $\mathcal{R}_i$ when one cell away, an observation which is correct with probability 0.8, when the Manhattan distance is smaller than two, and an uninformative observations otherwise. Similarly,

[4]The computational complexity of solving the high-level synthesis problem is a function of the dimension of the product MOMDP, which may grow exponentially for complex specifications. The analysis of the computational tractability of the high-level synthesis process is beyond the scope of this work and the code used to synthesize the high-level policy can be found at `https://github.com/urosolia/MOMDP`.

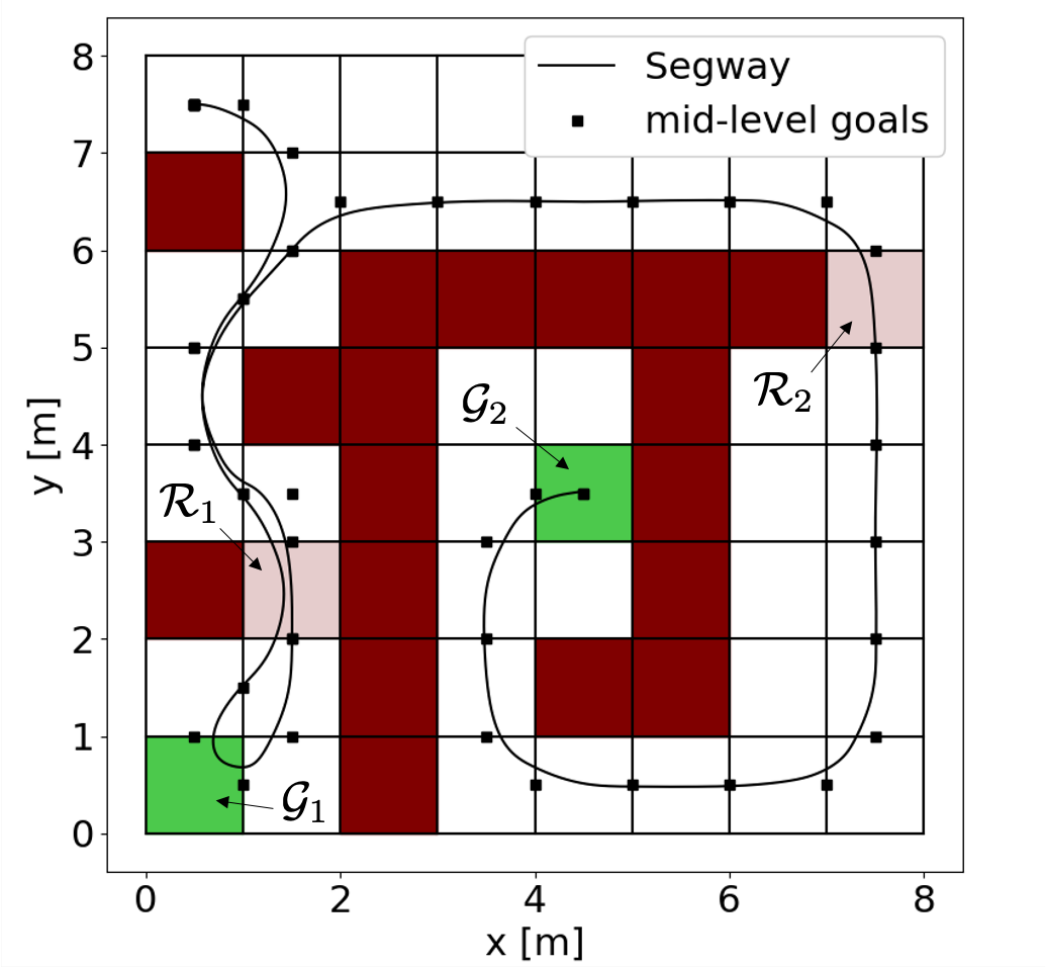



Figure 5.4: Closed-loop trajectory. The Segway first explores regions $\mathcal{R}_1$, which is traversable, and $\mathcal{G}_1$ that does not contain the science sample. Afterwards, it explores the traversable region $\mathcal{R}_2$ and it reaches $\mathcal{G}_2$.

the Segway receives a partial observation about the goal region $\mathcal{G}_i$ which is correct with probability 0.7, when one cell away and a perfect observations when the goal cell $\mathcal{G}_i$ is reached.

The state of the Segway is $x = [X, Y, \theta, v, \dot{\theta}, \psi, \dot{\psi}]$, where $(X - Y)$ represents the position of the center of mass, $(\theta, \dot{\theta})$ the heading angle and yaw rate, $v$ the velocity and $(\psi, \dot{\psi})$ the rod's angle and angular velocity. The control input $u = [T_l, T_r]$, where $T_l$ and $T_r$ are the torques to the left and right wheel motors, respectively. In order to implement the low-level CLF-CBF QP we used the following function:

$$h(e) = 1 - ||\text{diag}(v_h)(x - \bar{x})||_2^2, \tag{5.29}$$

where $v_h = [1/0.02, 1/0.02, 1/0.1, 1/0.1, 1/0.3, 1/0.1, 1/0.3]$ and $\bar{x} = [\bar{X}, \bar{Y}, \bar{\theta}, \bar{v}, \dot{\bar{\theta}}, \bar{\psi}, \dot{\bar{\psi}}]$ represents the state of the nominal system from (5.4). The candidate control Lya-

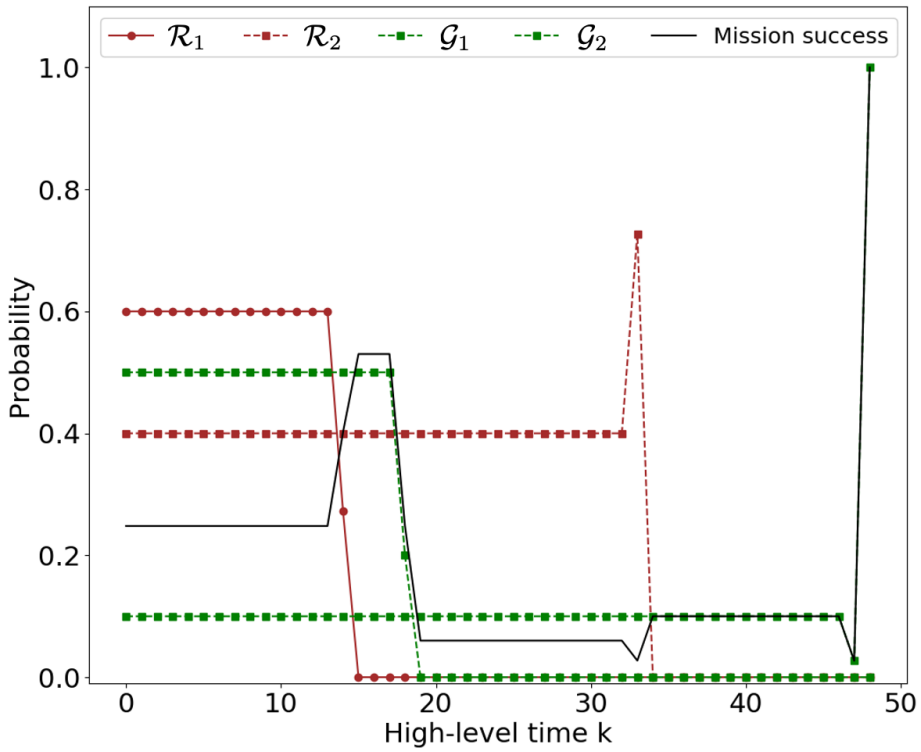



Figure 5.5: This figure shows the closed-loop probability of mission success, which equals the probability of satisfying the high-level specifications. Furthermore, we reported also the belief about regions $\mathcal{R}_1$ and $\mathcal{R}_2$ being travel about the goal regions $\mathcal{G}_1$ and $\mathcal{G}_2$ containing the science sample.

punov function is

$$V(e) = ||\text{diag}(v_v)(x - \bar{x})||_2^2,$$

where $v_v = [100, 100, 100, 100, 10000, 10000, 100]$ and in the CLF-QBF QP (5.8) we used $c_1 = 1$, $c_2 = 10$ and $\alpha_2(x) = x$. The planning model (5.4) is computed iteratively linearizing the Segway dynamics around the predicted MPC trajectory. This strategy is standard in MPC, for more details on the linearization strategy please refer to [125]. The stage cost $h(x, u) = x^\top Q u + u^\top R u$ and the tuning matrices are $Q = \text{dial}(0.1, 0.1, 0, 0, 10, 1, 10)$, $R = \text{diag}(0.01, 0.01)$ and $Q_f = \text{diag}(100, 100)$. Furthermore, we added an input rate cost with penalty $Q_{\text{rate}} = 0.1$ and a slack variable for the terminal constraint on the state $q_{t+N|i}$ with weight $Q_{\text{slack}} = \text{diag}(100, 100, 100)$. Finally, we approximated $\mathcal{S} = \{e = x - \bar{x} \in \mathbb{R}^n : h(e) \geq 0\} = \{e = x - \bar{x} \in \mathbb{R}^n : ||\text{diag}(v_v)(x - \bar{x})||_2^2 \leq 1\}$ with $\bar{\mathcal{S}} = \{e = x - \bar{\in}\mathbb{R}^n : ||\text{diag}(v_v)(x - \bar{x})||_\infty \leq 1\}$. This strategy allows us to write the MPC problem (5.14)

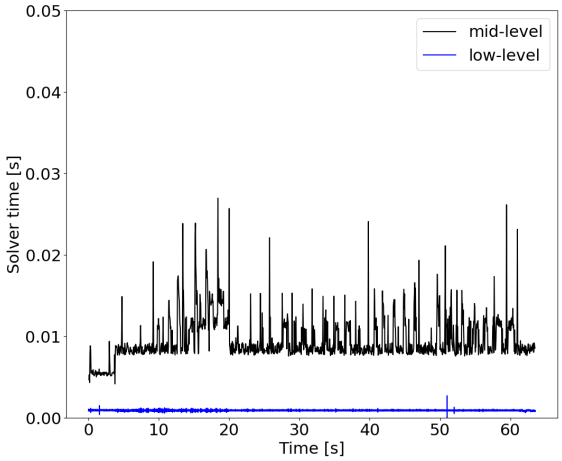



Figure 5.6: This figure shows the computational time associated with middle and low layers. It takes on average 12 ms to compute the mid-level control actions and less than 1 ms to compute the low-level commands. In this example the middle layer is discretized at 20 Hz and the lowest level at 1 kHz.

as a QP[5], which we solved using OSQP [145].

**Simulation**

We implemented the proposed strategy in our high-fidelity Robotic Operating System (ROS) simulator. Figure 5.4 shows the locations of the uncertain and goal regions. The code can be found at `https://github.com/DrewSingletary/segway_sim`, please check the `REAME.md` to replicate our results. In this example the goal regions $\mathcal{G}_1$ and $\mathcal{G}_2$ may contain a science sample with probability 0.6 and 0.4, respectively. Whereas, regions $\mathcal{R}_1$ and $\mathcal{R}_2$ may be traversable with probability 0.5 and 0.1, as shown in Figure 5.5.

Figure 5.4 shows the closed-loop trajectory of the Segway. We notice that the controller explores the uncertain region $\mathcal{R}_1$, which in this example is traversable and afterwards it reaches the goal regions $\mathcal{G}_1$. As shown in Figure 5.5, at the high-level time $k = 19$ the controller figures out that the goal cell $\mathcal{G}_1$ does not contain a science sample and, consequently, the probability of mission success drops. Afterwards, the controller steers the Segway to the traversable region $\mathcal{R}_2$ and to the goal regions $\mathcal{G}_2$. In this example, the goal regions $\mathcal{G}_2$ contains a science sample and the mission is completed successfully, as shown in Figure 5.5.

[5]Note that using $\mathcal{S}$ renders the MPC problem an SOCP, which is convex but computationally more demanding.

The mid-level is discretized for $T = 50$ ms and the low-level at 1 kHz. Figure 3.7 shows the computational time associated with mid-level and low-level control actions. It takes on average 12 ms to compute the mid-level control action $u_m(t)$ and less than 1 ms to compute the low-level action $u_l(t)$.

Finally, we analyse the evolution of the barrier function (5.29), which quantities the difference between the trajectory $x(t)$ of system (1.1) and the reference trajectory $\bar{x}(t)$ associated with nominal model[6] (5.4). We compared the proposed strategy with a naive MPC which is synthesized as in (5.14), but without taking into account the effect of the tracking error, i.e., we do not tighten the constraints and we set $x_{i|i} = x(t)$. Figure 5.7 shows the evolution of the barrier function for the proposed strategy and the naive MPC. We notice that when the low-level controller is not used, the barrier function becomes negative and in general has a lower magnitude. Therefore, this figure shows the advantage of the proposed hierarchical control architecture, where the low-level high-frequency controller is leveraged to track the reference trajectory. Indeed, this high-frequency feedback is used to modify the mid-level control actions, as shown in Figure 5.8. As discussed, the mid-level control action is updated at 20 Hz and the low-level input at 1 kHz. Notice that after the update of the mid-level input, the contribution of the low-level input towards the total control action is limited. However, as time progresses the linearization used to plan the reference trajectory is less and less accurate and for this reason, the magnitude of low-level controller increases.

**Experiment**

We implemented the proposed multi-rate hierarchical control strategy on the Segway-like robot shown in Figure 5.2. State estimation is based on wheel encoders and IMU data from a VectorNav VN-100. The state estimate and the low-level control action $u_l$ are computed at 800 Hz on the Segway, which is equipped with an ARM Cortex-A57 (quad-core) @ 2 GHz CPU running the ERIKA3 RTOS. On the other hand, the mid-level planner discretized at 20 Hz and the high-level decision maker run on a desktop with an Intel Core i7-8700 CPU (6-cores) @ 3.7 GHz CPU, which sends the reference trajectory $\bar{x}$ and the reference input $u_m$ via WiFi.

Figure 5.2 shows the location of the three uncertain regions $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$ which may be traversable with probability 0.9, 0.3 and 0.2, respectively. In this example, we assume that the goal region $\mathcal{G}_1$ contains the science sample with probability 1. Figure 5.10 shows the closed-loop trajectory. First, the controller explores region

[6] In this example the nominal model is computed iteratively linearizing the nonlinear dynamics.

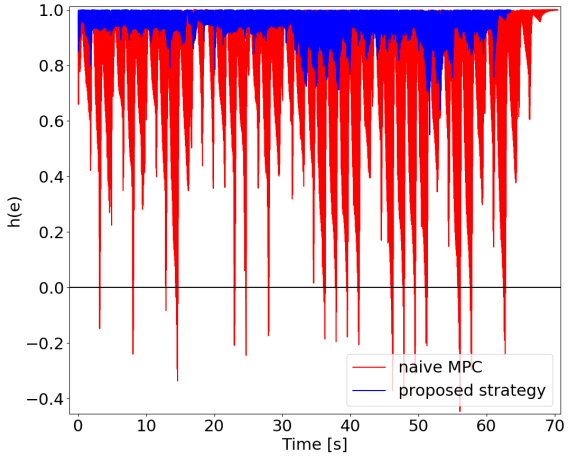



Figure 5.7: Comparison between the barrier function associated with the proposed strategy and a naive strategy MPC which is based on the linearized dynamics. As shown in the figure, when the low-level controller is not used the difference between the planner trajectory and the MPC trajectory grows and, as a results, the barrier function (5.29) becomes negative.

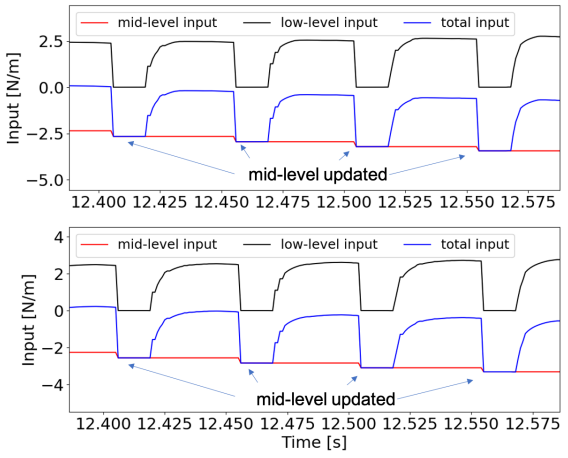



Figure 5.8: Input torque sent to the right (top) and left (bottom) motor over a period of 0.2 second. The mid-level input is updated at 20 Hz, whereas the low-level action is updated at 1 kHz. Notice that the total input is the summation of the low and mid-level inputs.

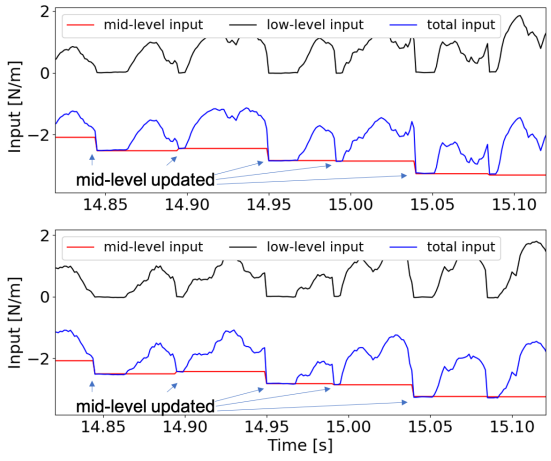



Figure 5.9: Experimental results. Input torque sent to the right (top) and left (bottom) motor over a period of 0.3 seconds. The mid-level input is updated at 20 Hz, whereas the low-level action is updated at 800 Hz. Notice that the total input is the summation of the low and mid-level inputs.

$\mathcal{R}_1$, which is not traversable and afterwards it steers the Segway towards regions $\mathcal{R}_2$ and $\mathcal{R}_3$. After collecting observations about the environment, the controller detects that region $\mathcal{R}_2$ is not traversable and that region $\mathcal{R}_3$ is free space that the Segway can navigate through to reach the goal region $\mathcal{G}_1$. A video of the experiment and comparison with a naive MPC can be found at `https://www.youtube.com/watch?v=Q-Mm0ywPh_I`.

Figure 5.11 shows the evolution of the control barrier function (5.29). We compare the proposed strategy with a naive MPC which is designed as in (5.14), but without robustifying the constraint sets and setting $x_{i|i} = x(t)$. Also in this case, when the high-frequency low-level controller is not active, the barrier function becomes negative meaning that the error $e$ does not belong to the safe set $\mathcal{E}$, i.e., $e(t) \notin \mathcal{E}$ for all $t \in \mathbb{R}_{0+}$. This result highlights the importance of the low-level high-frequency feedback from the CLF-CBF QP, which compensates for the model mismatch at the planning layer. Indeed, the MPC planner uses a linearized and discretized model, which is a first order approximation of the true dynamics. This approximation is accurate at the discrete time instances when the MPC input is computed. For this reason, the low-level CLF-CBF QP tracking controller computes the high-frequency component $u_l(t)$ which corrects the mid-level piecewise constant input $u_m(t)$, as shown in Figure 5.9.

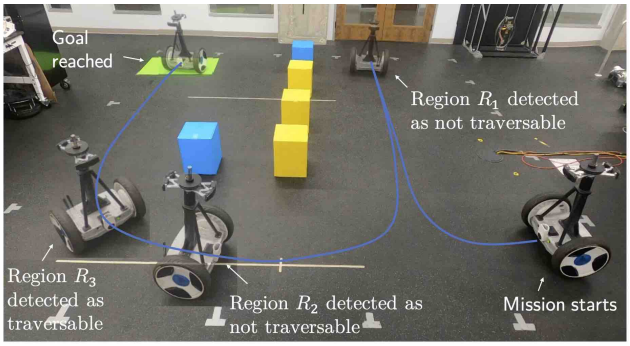



Figure 5.10: Closed-loop trajectory during the experiment. The Segway first explores the uncertain regions $\mathcal{R}_1$, $\mathcal{R}_1$ and $\mathcal{R}_1$ and then it reaches the goal region.

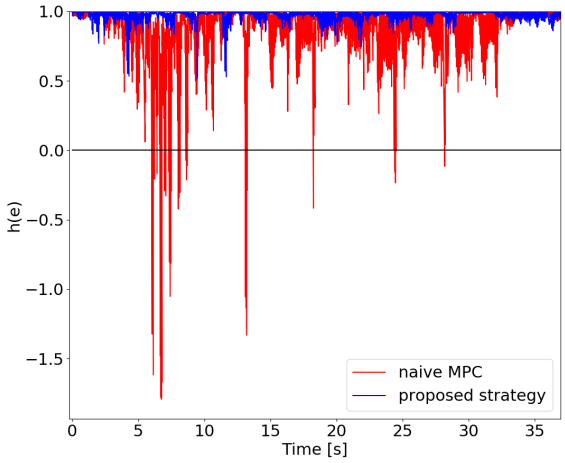



Figure 5.11: Experimental comparison between the barrier function associated with the proposed strategy and a naive MPC which is based on the linearized dynamics. Also in this case, when the low-level controller is not used, the difference between the planner trajectory and the MPC trajectory grows and, as a result, the barrier function (5.29) becomes negative.

## 5.2 Conclusion and future work

In this work, we presented four major types of safety regulator implementations, each with their own unique set of strengths and requirements. Chapter 2 showed the kinematic and model-free methods, Chapter 3 formulated the backup set methods, Chapter 4 demonstrated discete-time safety filtering, and Chapter 5 provided a unified framework for robotic systems with controllers and planners running at different rates.

We implemented these algorithms on real-world hardware, including industrial manipulators, quadrotors, racing drones, and a Segway robot, all with onboard sensing and control. No single method could have been used to achieve safety in all of these separate cases, and each method has a practical use in robotics.

In my opinion, the most relevant research area for future work would be a tighter integration of these safety regulators with perception. As of now, the methods require an estimate of the state or environment with associated uncertainty. By having tighter integration with the perception system, perhaps better performance and tighter safety guarantees could be achieved, and the number of safety failures due to localization errors would be minimized.

# BIBLIOGRAPHY


[1] A. Abate et al. “Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems”. In: *Automatica* 44.11 (2008), pp. 2724–2734.

[2] Masoud Abbaszadeh. “Is Lipschitz continuity preserved under sampled-data discretization?” In: *arXiv preprint arXiv:1612.08469* (2016).

[3] A. Agrawal and K. Sreenath. “Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation.” In: *Robotics: Science and Systems*. 2017.

[4] Amir Ali Ahmadi and Anirudha Majumdar. “DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization”. In: *2014 48th annual conference on information sciences and systems (CISS)*. IEEE. 2014, pp. 1–5.

[5] M. Ahmadi, H. Mojallali, and R. Wisniewski. “Guaranteed cost Hinf controller synthesis for switched systems defined on semi-algebraic sets”. In: *Nonlinear Analysis: Hybrid Systems* 11 (2014), pp. 37–56.

[6] M. Ahmadi, G. Valmorbida, and A. Papachristodoulou. “Safety verification for distributed parameter systems using barrier functionals”. In: *Systems & Control Letters* 108 (2017), pp. 33–39.

[7] M. Ahmadi et al. “Barrier Certificates for Assured Machine Teaching”. In: *2019 American Control Conference* (2019).

[8] M. Ahmadi et al. “Control theory meets POMDPs: A hybrid systems approach”. In: *arXiv preprint arXiv:1905.08095* (2019).

[9] M. Ahmadi et al. “Privacy verification in POMDPs via barrier certificates”. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018, pp. 5610–5615.

[10] M. Ahmadi et al. “Verification of uncertain POMDPs using barrier certificates”. In: *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2018, pp. 115–122.

[11] Anil Alan et al. “Safe controller synthesis with tunable input-to-state safe control barrier functions”. In: *IEEE Control Systems Letters* 6 (2021), pp. 908–913.

[12] Matthias Althoff, Olaf Stursberg, and Martin Buss. “Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization”. In: *2008 47th IEEE Conference on Decision and Control*. IEEE. 2008, pp. 4042–4048.

[13] Rajeev Alur et al. “Discrete abstractions of hybrid systems”. In: *Proceedings of the IEEE* 88.7 (2000), pp. 971–984.

[14] C. Amato and F. A. Oliehoek. “Scalable planning and learning for multiagent POMDPs”. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[15] Aaron D Ames et al. “Control barrier function based quadratic programs for safety critical systems”. In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 3861–3876.

[16] Aaron D Ames et al. “Control barrier functions: Theory and applications”. In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 3420–3431.

[17] David Angeli. “A Lyapunov approach to incremental stability properties”. In: *IEEE Transactions on Automatic Control* 47.3 (2002), pp. 410–421.

[18] Gianluca Antonelli and Stefano Chiaverini. “Kinematic control of platoons of autonomous vehicles”. In: *IEEE Transactions on Robotics* 22.6 (2006), pp. 1285–1292.

[19] Oktay Arslan and Panagiotis Tsiotras. “Use of relaxation methods in sampling-based algorithms for optimal motion planning”. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 2421–2428.

[20] P. Artzner et al. “Coherent measures of risk”. In: *Mathematical finance* 9.3 (1999), pp. 203–228.

[21] K. J. Astrom. “Optimal control of Markov decision processes with incomplete state estimation”. In: *Journal of mathematical analysis and applications* 10 (1965), pp. 174–205.

[22] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre. *Viability theory: new directions*. Springer Science & Business Media, 2011.

[23] Jerome Barraquand, Bruno Langlois, and J-C Latombe. “Numerical potential field techniques for robot path planning”. In: *IEEE transactions on systems, man, and cybernetics* 22.2 (1992), pp. 224–241.

[24] N. Bäuerle and J. Ott. “Markov decision processes with average-value-at-risk criteria”. In: *Mathematical Methods of Operations Research* 74.3 (2011), pp. 361–379.

[25] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*. Vol. 89. Springer, 2017.

[26] D. S. Bernstein et al. “The complexity of decentralized control of Markov decision processes”. In: *Mathematics of operations research* 27.4 (2002), pp. 819–840.

[27] F. Blanchini. “Set invariance in control”. In: *Automatica* 35.11 (1999), pp. 1747–1767.

[28] U. Borrmann et al. “Control barrier certificates for safe swarm behavior”. In: *IFAC-PapersOnLine* 48.27 (2015), pp. 68–73.

[29] Maxime Bouton, Jana Tumova, and Mykel J Kochenderfer. “Point-Based Methods for Model Checking in Partially Observable Markov Decision Processes.” In: *AAAI*. 2020, pp. 10061–10068.

[30] Alexander Broad, Todd Murphey, and Brenna Argall. “Highly parallelized data-driven MPC for minimal intervention shared control”. In: *arXiv preprint arXiv:1906.02318* (2019).

[31] Justin Carpentier et al. “The Pinocchio C++ library”. In: ().

[32] Margaret P Chapman et al. “A Risk-Sensitive Finite-Time Reachability Approach for Safety of Stochastic Dynamic Systems”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 2958–2963.

[33] Y. Chen et al. “Enhancing the performance of a safe controller via supervised learning for truck lateral control”. In: *arXiv preprint arXiv:1712.05506* (2017).

[34] Yong-bo Chen et al. “UAV path planning using artificial potential field method updated by optimal control theory”. In: *International Journal of Systems Science* 47.6 (2016), pp. 1407–1420.

[35] Yuxiao Chen, Andrew Singletary, and Aaron D Ames. “Guaranteed obstacle avoidance for multi-robot operations with limited actuation: a control barrier function approach”. In: *IEEE Control Systems Letters* 5.1 (2020), pp. 127–132. DOI: 10.1109/LCSYS.2020.3000748.

[36] Yuxiao Chen et al. “Backup Control Barrier Functions: Formulation and Comparative Study”. In: *arXiv preprint arXiv:2104.11332* (2021).

[37] Yuxiao Chen et al. “Data-driven computation of minimal robust control invariant set”. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018, pp. 4052–4058.

[38] Z. Chen, K. He, R. Kulperger, et al. “Risk measures and nonlinear expectations”. In: *Journal of Mathematical Finance* 3.03 (2013), p. 383.

[39] Y. Chow and M. Ghavamzadeh. “Algorithms for CVaR optimization in MDPs”. In: *Advances in neural information processing systems*. 2014, pp. 3509–3517.

[40] Y. Chow et al. “Risk-sensitive and robust decision-making: a cvar optimization approach”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1522–1530.

[41] Andrew Clark. “Control barrier functions for complete and incomplete information stochastic systems”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 2928–2935.

[42] Edmund M Clarke. “Model checking”. In: *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer. 1997, pp. 54–56.

[43] David Coleman et al. “Reducing the barrier to entry of complex robotic software: a moveit! case study”. In: *arXiv preprint arXiv:1404.3785* (2014).

[44] T. H. Cormen et al. *Introduction to algorithms*. MIT press, 2009.

[45] Wenceslao Shaw Cortez et al. “Control barrier functions for mechanical systems: Theory and application to robotic grasping”. In: *IEEE Transactions on Control Systems Technology* (2019).

[46] Patrick Cousot. “Abstract interpretation”. In: *ACM Computing Surveys (CSUR)* 28.2 (1996), pp. 324–328.

[47] Jan Dentler et al. “A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors”. In: *2016 IEEE conference on control applications (CCA)*. IEEE. 2016, pp. 519–525.

[48] B. Faverjon and P. Tournassoud. “A local based approach for path planning of manipulators with a high number of degrees of freedom”. In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. 1987, pp. 1152–1159. DOI: 10.1109/ROBOT.1987.1087982.

[49] Roy Featherstone. “A divide-and-conquer articulated-body algorithm for parallel O (log (n)) calculation of rigid-body dynamics. Part 1: Basic algorithm”. In: *The International Journal of Robotics Research* 18.9 (1999), pp. 867–875.

[50] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. “Maneuver-based motion planning for nonlinear systems with symmetries”. In: *IEEE transactions on robotics* 21.6 (2005), pp. 1077–1091.

[51] Randy Freeman and Petar V Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science & Business Media, 2008.

[52] Emilia Fridman. “A refined input delay approach to sampled-data control”. In: *Automatica* 46.2 (2010), pp. 421–427.

[53] Olivier Gay, David Coeurjolly, and Nathan Hurst. *Libaffa-C++ affine arithmetic library for GNU/Linux*. 2006.

[54] Veysel Gazi et al. “Aggregation, foraging, and formation control of swarms with non-holonomic agents using potential functions and sliding mode techniques”. In: *Turkish Journal of Electrical Engineering & Computer Sciences* 15.2 (2007), pp. 149–168.

[55] Shuzhi Sam Ge and Yun J Cui. “Dynamic motion planning for mobile robots using potential field method”. In: *Autonomous robots* 13.3 (2002), pp. 207–222.

[56] Dibyendu Ghosh et al. “Kinematic constraints based Bi-directional RRT (KB-RRT) with parameterized trajectories for robot path planning in cluttered environment”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8627–8633.

[57] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. “A fast procedure for computing the distance between complex objects in three-dimensional space”. In: *IEEE Journal on Robotics and Automation* 4.2 (1988), pp. 193–203.

[58] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. “Nonsmooth barrier functions with applications to multi-robot systems”. In: *IEEE control systems letters* 1.2 (2017), pp. 310–315.

[59] Thomas Gurriet et al. “An Online Approach to Active Set Invariance”. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018, pp. 3592–3599.

[60] Thomas Gurriet et al. “Realizable set invariance conditions for cyber-physical systems”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 3642–3649. DOI: 10.23919/ACC.2019.8815332.

[61] Thomas Gurriet et al. “Towards a framework for realizable safety critical control through active set invariance”. In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE. 2018, pp. 98–106. DOI: 10.1109/ICCPS.2018.00018.

[62] S. Haesaert et al. “Temporal logic control of pomdps via label-based stochastic simulation relations”. In: *IFAC-PapersOnLine* 51.16 (2018), pp. 271–276.

[63] Sofie Haesaert et al. “Temporal logic planning in uncertain environments with probabilistic roadmaps and belief spaces”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 6282–6287.

[64] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. “Dynamic programming for partially observable stochastic games”. In: *AAAI*. Vol. 4. 2004, pp. 709–715.

[65] Sylvia L Herbert et al. “FaSTrack: a modular framework for fast and guaranteed safe motion planning”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 1517–1522.

[66] Elie Hermand et al. “Constrained control of UAVs in geofencing applications”. In: *2018 26th Mediterranean Conference on Control and Automation (MED)*. IEEE. 2018, pp. 217–222.

[67] M W Hirsch, Stephen Smale, and Robert L Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. 3rd ed. Academic Press, 2012.

[68] Stefan Hrabar. “3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 807–814.

[69] Ali Jadbabaie, Jie Yu, and John Hauser. “Unconstrained receding-horizon control of nonlinear systems”. In: *IEEE Transactions on Automatic Control* 46.5 (2001), pp. 776–783.

[70] Mrdjan Jankovic. “Control barrier functions for constrained control of linear systems with input delay”. In: *2018 Annual American Control Conference (ACC)*. IEEE. 2018, pp. 3316–3321.

[71] Mrdjan Jankovic. “Control Lyapunov-Razumikhin functions and robust stabilization of time delay systems”. In: *IEEE Transactions on Automatic Control* 46.7 (2001), pp. 1048–1060.

[72] Luc Jaulin et al. “Interval analysis”. In: *Applied interval analysis*. Springer, 2001, pp. 11–43.

[73] Z. Jiang and Y. Wang. “A converse Lyapunov theorem for discrete-time systems with disturbances”. In: *Systems & control letters* 45.1 (2002), pp. 49–58.

[74] A. Jones, M. Schwager, and C. Belta. “Distribution temporal logic: Combining correctness with quality of estimation”. In: *52nd IEEE Conference on Decision and Control*. IEEE. 2013, pp. 4719–4724.

[75] O. Kanoun, F. Lamiraux, and P. Wieber. “Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task”. In: *IEEE Transactions on Robotics* 27.4 (2011), pp. 785–792. DOI: 10.1109/TRO.2011.2142450.

[76] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.

[77] Elia Kaufmann et al. “Beauty and the beast: Optimal methods meet learning for drone racing”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 690–696.

[78] Elia Kaufmann et al. “Deep drone racing: Learning agile flight in dynamic environments”. In: *Conference on Robot Learning*. PMLR. 2018, pp. 133–145.

[79] Wisama Khalil. “Dynamic modeling of robots using recursive newton-euler techniques”. In: *ICINCO2010*. 2010.

[80] Oussama Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE. 1985, pp. 500–505.

[81] Oussama Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[82] Sven Koenig and Reid G Simmons. “Risk-sensitive planning with probabilistic decision graphs”. In: *Principles of Knowledge Representation and Reasoning*. Elsevier. 1994, pp. 363–373.

[83] S. Kolathaya and A. D. Ames. “Input-to-State Safety With Control Barrier Functions”. In: *IEEE control systems letters* 3.1 (2019), pp. 108–113.

[84] Shishir Kolathaya. “Energy based Control Barrier Functions for Robotic Systems”. In: (Aug. 2020). DOI: 10.36227/techrxiv.12831503.v1. URL: https://www.techrxiv.org/articles/preprint/Energy_based_Control_Barrier_Functions_for_Robotic_Systems/12831503.

[85] Shishir Kolathaya. “Local stability of PD controlled bipedal walking robots”. In: *Automatica* 114 (2020), p. 108841. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2020.108841.

[86] Shishir Kolathaya and Aaron D Ames. “Input-to-state safety with control barrier functions”. In: *IEEE control systems letters* 3.1 (2018), pp. 108–113.

[87] Shreyas Kousik et al. “Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots”. In: *arXiv preprint arXiv:1809.06746* (2018).

[88] James J Kuffner and Steven M LaValle. “RRT-connect: An efficient approach to single-query path planning”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[89] M. Lahijanian et al. “Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 3227–3232.

[90] Shupeng Lai et al. “A robust online path planning approach in cluttered environments for micro rotorcraft drones”. In: *Control Theory and Technology* 14.1 (2016), pp. 83–96.

[91] Chiara Talignani Landi et al. “Safety barrier functions for human-robot interaction with industrial manipulators”. In: *2019 18th ECC*. IEEE. 2019, pp. 2565–2570.

[92] Grüne Lars and P Jürgen. *Nonlinear model predictive control theory and algorithms*. 2011.

[93] Min Cheol Lee and Min Gyu Park. “Artificial potential field based path planning for mobile robots using a virtual obstacle concept”. In: *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*. Vol. 2. IEEE. 2003, pp. 735–740.

[94] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. "Geometric tracking control of a quadrotor UAV on SE(3)". In: *49th IEEE Conference on Decision and Control (CDC)*. 2010, pp. 5420–5425. DOI: 10.1109/CDC.2010.5717652.

[95] Randall J LeVeque. "Finite difference methods for differential equations". In: *Draft version for use in AMath* 585.6 (1998), p. 112.

[96] Guanghui Li et al. "An efficient improved artificial potential field based regression search method for robot path planning". In: *2012 IEEE International Conference on Mechatronics and Automation*. IEEE. 2012, pp. 1227–1232.

[97] Yucong Lin and Srikanth Saripalli. "Sampling-based path planning for UAV collision avoidance". In: *IEEE Transactions on Intelligent Transportation Systems* 18.11 (2017), pp. 3179–3192.

[98] Sikang Liu et al. "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments". In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1688–1695.

[99] A. Majumdar and M. Pavone. "How should a robot assess risk? Towards an axiomatic theory of risk in robotics". In: *Robotics Research*. Springer, 2020, pp. 75–84.

[100] Daniel Mellinger and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *Proc. IEEE ICRA*. 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[101] J. V. Messias, M. Spaan, and P. U. Lima. "Efficient offline communication policies for factored multiagent POMDPs". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1917–1925.

[102] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games". In: *IEEE Transactions on automatic control* 50.7 (2005), pp. 947–957.

[103] Tamas G Molnar et al. "Model-free safety-critical control for robotic systems". In: *IEEE Robotics and Automation Letters* 7.2 (2021), pp. 944–951. DOI: 10.1109/LRA.2021.3135569.

[104] Hyungpil Moon et al. "Challenges and implemented technologies used in autonomous drone racing". In: *Intelligent Service Robotics* 12.2 (2019), pp. 137–148.

[105] V. Murashov, F. Hearl, and J. Howard. "Working safely with robot workers: Recommendations for the new workplace". In: *Journal of occupational and environmental hygiene* 13.3 (2016), pp. D61–D71.

[106] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[107] Mitio Nagumo. “Über die lage der integralkurven gewöhnlicher differentialgleichungen”. In: *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series* 24 (1942), pp. 551–559.

[108] Q. Nguyen et al. “3d dynamic walking on stepping stones with control barrier functions”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 827–834.

[109] Quan Nguyen and Koushil Sreenath. “Exponential control barrier functions for enforcing high relative-degree safety-critical constraints”. In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 322–328.

[110] Petter Nilsson et al. “Toward specification-guided active mars exploration for cooperative robot teams”. In: *Robotics: Science and Systems (RSS)* (2018).

[111] Petter Ogren, Magnus Egerstedt, and Xiaoming Hu. “A control Lyapunov function approach to multi-agent coordination”. In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*. Vol. 2. IEEE. 2001, pp. 1150–1155.

[112] F. A. Oliehoek and M. T. J. Spaan. “Tree-based solution methods for multiagent POMDPs with delayed communication”. In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[113] Sylvie CW Ong et al. “Planning under uncertainty for robotic tasks with mixed observability”. In: *The International Journal of Robotics Research* 29.8 (2010), pp. 1053–1068.

[114] M. Ono et al. “Chance-constrained dynamic programming with application to risk-aware robotic space exploration”. In: *Autonomous Robots* 39.4 (2015), pp. 555–571.

[115] Gábor Orosz and Aaron D Ames. “Safety Functionals for Time Delay Systems”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 4374–4379.

[116] Daniel Pickem et al. “The robotarium: A remotely accessible swarm robotics research testbed”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1699–1706.

[117] S. Prajna, A. Jadbabaie, and G. J. Pappas. “A framework for worst-case and stochastic safety verification using barrier certificates”. In: *IEEE Transactions on Automatic Control* 52.8 (2007), pp. 1415–1428.

[118] L. Prashanth. “Policy gradients for CVaR-constrained MDPs”. In: *International Conference on Algorithmic Learning Theory*. Springer. 2014, pp. 155–169.

[119] D. V. Pynadath and M. Tambe. “The communicative multiagent team decision problem: Analyzing teamwork theories and models”. In: *Journal of artificial intelligence research* 16 (2002), pp. 389–423.

[120] Manuel Rauscher, Melanie Kimmel, and Sandra Hirche. "Constrained robot control using control barrier functions". In: *2016 IEEE/RSJ International Conference on IROS*. IEEE. 2016, pp. 279–285.

[121] Jenna Reher and Aaron D Ames. "Dynamic Walking: Toward Agile and Efficient Bipedal Robots". In: *Annual Reviews* (2020).

[122] Jing Ren, Kenneth A McIsaac, and Rajnikant V Patel. "Modified Newton's method applied to potential field-based navigation for mobile robots". In: *IEEE Transactions on Robotics* 22.2 (2006), pp. 384–391.

[123] R Tyrrell Rockafellar and Stanislav Uryasev. "Conditional value-at-risk for general loss distributions". In: *Journal of banking & finance* 26.7 (2002), pp. 1443–1471.

[124] R Tyrrell Rockafellar, Stanislav Uryasev, et al. "Optimization of conditional value-at-risk". In: *Journal of risk* 2 (2000), pp. 21–42.

[125] Ugo Rosolia and Francesco Borrelli. "Learning how to autonomously race a car: a predictive control approach". In: *IEEE Transactions on Control Systems Technology* (2019).

[126] Ugo Rosolia et al. "Time-Optimal Navigation in Uncertain Environments with High-Level Specifications". In: *To appear on the IEEE Conference on Decision and Control (CDC), arXiv preprint arXiv:2103.01476* (2021).

[127] Tomáš Rouček et al. "Darpa subterranean challenge: Multi-robotic exploration of underground environments". In: *International Conference on Modelling and Simulation for Autonomous Systesm*. Springer. 2019, pp. 274–290.

[128] A. Ruszczyński. "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical programming* 125.2 (2010), pp. 235–261.

[129] Takashi Sakai. "On Riemannian manifolds admitting a function whose gradient is of constant norm". In: *Kodai Mathematical Journal* 19.1 (1996), pp. 39–51.

[130] Cesar Santoyo, Maxence Dutreix, and Samuel Coogan. "A barrier function approach to finite-time stochastic system verification and control". In: *arXiv preprint arXiv:1909.05109* (2019).

[131] John Schulman et al. "Motion planning with sequential convex optimization and convex collision checking". In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1251–1270.

[132] S. Seuken and S. Zilberstein. "Formal models and algorithms for decentralized decision making under uncertainty". In: *Autonomous Agents and Multi-Agent Systems* 17.2 (2008), pp. 190–250.

[133] G. Shani, J. Pineau, and R. Kaplow. "A survey of point-based POMDP solvers". In: *Autonomous Agents and Multi-Agent Systems* 27.1 (2013), pp. 1–51.

[134] Yifei Simon Shao et al. “Reachability-based Trajectory Safeguard (RTS): A Safe and Fast Reinforcement Learning Safety Layer for Continuous Control”. In: *arXiv preprint arXiv:2011.08421* (2020).

[135] R. Sharan and J. Burdick. “Finite state control of POMDPs with LTL specifications”. In: *2014 American Control Conference*. IEEE. 2014, pp. 501–508.

[136] Bruno Siciliano. “Kinematic control of redundant robot manipulators: A tutorial”. In: *Journal of intelligent and robotic systems* 3.3 (1990), pp. 201–212.

[137] S. Singh et al. “A framework for time-consistent, risk-sensitive model predictive control: Theory and algorithms”. In: *IEEE Transactions on Automatic Control* (2018).

[138] Sumeet Singh et al. “Robust online motion planning via contraction theory and convex optimization”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 5883–5890.

[139] Sumeet Singh et al. “Robust tracking with model mismatch for fast and safe planning: an SOS optimization approach”. In: *International Workshop on the Algorithmic Foundations of Robotics*. Springer. 2018, pp. 545–564.

[140] Andrew Singletary, Yuxiao Chen, and Aaron D Ames. “Control barrier functions for sampled-data systems with input delays”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 804–809. DOI: 10.1109/CDC42340.2020.9304281.

[141] R. D. Smallwood and E. J. Sondik. “The optimal control of partially observable Markov processes over a finite horizon”. In: *Operations research* 21.5 (1973), pp. 1071–1088.

[142] Yunlong Song et al. “Autonomous Drone Racing with Deep Reinforcement Learning”. In: *arXiv preprint arXiv:2103.08624* (2021).

[143] Eduardo D Sontag. “A Lyapunov-like characterization of asymptotic controllability”. In: *SIAM journal on control and optimization* 21.3 (1983), pp. 462–471.

[144] M. Srinivasan, S. Coogan, and M. Egerstedt. “Control of Multi-Agent Systems with Finite Time Control Barrier Certificates and Temporal Logic”. In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 1991–1996.

[145] Bartolomeo Stellato et al. “OSQP: An operator splitting solver for quadratic programs”. In: *2018 UKACC 12th International Conference on Control (CONTROL)*. IEEE. 2018, pp. 339–339.

[146] Paulo Tabuada and George J Pappas. “Linear time logic control of discrete-time linear systems”. In: *IEEE Transactions on Automatic Control* 51.12 (2006), pp. 1862–1877.

[147] A. J. Taylor and A. D. Ames. “Adaptive Safety with Control Barrier Functions”. In: *2020 American Control Conference (ACC)*. 2020, pp. 1399–1405.

[148] Ben Tearle et al. “A predictive safety filter for learning-based racing control”. In: *arXiv preprint arXiv:2102.11907* (2021).

[149] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[150] Jesus Tordesillas, Brett T Lopez, and Jonathan P How. “Faster: Fast and safe trajectory planner for flights in unknown environments”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 1934–1940.

[151] Jesus Tordesillas, Brett T. Lopez, and Jonathan P. How. “FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments”. In: *arXiv e-prints*, arXiv:1903.03558 (Mar. 2019), arXiv:1903.03558. arXiv: 1903.03558 [cs.RO].

[152] Jesus Tordesillas et al. “Real-time planning with multi-fidelity models for agile flights in unknown environments”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 725–731.

[153] Gino Van Den Bergen. “Proximity queries and penetration depth computation on 3D game objects”. In: *Game developers conference*. Vol. 170. 2001.

[154] M. Vasic and A. Billard. “Safety issues in human-robot interactions”. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 197–204.

[155] C.-I. Vasile et al. “Control in belief space with temporal logic specifications”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 7419–7424.

[156] *Video of the experiments.* https://vimeo.com/468799586.

[157] A. Wang, A. M Jasour, and B. Williams. “Non-gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty”. In: *IEEE Robotics and Automation Letters* (2020).

[158] Li Wang, Aaron D Ames, and Magnus Egerstedt. “Safety barrier certificates for collisions-free multirobot systems”. In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 661–674.

[159] Yue Wang, Swarat Chaudhuri, and Lydia E Kavraki. “Bounded policy synthesis for POMDPs with safe-reachability objectives”. In: *arXiv preprint arXiv:1801.09780* (2018).

[160] Charles W Warren. “Global path planning using artificial potential fields”. In: *1989 IEEE International Conference on Robotics and Automation*. IEEE Computer Society. 1989, pp. 316–317.

[161] T. Wongpiromsarn, U. Topcu, and R. M. Murray. "Receding Horizon Temporal Logic Planning". In: *IEEE Transactions on Automatic Control* 57.11 (2012), pp. 2817–2830.

[162] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. "Receding horizon control for temporal logic specifications". In: *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*. 2010, pp. 101–110.

[163] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. "Synthesis of control protocols for autonomous systems". In: *Unmanned Systems* 1.01 (2013), pp. 21–39.

[164] Ji Xiang, Congwei Zhong, and Wei Wei. "General-weighted least-norm control for redundant manipulators". In: *IEEE Transactions on Robotics* 26.4 (2010), pp. 660–669.

[165] Huan Xu and Shie Mannor. "Distributionally robust Markov decision processes". In: *Advances in Neural Information Processing Systems*. 2010, pp. 2505–2513.

[166] X. Xu et al. "Robustness of control barrier functions for safety critical control". In: *IFAC-PapersOnLine* 48.27 (2015), pp. 54–61.

[167] Shuyou Yu et al. "Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems". In: *Systems & Control Letters* 62.2 (2013), pp. 194–200.

[168] Songyuan Zhang et al. "Model predictive control based dynamic geofence system for unmanned aerial vehicles". In: *AIAA Information Systems-AIAA Infotech@ Aerospace*. 2017, p. 0675.

[169] Ping Zhao, Yu Kang, and Yun-Bo Zhao. "A Brief Tutorial and Survey on Markovian Jump Systems: Stability and Control". In: *IEEE Systems, Man, and Cybernetics Magazine* 5.2 (2019), pp. 37–C3.

[170] Qing-Chang Zhong. *Robust control of time-delay systems*. Springer Science & Business Media, 2006.

[171] Dingjiang Zhou and Mac Schwager. "Vector field following for quadrotors using differential flatness". In: *Proc. IEEE ICRA* (2014), pp. 6567–6572. DOI: 10.1109/ICRA.2014.6907828.